ESCUELA POLITÉCNICA DEL EJÉRCITO

FACULTAD DE SISTEMAS E INFORMÁTICA

GESTION DE CALIDAD DE SOFTWARE

Previo a la obtención del titulo de:

Ingeniero de Sistema e Informática

DIANA ARIAS
ESTEBAN MONTESDEOCA

Sangolquí, Enero 2005

CERTIFICACION

Certifico que el presente trabajo fue realizado en su totalidad por la Srta. DIANA ELIZABETH ARIAS LOPEZ y el Sr. DIEGO ESTEBAN MONTESDEOCA STACEY como requerimiento parcial a la obtención del título de INGENIEROS EN SISTEMAS E INFORMÁTICA.

Sangolquí, 02 de Diciembre de 2004

_____

Ing. Jofre Villagómez

Índice

Índice Figuras

Índice de tablas

Índice de Cuadros

Resumen

El concepto de calidad total ha evolucionado a través de la historia hasta volverse en nuestros días un tema importante al momento de promocionar un producto o servicio en el mercado. El software ha entrado en una fuerte transición desde el desarrollo artesanal hasta el seguimiento de procedimientos estandarizados y normadas que permitan obtener mejor desempeño del producto desarrollado, la adecuación del concepto de calidad total a proceso de la ingeniería de software es una actividad ampliamente discutida y muy poco aceptada entre los grupos de desarrollo de software.

Las técnicas de calidad aplicadas a los ámbitos empresariales y de manufactura ha sido adaptadas al software por medio de estándares y modelos que permiten enfocar la calidad en aspectos específicos de software resaltando siempre la idea fundamental de la calidad total; *Mejoramiento Continuo*.

La evaluación de la calidad es, en esencia, subjetiva, por lo que es necesario recurrir a indicadores para que presenten de una manera mas directa el impacto de cada actividad realizada, por la ingeniería de software, en las expectativas del cliente.

Introducción

Las herramientas de software, en la actualidad, se han convertido en parte de las actividades cotidianas que cada persona realiza en su lugar de trabajo, es así que existen varios programas que se distribuyen de diferentes maneras (gratuitos, bajo licencia), y enfocados a diversas áreas volviendo difícil la labor de escoger un programa que se adecue a las necesidades informáticas. Este incremento en la oferta de software ha generado bajas en el precio de producto de software a demás de desconfianza entre los consumidores del mismo.

Debido a que el desarrollo del software de nuestro país no se encuentra en una etapa avanzada, sino que mas bien en sus inicios como un proceso parametrizado, el tema de la calidad del producto de software no es un tema al que se preste mayor interés, a pesar de esto, algunas de las empresa que tienen algunos años en el ámbito del desarrollo de software sienten la necesidad de entregar calidad en su producto a los clientes.

Es por esta razón que se ha visto la necesidad de determinar mecanismos que permitan evaluar la calidad del software en cada una de sus etapas asegurando su efectividad sustentado por normas internacionales.

I. Antecedentes

La Calidad Total es más que una filosofía empresarial originada en Japón y que parte del concepto de "calidad de producto". Se refleja en el cumplimiento de las necesidades del cliente por medio del producto. *"La Calidad es total porque comprende todos y cada uno, de los aspectos de la organización, porque involucra y compromete a todas y cada una de las personas de la organización."*[1]. La forma en la que la Calidad Total puede ser aplicada en todas las áreas de una organización se respalda en una metodología que es una secuencia de procesos que permiten alcanzar la calidad.

En el mundo de la informática, la calidad puede traducirse de manera subjetiva como el cumplimiento de los requerimientos, pero este cumplimiento puede ser únicamente evaluado por el usuario final. Como consecuencia de la evolución en las necesidades informáticas y a la sofisticación de herramientas de desarrollo, se hace indispensable definir parámetros de calidad en el software, este proceso ha encontrado ciertos inconvenientes como es el hecho de que *"... la producción industrial de software es una actividad relativamente joven con respecto a otras actividades productivas, mercantiles o industriales"*[2] ocasionando que la determinación de estos aspectos evaluables respecto al software sean diversos.

Por las necesidades planteadas anteriormente y considerando que la existencia del software como un producto es una industria joven, se ve la necesidad de plantear una metodología que asegure la calidad del software.

[1] Fuente: www.geocities.com/CollegePark/Lab/2960/TQM1.htm
[2] Dr. Vladimir Estivill-Castro, www.lania.mx

II. JUSTIFICACIÓN

La importancia que tiene la aceptación de un cliente a un producto de software ofrecido ha obligado a que los ingenieros de software dirijan sus esfuerzos a la elaboración de un software de calidad.

Desde el punto de vista de las fases de ingeniería del software, es importante contar con un documento que sustente las fases de desarrollo, los mismos que determinen el tipo de producto que se debe esperar a la finalización de cada fase, así como la manera de determinar la calidad que tiene dicho producto. Dicho de otra manera, el proceso de gestión de calidad del software debe visualizarse como un conjunto de procesos y productos que llevan a la culminación de un software de calidad, diseñado bajo la utilización de normas o procedimientos para satisfacer eficientemente los requerimientos influyendo directamente en el grado de efectividad del software.

III. Alcance

1. Describir y analizar la Calidad Total (Calidad, Mejoramiento y Aseguramiento) en el ambiente del desarrollo de software.
2. Determinar las técnicas para la gestión de la calidad del software para las etapas de la Ingeniería de Software.
3. Especificar una guía para la Gestión de Calidad en las fases de la Ingeniería del Software.
4. Aplicar las técnicas determinadas al caso de estudio *"Sistema Integrado de Gestión Hospitalaria para el Hospital Pediátrico Baca Ortiz"* en las fases de Análisis y Diseño.

IV. Objetivo

a. Objetivo General

Determinar mecanismos que permitan realizar la Gestión de Calidad del Software en las fases de análisis y diseño dentro del proceso de la Ingeniería del Software, la misma que sirva como referencia para los ingenieros del software a través de una guía que permita obtener directrices en las etapas de la Ingeniería de Software.

b. Objetivos Específicos

- Analizar los métodos de gestión de Calidad Total y su relación con la Ingeniería del Software.
- Estudiar las técnicas para la gestión de calidad del software en las etapas de la Ingeniería de Software.
- Determinar los mecanismos que permitan entregar y asegurar la calidad del software.
- Aplicar los mecanismos para el aseguramiento de la calidad del software a las fases de análisis y diseño de la Ingeniería del Software a un caso de estudio.

CAPITULO I: ADMINISTRACIÓN DE LA CALIDAD TOTAL

Introducción

La Calidad es concebida como el cumplimiento de los deseos del cliente de una manera óptima, este cumplimiento se reflejan en un conjunto de características que satisfacen las necesidades del cliente en un producto o servicio.

La necesidad de entregar calidad a productos o servicios es un concepto que ha tenido una lenta evolución a través de los años iniciando como el entregar productos y servicios a un precio justo hasta llegar ahora al concepto de calidad en procesos y productos, lo que se conoce como la Calidad Total, que no es sino un proceso de mejoramiento continuo que abarca todas las etapas de realización o entrega de un producto o servicio. (Fig. 1.1)

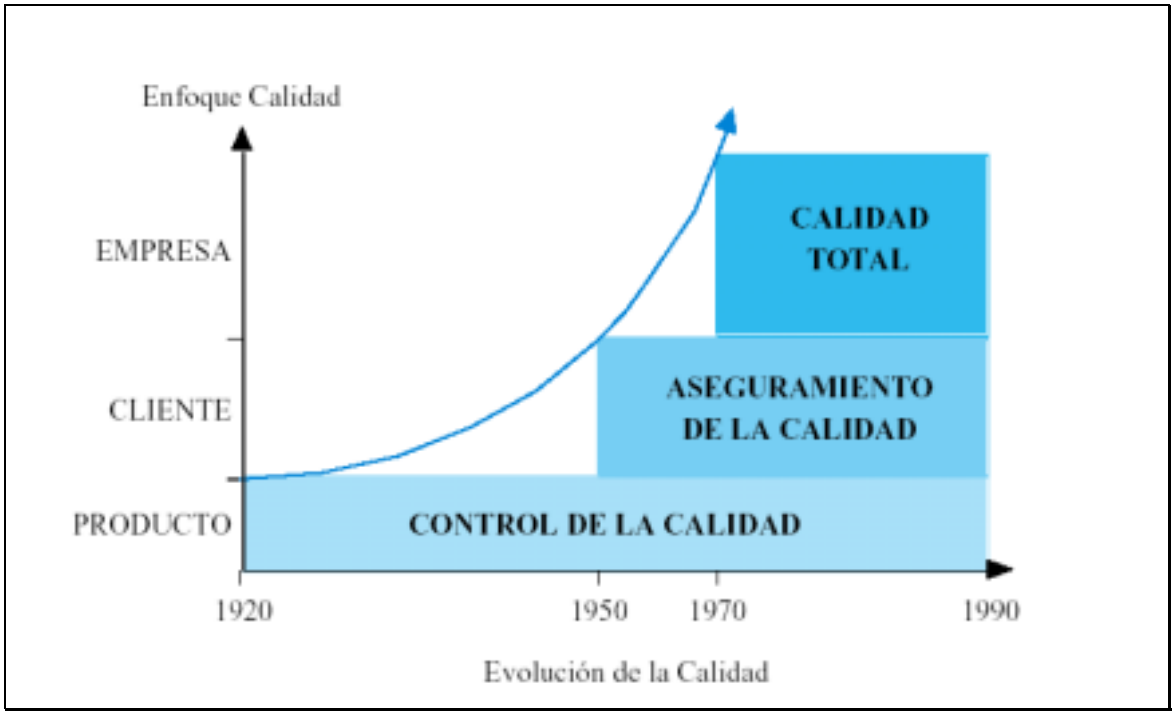


Fig. 1.1 Evolución de la Calidad

Deming fue el primero en tratar el tema de la Calidad Total realizando, con sus teorías, cambios notables en la economía japonesa. Estos cambios cautivaron la atención de los empresarios estadounidenses que aprovecharon también los conocimientos de Deming, instituyendo sus teorías como *"Teorías Universales de Calidad Total"*.

1.1 Calidad Total

"La Calidad es total porque comprende todos y cada uno, de los aspectos de la organización, porque involucra y compromete a todas y cada una de las personas de la organización"[3]. La calidad total no es otra cosa que una manera de mirar a una organización como un todo, y relacionar su rendimiento como el funcionamiento conjunto de todos sus componentes para lograr la satisfacción del cliente con un producto o un servicio.

La filosofía de la Calidad Total proporciona una concepción global que fomenta el mejoramiento continuo de los procesos y productos en la organización y la inclusión de todos sus miembros centrándose en la satisfacción tanto del cliente interno como externo. Es importante entender que la calidad en cada actividad dentro de un proceso permite llegar más fácilmente a la calidad del producto o servicio final, sin embargo, esto no garantiza calidad al final del proceso.

La experiencia del Dr. Edwards Deming en Japón y Estados Unidos permitió que desarrollara un conjunto de principios agrupados en Catorce Puntos y Siete Pecados Mortales para llegar a la Calidad Total:

[3] Fuente: www.geocities.com/CollegePark/Lab/2960/TQM1.htm

- **Los Catorce Puntos:**

1. Crear constancia en los propósitos
2. Adoptar una nueva filosofía
3. Terminar con la práctica de comprar a los más bajos precios
4. Establecer liderazgo
5. Eliminar slogans vacíos
6. Eliminar cuotas numéricas
7. Establecer entrenamiento dentro del trabajo
8. Desechar temores
9. Romper barreras entre departamentos
10. Tomar acciones para lograr la transformación
11. Mejorar constantemente y siempre el proceso de producción y servicio
12. Desistir de la dependencia en la inspección en masa
13. Remover barreras para apreciar la mano de obra
14. Reeducar vigorosamente

- **Los Siete Pecados Mortales**

Demming propone estos siete puntos como las más grandes falencias en las que una empresa u organización puede incurrir al gestionar calidad.

1. Carencia de constancia en los propósitos
2. Enfatizar ganancias a corto plazo y dividendos inmediatos
3. Evaluación de rendimiento, calificación de mérito o revisión anual
4. Movilidad de la administración principal
5. Manejar una organización basado solamente en las figuras visibles
6. Costos médicos excesivos
7. Costos de garantía

La aplicación de estos principios sirve a las empresas para mantenerse a la par con los cambiantes ambientes económicos a los que se enfrentan, pero estos puntos no garantizan el éxito en si mismos, sino que requieren de un cambio de mentalidad en el tratamiento de los recursos de las empresas (Humanos, Materia Prima, etc.)

Algunos de los pilares en los que se fundamenta la Calidad Total son los siguientes:

- **Orientar la organización hacia el cliente.** Satisfacer los requerimientos del cliente es lo principal. Con este objetivo, la organización debe girar en torno a los procesos que son importantes para este fin y que aportan valor añadido. Esto implica superar la visión clásica de que la responsabilidad sobre la Calidad es exclusiva de los departamentos encargados del producto o servicio.
- **Ampliar el concepto de cliente.** Es necesario concebir a la organización como un sistema integrado por proveedores y clientes internos
- **Poseer liderazgo en costos.** La calidad cuesta, pero es más cara la no-calidad. Si se trata de centrar la atención en las necesidades y expectativas del cliente, éstas serán mejor atendidas si el costo trasladado al cliente es más bajo. Se hace necesario, por tanto, reducir los costos de no-calidad.
- **Gestionar basándose en la prevención.** La idea subyacente es la de hacer las cosas bien a la primera. Es mejor que las acciones clásicas de detectar y corregir. Se reduce la necesidad de aplicar acciones de control, minimizando los costos.

- **Potenciar el factor humano.** La calidad no se controla, se hace. Y es realizada por las personas que conforman la organización. Todas, sin excepción. Por lo tanto es imprescindible establecer una gestión de los recursos humanos desde la motivación para la Calidad y la participación.
- **Mejora permanente continua.** La Calidad ha de ser concebida como un horizonte, no como una meta. No se llega a la Calidad Total, se persigue un horizonte que se amplía a medida que se avanza. Aquí está implícita esta idea de mejora continua.

La planificación de la Calidad de un producto o un servicio debe estar presente desde la etapa inicial del proceso, ya que seria demasiado tarde intentar introducir la calidad en fases posteriores, por eso es tan importante, que el diseño de los servicios o productos sea resultado de un trabajo en equipo. Así mismo, los productos obtenidos en la finalización de cada etapa de un proyecto o un proceso, son presentados en forma de entregables para la siguiente etapa, respecto a los cuales se evalúa la calidad, por lo que es necesario realizar revisiones periódicas de dichos entregables.

Es necesario para asegurar la calidad, identificar factores que permitan obtener parámetros respecto a los diferentes aspectos que giran en torno a un producto o servicio para lograr que la Calidad se convierta en un valor agregado que sea objeto de evaluación por el cliente interno y externo.

1.2 Técnicas de Gestión de Calidad Total

La gestión de la calidad es una actividad que se vuelve algo subjetiva ya que la calidad significa cosas distintas para diferentes personas. Hay dos tipos de calidad en un proyecto: La calidad de los entregables físicos que son producidos y la calidad del servicio brindado.

La gestión de calidad se basa en paradigmas que permiten construir una técnica tales como: Administración Total de la Calidad (TQM), Desarrollo de Funciones de calidad (QFD) o Just in Time, las mismas que serán explicadas a continuación.

1.2.1. Administración Total de la Calidad (TQM)

TQM es un estilo de administración participativo que comprende a todo el personal comprometiéndolo con la satisfacción al cliente. Reemplaza la administración vertical (TOP-DOWN) con una administración descentralizada orientada al cliente. TQM es un sistema de administración integrado para la creación e implementación de un proceso de mejora continua asumiendo que el noventa por ciento de los problemas son resultado de los procesos y no de los empleados.

El entender y cumplir las expectativas del cliente es una propuesta retadora, y requiere procesos que soporten un progreso continuo hacia la meta de cumplir las expectativas del cliente todo el tiempo.

Existe un gran beneficio para las organizaciones que adoptan esta forma de pensar en todas las áreas de la organización. Se obtendrán clientes satisfechos que querrán seguir haciendo negocios. UN CLIENTE SATISFECHO ES LA MANERA MAS BARATA DE GENERAR INGRESOS Y BENEFICIOS.

Con la aplicación de esta técnica, se posibilita la expansión de negocios sin la necesidad de contratar nuevo personal ya que el personal será mas productivo al contar con una documentación de procesos para la realización de su trabajo sin existir mal entendimiento sobre lo que los clientes desean. Los clientes no esperan que se sobrepase sus requerimientos por que saben

que eso tiene un costo para el productor y que se evidenciará en el costo hacia el cliente.

Los procesos relacionados con TQM son los siguientes:

- Hallar los requerimientos del consumidor.
- Reducción de redundancia en las etapas de un proceso de desarrollo.
- Ideología de “Justo a Tiempo”.
- Implementación de grupos de trabajo.
- Reducción de costos de productos y servicios.
- Implementación de un sistema de adiestramiento administrativo.

Los principios que se aplican para la Administración de la Calidad Total son los siguientes:

- Consecución de la plena satisfacción de las necesidades y expectativas del cliente (interno y externo).
- Desarrollo de un proceso de mejora continua en todas las actividades y procesos llevados a cabo en la empresa (implantar la mejora continua tiene un principio pero no un fin).
- Total compromiso de la Dirección y un liderazgo activo de todo el equipo directivo.
- Participación de todos los miembros de la organización y fomento del trabajo en equipo hacia una Gestión de Calidad Total.
- Integración del proveedor en el sistema de Calidad Total de la organización, dado el fundamental papel de éste en la consecución de la calidad en la misma.

- Identificación y Gestión de los Procesos Clave de la Organización, superando las barreras departamentales y estructurales que esconden dichos procesos.
- Toma de decisiones de gestión basada en datos y hechos objetivos suprimiendo la gestión basada en la intuición. Dominio del manejo de la información.

1.2.2. Just In Time

"Just in time" (también se usa con sus siglas JIT), literalmente quiere decir "Justo a tiempo". Es una filosofía que define la forma en que debería optimizarse un sistema de producción. Se trata de entregar materias primas o componentes a la línea de fabricación de forma que lleguen "justo a tiempo" a medida que son necesarios.

La ventaja competitiva ganada deriva de la capacidad que adquiere la organización para entregar al mercado el producto solicitado, en un tiempo adecuado, con la cantidad requerida, evitando los costos que no producen valor agregado también se obtendrán precios competitivos.

1.2.2.1. Pilares Fundamentales del JIT

El JIT tiene 4 objetivos esenciales (Fig. 1.2) que son:

- Poner en evidencia los problemas fundamentales.
- Eliminar despilfarros.
- Buscar la simplicidad.
- Diseñar sistemas para identificar problemas.

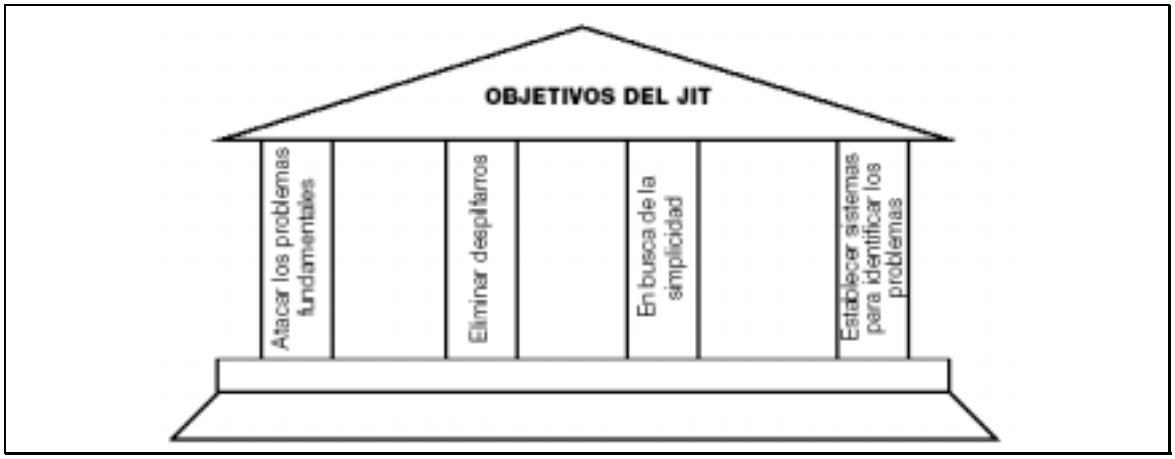


Fig. 1.2 Los 4 pilares del JIT

Estos principios forman una estructura alrededor de la cual es posible formular la aplicación del sistema JIT y serán descritos a continuación.

1. Atacar los problemas fundamentales

Para describir el primer objetivo de la filosofía JIT los japoneses utilizan la analogía del "Río de las Existencias" (Fig. 1.3). El nivel del río representa las existencias y las operaciones de la empresa se visualizan como un barco. Cuando una empresa intenta bajar el nivel del río, en otras palabras, reducir el nivel de las existencias, descubre rocas, es decir, problemas. Hasta hace poco tiempo atrás, cuando estos problemas surgían en algunas empresas, la respuesta era aumentar las existencias para tapar el problema.

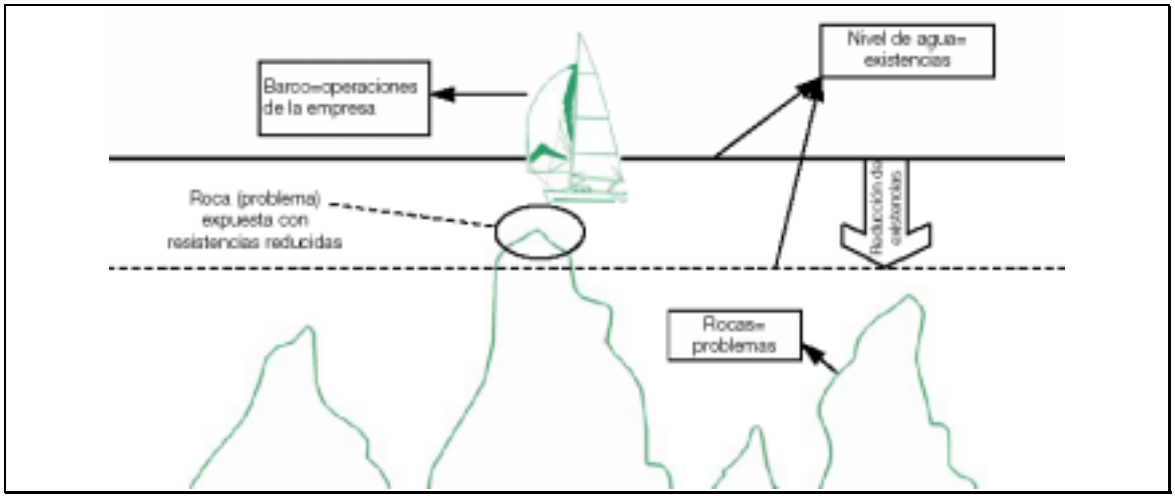


Fig. 1.3 Río de las experiencias

En la Tabla 1.1 se muestran algunos de los problemas con los que se encuentran algunas empresas en sus operaciones cotidianas y las soluciones que el JIT sugiere.

Tabla. 1.1 Problemas típicos de empresas

Problema (Rocas)	Solución tradicional	Solución JIT
Máquina poco fiable	Stock de seguridad grande	Mejorar la fiabilidad
Zonas con cuellos de botella	Programación mejor y más.	Aumentar la capacidad compleja y la polivalencia de los operarios y máquinas
Tamaños de lote grandes	Almacenar	Reducir el tiempo de preparación
Plazos de fabricación largos	Acelerar algunos pedidos en base a prioridades	Reducir esperas, etc, mediante sistema de arrastre
Calidad deficiente	Aumentar los controles	Mejorar los procesos y/o proveedores

El enfoque tradicional y el enfoque JIT se diferencia por el mantenimiento de inventarios grandes despreocupando el estado de la maquinaria, lo cual genere fallas en el producto final (Fig. 1.4).

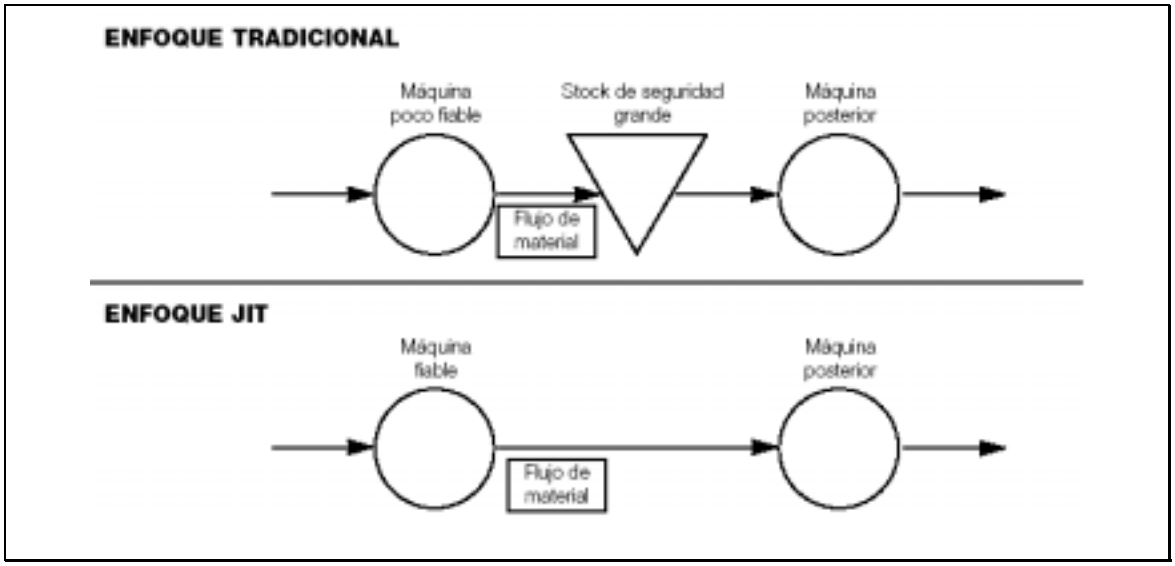


Fig. 1.4 Enfoques respecto a máquinas poco fiables

2. Eliminar despilfarros

Eliminar despilfarros implica eliminar todas las actividades que no añaden valor al producto con lo que se reduce costos, mejora la calidad, reduce los plazos de fabricación y aumenta el nivel de servicio al cliente.

En este caso el enfoque JIT consiste en:

- Hacerlo bien a la primera vez.
- El operador asume la responsabilidad de controlar, es decir, trabaja en autocontrol.
- Garantizar el proceso mediante el control estadístico.
- Analizar y prevenir los riesgos potenciales que hay en un proceso.
- Reducir el mantenimiento de existencias excesivo al máximo.

En el área informática, la obtención de un producto de calidad al primer intento es una tarea complicada, pero en su lugar se puede optar por un proceso de madurez con el que se asegura que en producciones posteriores, el desarrollo de un producto se realizará de manera repetitiva y por lo tanto se tiende a hacerlo bien a la primera vez.

Hay que considerar que el proceso de entrega de calidad a un producto no es responsabilidad única de un grupo de calidad, sino de todos los involucrados en la línea de producción, tomando el desarrollo de software como ejemplo, cada uno de los profesionales envueltos en este proceso deben estar convencidos de que la calidad en su área de responsabilidad influye de manera directa en la calidad del producto final y por tanto debe aportar con su mejor esfuerzo de auto control en su trabajo para evitar carencia de calidad.

La elaboración de un proyecto de software debe necesariamente, como todo proyecto, evaluar los riesgos de cada una de sus etapas para poder desarrollar medios de contingencia o eliminación de estos riesgos.

3. Buscar la simplicidad.

Permite evitar redundancia o confusión en los procesos que se automatizan, y permiten entregar un resultado más fiable y eficiente. El diseñar sistemas para identificar problemas permite evitar inconsistencias en las fases del ciclo de vida que pueden, y van a, generar problemas en las etapas posteriores, para lo cual se tienen que considerar dos puntos importantes:

- Flujo de material.
- Control de estas líneas de flujo.

Un enfoque simple respecto al flujo de material es eliminar las rutas complejas y buscar líneas de flujo más directas, si es posible unidireccionales. Otro es agrupar los productos en familias que se fabrican en una línea de flujo, con lo que se facilita la gestión en células de producción.

4. **Establecer sistemas para identificar los problemas.**

Con el JIT cualquier sistema que identifique los problemas se considera beneficioso y cualquier sistema que los enmascare, perjudicial.

Si realmente se quiere aplicar el JIT se debe que considerar 2 aspectos:

- Establecer mecanismos para identificar los problemas.
- Estar dispuestos a aceptar una reducción de la eficiencia a corto plazo con el fin de obtener una ventaja a largo plazo.

Los cuatro pilares del JIT puede ser comparable con las características deseables de un producto de software de calidad, así, poner en evidencia los problemas fundamentales es esencial para la entrega de soluciones informáticas que se acoplen a las necesidades directas del cliente además de ofrecer un mejor desempeño.

1.2.2.2. Diagrama de Flujo de la Aplicación del JIT

El diagrama del JIT consta de 5 fases (Fig. 1.5) que permiten desarrollar un modelo de calidad en la organización.

1. **Primera fase: cómo poner el sistema en marcha**

Establece la base sobre la cual se construirá la aplicación. La misma exige un cambio en la actitud de la empresa, y esta fase será determinante para conseguirlo. En caso de decidir no aplicar JIT, se puede optar por la aplicación de otro modelo o no usar ninguno.

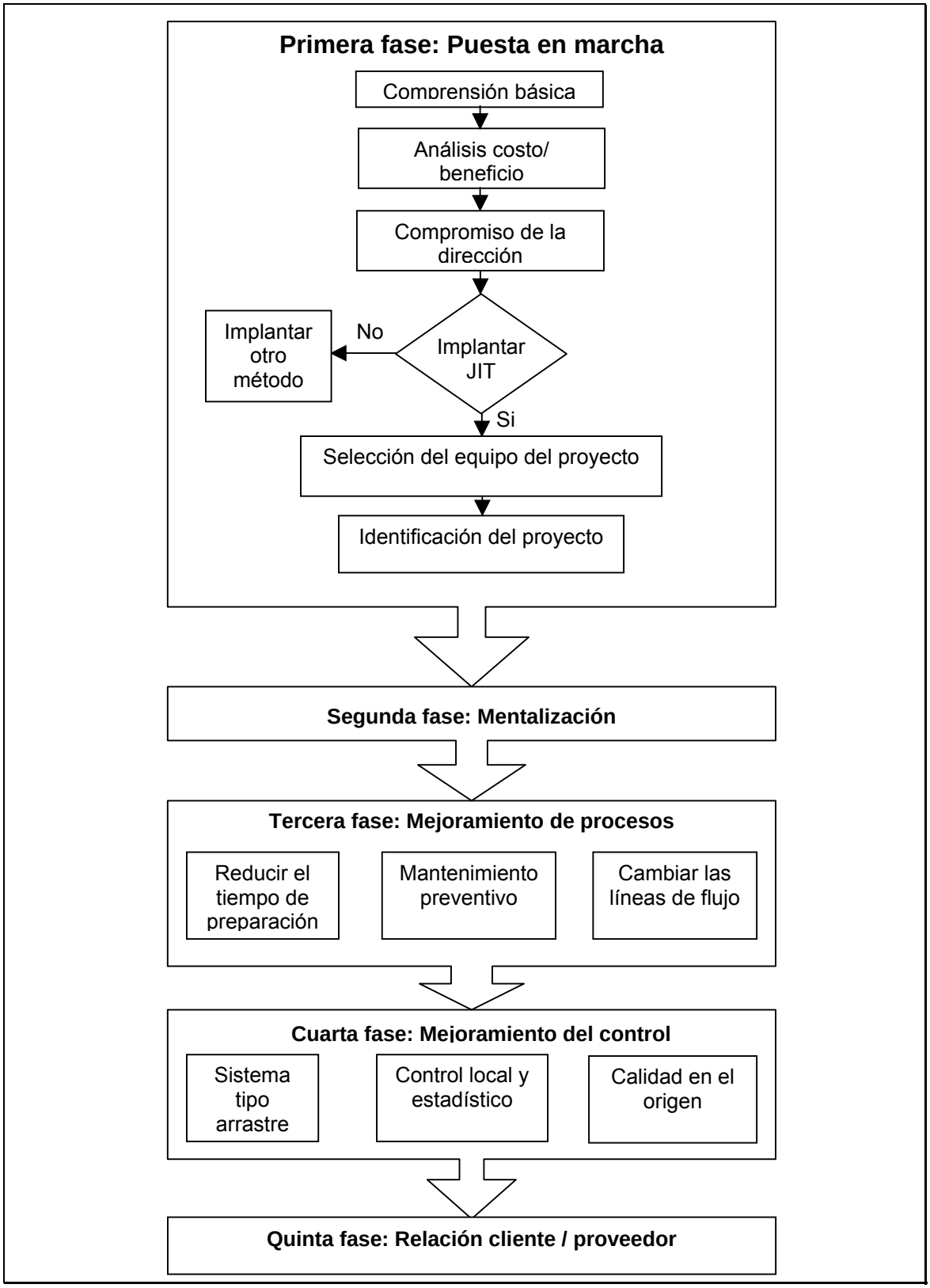


Fig. 1.5 Diagrama de Flujo de la Aplicación del JIT

2. **Segunda fase: Mentalización - Clave del Éxito**

Esta fase implica la educación de todo el personal. Se le llama clave del éxito porque si la empresa escatima recursos en esta fase, la aplicación resultante podría tener muchas dificultades.

3. **Tercera fase: Mejorar los procesos**

La tercera fase se refiere a cambios físicos del proceso de fabricación que mejorarán el flujo de trabajo.

4. **Cuarta fase: Mejoras en el control**

La forma en que se controle el sistema de fabricación determinará los resultados globales de la aplicación del JIT.

5. **Quinta fase: Relación cliente - proveedor**

Es la fase final de la aplicación del JIT. Para poder continuar el proceso de mejora se debe integrar a los proveedores externos y a los clientes externos.

Esta quinta fase se debe empezar en paralelo con parte de la fase 2 y con las fases 3 y 4, ya que requiere tiempo el discutir los requisitos del JIT y los cambios que hay que realizar con los proveedores y los clientes.

1.2.3. Desarrollo de Funciones de Calidad (QFD – Quality Function Deployment)

Es un proceso sistemático de planeación que fue creado para ayudar a un equipo de proyectos a integrar y administrar los elementos necesarios para definir, diseñar y producir un producto (o entregar un servicio) que pudiera satisfacer o exceder las necesidades del cliente. Su principio central es capturar la «voz del cliente » y garantizar que se traduzca en una estrategia apropiada para los productos y los requisitos de los procesos. "[4]

El proceso de QFD se utiliza para diseñar o rediseñar productos/ servicios con dos objetivos fundamentales:

1. Asegurar que el producto/ servicio responderá a las expectativas y necesidades de los clientes (Requerimientos del cliente).
2. Acortar el tiempo que transcurre desde la concepción del producto o proyecto (o de las modificaciones a realizar) hasta su lanzamiento o implantación (Fig. 1.6).

En la Fig. 1.6, se denota un tiempo de definición del producto o proyecto, en el que lleva implícito las actividades de planeación, se puede observar que el tiempo esperado para la culminación del proyecto con el método de QFD es menor comparado con el tiempo esperado con otra técnica, esto se debe a que con QFD la planeación se realiza tomando en cuenta la mayor cantidad de requerimientos por parte del cliente, mientras que con otras técnicas no se realiza un estudio detallado de dichos requerimientos, en la etapa de diseño, los tiempos dependen de la planificación y de la

[4] Carlos Franco. La integración de las necesidades del cliente en los productos y servicios de la organización. Cali Colombia.

recolección de requerimientos del cliente, es por eso que cuando la recolección de requerimientos es leve, el tiempo de diseño tiene una duración mayor que cuando los requerimientos son recolectados con mayor especificad. Y por último, si el diseño se realiza de a cuerdo a los requerimientos del cliente, el rediseño es mínimo.

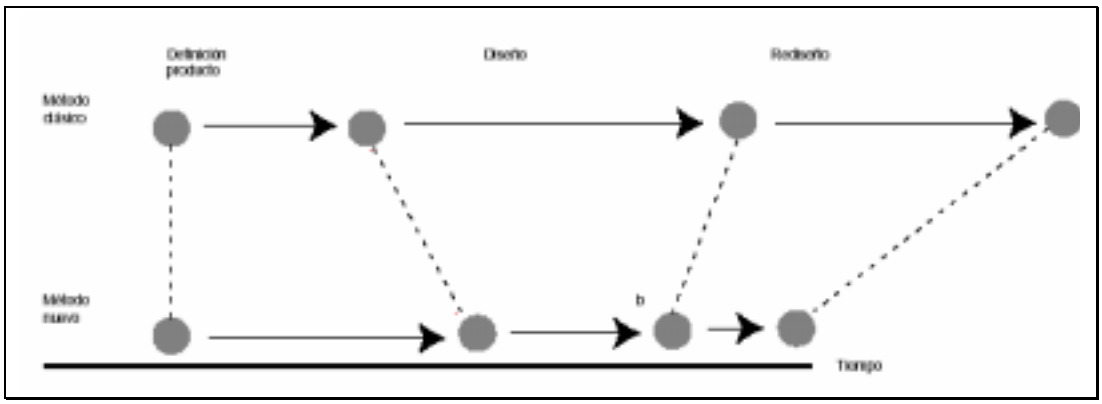


Fig. 1.6 Ciclo de vida de un proyecto; Método clásico vs. Método QFD

Esta disminución de tiempo en el desarrollo de un proyecto puede verse reflejado en un proyecto de software, ya que el software se basa en la especificación de requerimientos del usuario (ERS), y mientras mas claros sean estos requerimientos, mejor se podrá plasmar en el producto final, optimizando el proceso tal como indica el QFD.

1.2.3.1. Ventajas de aplicación de QFD

La aplicación de QFD implica la consideración de ciertos aspectos importantes como son la comunicación, la mejora del producto y la documentación:

- **Comunicación**: Este proceso es esencialmente ínterfuncional y en él deben participar diversas dependencias de la empresa en donde diferentes personas aportan puntos de vista distintos, contribuyendo así a la mejora de las comunicaciones

- **Mejora del producto**: El objetivo más importante que se persigue es el de satisfacer las necesidades del cliente, para ello se requiere realizar un análisis detenido de éstas, priorizando las más esenciales y determinando cuál es viable a satisfacer.
- **Documentación:** Cuando se aplica el proceso del QFD se obtiene una base de datos importante sobre las necesidades del cliente, características del producto, tecnologías, productos y procesos. Estos datos deben ordenarse en forma de matrices y se podrán utilizar en diseños o modificaciones posteriores.

El QFD aporta una serie de beneficios que se deben considerar:

- Posibilita la obtención de bases de datos para diseño y modificaciones futuras.
- Se dispone de un proceso sistemático para diseño y modificación de productos y servicios.
- Se reducen tiempos de lanzamiento de productos / servicios.
- Los costos se pueden reducir, pues se evitan errores y reproceso en etapas críticas.
- Se incrementa la satisfacción del cliente, al dar respuesta rápida y precisa a sus necesidades y requerimientos.

1.2.3.2. Metodología del QFD

La aplicación del QFD se lleva a cabo con la consecución de cuatro etapas:

1. Etapa de organización y planeación. Se fijan objetivos y se elige el proyecto teniendo en cuenta los beneficios que se esperan.

2. Etapa descriptiva. Se definen necesidades del cliente y las características del producto / proceso, así como las diversas maneras de fabricar el producto y las tecnologías disponibles.
3. Etapa de progreso. Aquí se reúnen todos los datos, se realizan pruebas y se construyen prototipos, buscando obtener mejoras significativas.
4. Etapa de implantación. Se elabora un plan de acción detallado para poner en práctica las ideas propuestas y desarrolladas en las etapas anteriores. Cuando se intenta diseñar o rediseñar un producto se debe considerar los siguientes factores:
    - La calidad en cuanto a atributos del producto / servicio vs. necesidades y expectativas del cliente.
    - Los costos de posibles alternativas.
    - Tipo de tecnología requerida.
    - Confiabilidad del producto.
    - Nuevos conceptos que se reflejarán en innovaciones y consecuentes ventajas competitivas.

Los factores anteriores se deberán observar desde diferentes perspectivas:

- Requisitos del cliente.
- La función del producto.
- Características sustitutivas de calidad.
- Partes que integran el producto.

La consideración simultánea de cada factor y cada perspectiva origina una matriz. Es así posible llegar a reunir veinte matrices distintas, con

cada uno de los cinco factores expresados y las cuatro perspectivas consideradas. De estas veinte matrices hay cuatro que se pueden considerar más importantes y que se utilizan para llevar a la práctica el proceso del QFD. Estas matrices se designan como A1, A2, A3, y A4 y se les denomina así:

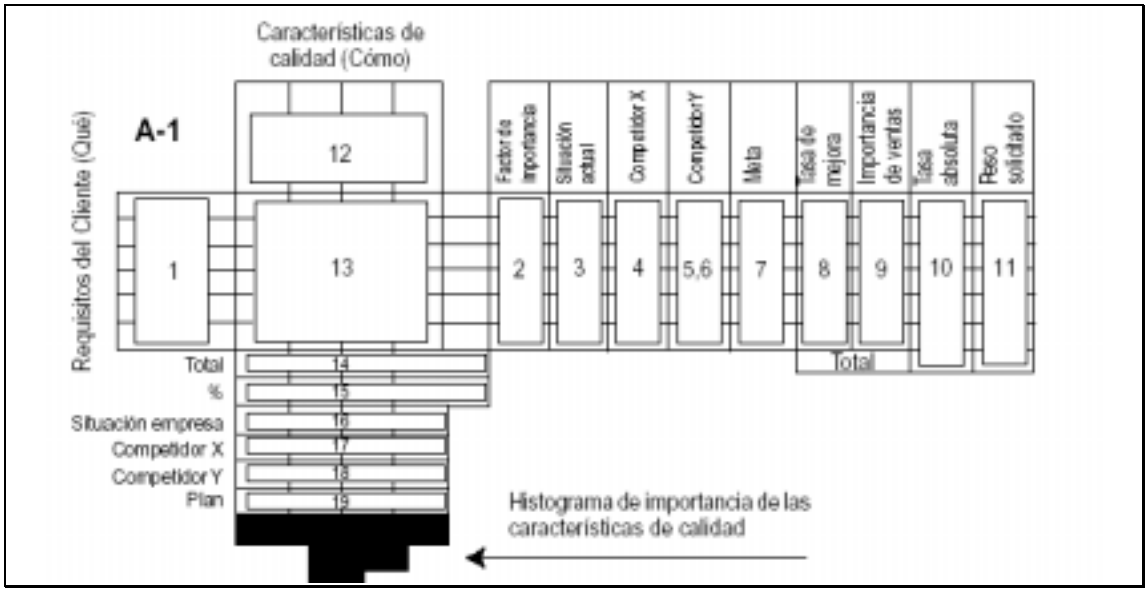


Fig. 1.7 Matriz voz del cliente. Características de calidad vs. Requisitos del cliente

Matriz A1 (Fig. 1.7). Esta es una matriz compleja que consta del esquema que se presenta en la Figura 6. Posee varias entradas así:

1. **Requisitos del cliente**.
2. **Factor de importancia** de cada requisito, en escala 1 a 5.
3. **Situación actual** con respecto a cada requisito, en escala 1 a 5.

4, 5, 6. Posición de la competencia en cada requisito, en escala 1 a 5.

7. **Meta de la empresa** con cada requisito, en escala 1 a 5.
8. **Tasa de mejora** definida como el cociente entre la meta y la situación actual de la empresa.

9. **Importancia de cada requisito para ventas** así: 1.5 muy importante; 1.2 relativamente importante; 1 poco importante.
10. **Tasa absoluta de mejora e importancia ponderada** = Factor de importancia x Tasa de mejora x Importancia ventas.
11. **Tasa relativa.** Convertir el factor anterior a cifras relativas usando porcentajes.
12. **Características de calidad.** Aquí se listan los factores cuantificables y controlables por medio de los cuales la empresa espera satisfacer los requisitos del cliente. Las características de calidad indican la manera en la que llega a la satisfacción del cliente.
13. **Relaciones que existen entre los requisitos del cliente y las características de calidad.** Cada entrada de esta sub. - matriz tiene dos datos:
    - **13.1.** Un dato de relación entre el requisito y la característica. Esto se expresa con un símbolo y con un número. Si la relación es fuerte el símbolo es ⊕ y posteriormente se le asignará un valor de 9. Si la relación es media el símbolo es O y se le dará un valor de 3. Si la relación es débil el símbolo será △ y se le asignará un 1.
    - **13.2.** Cada intersección requisito- característica tendrá un valor que obtendrá de multiplicar el valor de 11 por el valor numérico asignado a su símbolo.
14. **En esta fila se suman los totales** numéricos de las diversas columnas de la matriz requisitos características.
15. **Valor porcentual** de la suma de los valores de la fila
16. **Características actuales** que cumple la empresa.

17, 18. En estas dos filas se colocan los valores de las características de calidad de cada uno de los competidores.

19. Características de calidad que se desean lograr a la luz del contraste con la competencia.

Existen otras matrices (A2, A3, y A4) denominadas respectivamente *La voz del Ingeniero* Fig. 1.7*, la comprobación de relaciones* Fig. 1.8 *y las relaciones entre partes y características de calidad* Fig. 1.9.

En este gráfico (Fig. 1.8) se expresa las funciones o actividades que se realizan dentro de un proceso y se lo compara con las características que evalúan la calidad de dicha actividad utilizando los símbolos descritos anteriormente (⊕, △ y ○).

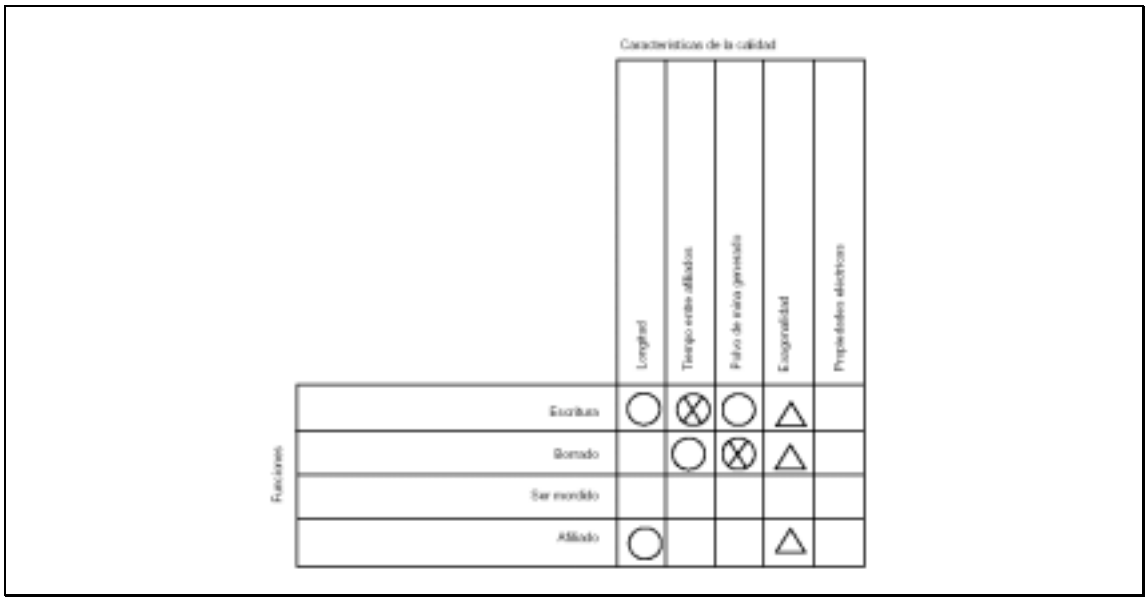


Fig. 1.8 Matriz de la Voz del Ingeniero

Relaciones entre las características de calidad (Fig. 1.8), para dar una idea de la interrelación entre las características y la importancia que tienen (¿Qué pasa si no se cumple una? ¿Qué consecuencias tendría?).

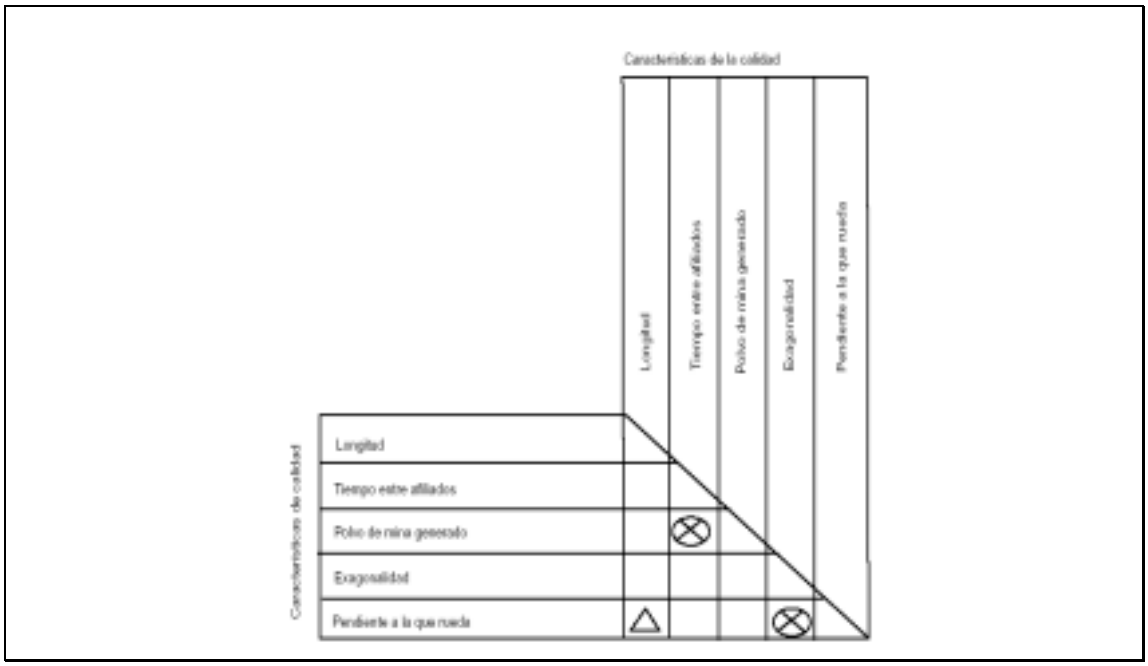


Fig. 1.9 Matriz de la comprobación de Relaciones

Indica los recursos que se utilizan y los factores de calidad, esto ayuda a verificar la incidencia del estado, funcionamiento de aparatos, características técnicas, etc. para alcanzar la calidad (Fig. 1.10).

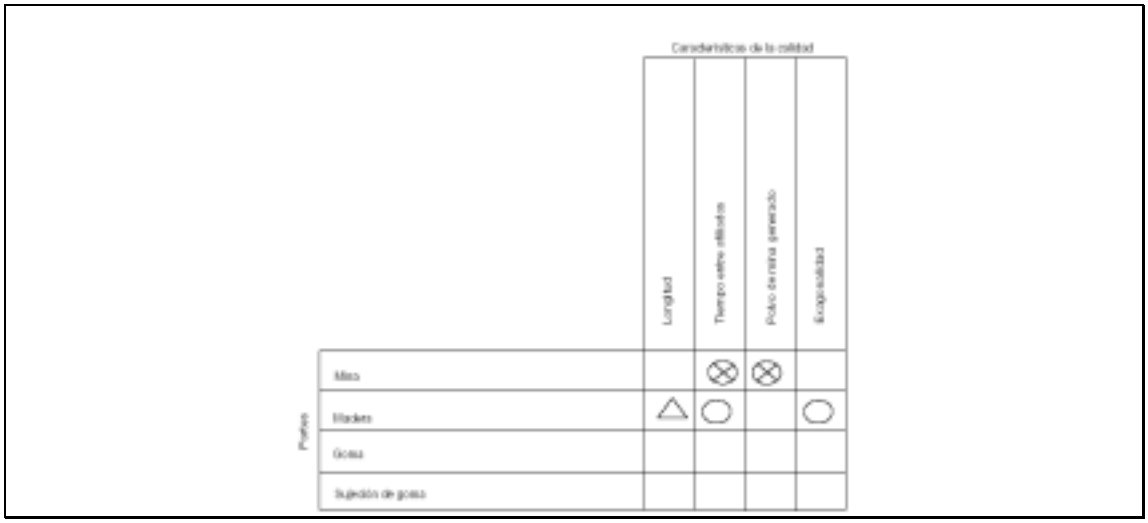


Fig. 1.10 Matriz de Características de la Calidad y las Relaciones entre partes.

1.2.3.3. La matriz Casa de la Calidad y el enfoque simplificado

La denominada matriz Casa de la Calidad presenta un esquema simplificado entre las matrices A1 y A3 y es uno de los esquemas más utilizados en el QFD. Un esquema simplificado de la Casa de la Calidad se presenta en la Fig. 1.11.

Existe otro enfoque para aplicación del QFD, particularmente utilizado por la empresa automotriz Ford y que considera el desarrollo del proceso de QFD en cuatro fases (Fig. 1.11) así:

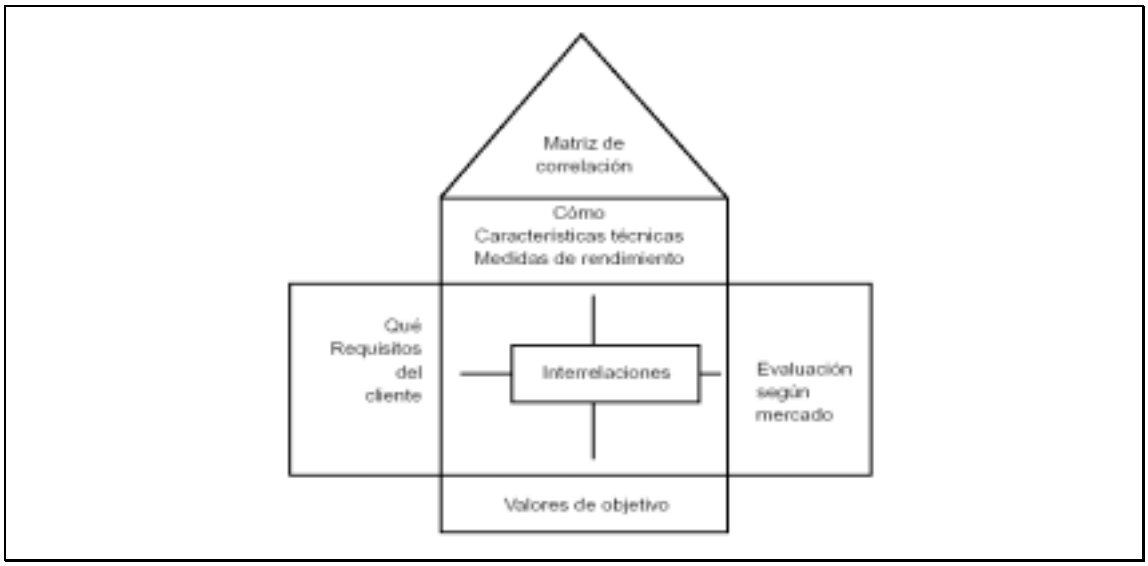


Fig. 1.11 Casa de la Calidad

- **Fase I:** Planeación del producto. Considera la matriz de requisitos del cliente vs. las características de calidad.
- **Fase II:** Planeación del diseño. Considera las características de calidad vs. las características de los componentes.
- **Fase III:** Planeación del proceso. Considera las características de los componentes vs. las características del proceso.
- **Fase IV:** Planeación de la producción. Considera las características del proceso vs. los requisitos de producción.

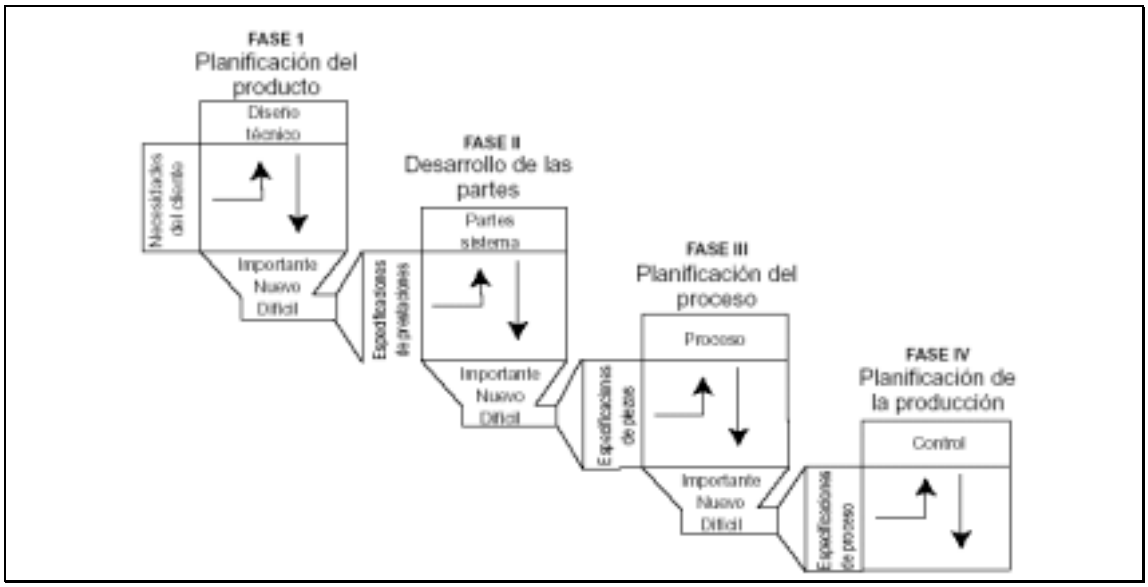


Fig. 1.12 El modelo de cuatro fases

1.3 Aseguramiento de la Calidad Total

La meta de la industria competitiva, respecto a la calidad del producto, se puede exponer claramente: suministrar un producto o servicio en el cual su calidad haya sido diseñada, producida y sostenida a un costo económico y que satisfaga por entero al consumidor.

Por medio del aseguramiento de la calidad, las gerencias de las organizaciones han sido capaces de aprovechar la fuerza y confianza de la calidad de los productos y servicios, lo que les permite adelantarse en el volumen de mercado y ampliar la mezcla de productos con un alto grado de aceptación del cliente y de estabilidad en utilidades y crecimiento.

El aseguramiento de la calidad constituye las bases fundamentales de la motivación positiva por la calidad en todos los empleados y representantes de la organización, desde altos ejecutivos hasta trabajadores de ensamble, personal de oficina, agentes y personal de servicio. El empleo del aseguramiento de la calidad total es uno de los factores principales para lograr un mejoramiento de la productividad total.

Las relaciones humanas eficientes son básicas en el aseguramiento de la calidad. Un resultado importante de esta actividad es su efecto positivo en el operario al crear responsabilidad e interés en producir calidad. El último análisis es como un par de manos humanas que efectúan operaciones importantes que se reflejan en la calidad del producto.

Los términos “control de calidad” y “aseguramiento de calidad” han llegado a tener diferentes significados en diversas empresas cada término significa diferentes aspectos de la actividad de la satisfacción del cliente con la calidad.

La labor de asegurar la calidad en un producto o servicio se logra por medio de un adiestramiento de la alta dirección en administración de calidad de todos sus recursos y procesos. Es importante la unificación de las áreas que conforman la empresa, lo cual es resultado de la entrega de funciones específicas a cada una de dichas áreas.

En primer lugar, es importante comprender lo que no es el aseguramiento de la calidad:

- **No es el control o la inspección de la calidad:** Aunque un programa de aseguramiento de la calidad incluirá el control y la inspección de la calidad, ambas actividades sólo forman parte de un compromiso total de la empresa hacia la calidad.
- **Se relaciona directamente con el control del artículo producido:**.Estas dos actividades no participan en actividades previas tales como diseño, compras, ventas y comercialización. Por lo tanto, sólo deben considerarse como uno de los elementos de la presentación total.

- **No es una actividad de verificación minuciosa:** El departamento de aseguramiento de la calidad, no debe tener la responsabilidad de comprobar todo lo que hagan los demás. La responsabilidad de estas verificaciones debe de corresponder a quienes tengan los suficientes conocimientos y experiencia para determinar la eficacia de la actividad que se está revisando.
- **No tiene la responsabilidad de las decisiones de ingeniería:** El departamento de aseguramiento de la calidad no debe tomar decisiones relacionadas con actividades de ingeniería o alguna otra área.
- **No es un enorme productor de papeleo:** Algunos papeles tales como: certificados de producción, de pruebas y documentos certificados de terceros, se han considerado comúnmente necesarios para cumplir los requisitos de aseguramiento de la calidad, por lo que existe la interpretación errónea de que todos esos documentos son requisitos indispensables en un programa de aseguramiento de la calidad. Sin embargo, un programa de aseguramiento de la calidad bien diseñado y completamente puesto en práctica, asegurará y comprobará que los requisitos de documentación y certificación se lleven a cabo en la forma más eficiente. La responsabilidad de la documentación y certificación no debe recaer en el departamento de aseguramiento de la calidad.
- **No es un área de costos excesivos:** En cuanto a lo que se refiere a documentación y certificación, el departamento de aseguramiento de la calidad no es un área de costos excesivos. Existen requisitos de

procedimientos para respaldar un programa de aseguramiento de la calidad, pero una vez dicho esto, en cualquier caso toda la organización que se respete debe de tener en vigor controles de procedimientos y no debe diferir su instalación y puesta en práctica hasta que la empresa se haya vuelto tan grande y/o se haya perdido el control. En estos casos el desarrollo y puesta en marcha de estos programas puede ser muy importante, pero el costo se debe comparar con las mejorías en eficiencia, productividad y rentabilidad. El mejor momento de poner en marcha un programa de aseguramiento de la calidad es al inicio de una empresa. La prevención es mejor que la cura.

- **No es una solución para todos los males:** El aseguramiento de la calidad no curará todo, pero sí será de gran ayuda para lograr que las cosas salgan siempre bien en la primera ocasión.

El aseguramiento total de la calidad es resultado de la combinación de esfuerzos del grupo de desarrollo, mantenimiento y aseguramiento de la calidad para lograr que los procesos involucrados en la producción (ingeniería, mercadotecnia, fabricación y servicios) satisfagan al consumidor de la manera más económica. Para lograr este objetivo es necesario realizar las siguientes tres actividades:

- Planeación de la calidad.
- Control de la calidad.
- Mejoramiento de la calidad.

Cada uno de estos procesos es universal y se lleva a cabo con una secuencia uniforme de actividades.

El punto de arranque es la planeación de la calidad, es decir, crear un proceso que será capaz de cumplir con metas establecidas, y hacerlo bajo las condiciones de operación de cada empresa.

El siguiente proceso que se debe efectuar es el de control de la calidad es con el que se pretende observar que todos los procesos se realicen de acuerdo a los lineamientos que se han determinado para entregar calidad, este control es esencial para lograr un mejoramiento en los procesos por medio de la retroalimentación proporcionada por los encargados de cada actividad.

El punto normal y común de arranque es establecer un consejo directivo de calidad que formule y coordine esta función en la organización. Generalmente los miembros de tal consejo son los gerentes de muy alto nivel.

El consejo prepara una lista de responsabilidades entre las que se podrían encontrar:

- Establecer las políticas corporativas de calidad.
- Establecer las metas corporativas de calidad; revisar las metas de calidad de las divisiones y de las funciones principales.
- Establecer los planes de calidad corporativos, revisando aquellos planes que son divisionales o funcionales.
- Proveer la infraestructura y los recursos necesarios para llevar a cabo estos planes.
- Revisar los resultados de la calidad en comparación con el plan y las metas.

- Revisar el sistema de evaluación de desempeño de los gerentes, incorporando el desempeño relativo a la calidad en comparación con las metas establecidas.

Tradicionalmente la fijación de metas y objetivos se han basado en los resultados obtenidos por cada empresa con anterioridad. Sin considerar que algunos procesos produjeron muchos desperdicios y nunca fueron examinados, debido a que los gerentes habían cumplido con sus presupuestos. Estos procesos fueron planificados con desperdicio por lo que se puede suponer que no existe calidad en sus resultados.

Algunos de los factores importantes en el proceso de aseguramiento de la calidad son los siguientes:

- **Capacitación:** Se debe entender a la capacitación como una inversión y no como un gasto, ya que la capacitación en áreas críticas como Recursos Humanos o Atención al Cliente ya que se verán reflejados irremediablemente al momento de entregar al cliente un producto o servicio.
- **Medición de la calidad:** Esta medición de la calidad se refiere a determinar un escala de calidad de productos o servicios similares en el mercado para estimar cual es el mínimo esperado por el cliente ante los productos o servicios que se encuentran ya a su alcance en el mercado
- **Mejoría de la calidad:** Se refiere a innovar en cualidades que aportan con calidad a un producto, proceso o procedimiento que ya cuenta con un grado de calidad.

Una vez que se ha elaborado y puesto en práctica un programa de aseguramiento de la calidad, la única forma en que la organización puede verificar su efectividad es realizando auditorias periódicas.

Por lo general al programa de calidad de una organización no se le da mucha importancia, sin embargo, debido al deficiente control de todas sus actividades, la organización quizá esté perdiendo una gran cantidad de dinero. La ineficiencia debida a la duplicidad de actividades, las altas tasas de reparación y desperdicios, los procedimientos inadecuados, etc., pueden causar que los costos de calidad sean más altos que el margen de utilidad global. Si se realiza con efectividad, la auditoria de calidad debe descubrir esos problemas, siempre y cuando se ejecute contra requisitos documentados.

Todas las normas de aseguramiento de la calidad contienen un requisito para la auditoria o la revisión del programa o el plan de calidad y, en general, esa actividad puede definirse como una actividad planeada y documentada realizada de acuerdo con procedimientos y listas de verificación por escrito, para comprobar mediante la investigación, el examen y evaluación de la evidencia objetiva, que se han elaborado, documentado y puesto en práctica con efectividad los elementos aplicables de un programa o plan de calidad, de acuerdo con requisitos especificados.

Al principio e inmediatamente después de la puesta en práctica de un programa o plan de calidad, debe realizarse una auditoria para confirmar que se cuente con todos los procedimientos e instrucciones relevantes en los lugares de la actividad y que el personal esté consciente de sus responsabilidades dentro del programa o plan.

El alcance de una auditoria está relacionado con la cantidad del programa o plan de aseguramiento de la calidad que debe revisarse para confirmar que las actividades estén de acuerdo con los requisitos.

Igual que todas las demás actividades, las auditorias requieren de preparación y planeación, para lo cual existen varios pasos que considerar:

- Preparar el programa de auditoria.
- Designar a una o varias personas como responsables de la auditoria.
- Notificar al departamento o persona que se auditará (auditado).
- Identificar, obtener y revisar toda la documentación relacionada con la auditoria.
- Preparar la lista de verificación de la auditoria.
- Convenir el programa de tiempos de la auditoria.

Una vez en el lugar de la auditoria, el auditor debe concertar una breve reunión entre él y los auditados. El propósito de esta junta es:

- Que el auditor presente a los representantes de los auditados, si es que no se conocen.
- Confirmar, en forma breve, el propósito y alcance de la auditoria.
- Revisar el alcance de la auditoria, el programa de tiempos y la agenda.
- Convenir una fecha tentativa para una ajunta de cierre.
- Nombrar a los que acompañarán a los auditores.

El objetivo de la auditoria es examinar lo adecuado de los sistemas para asegurar la calidad. Se espera que el auditor esté calificado y tenga experiencia, una auditoria de calidad no es tan sencilla como puede parecer. El auditor necesita tener la experiencia apropiada para llevar a cabo la

auditoria. Además del entrenamiento técnico o científico apropiado necesita conocer de:

- Normas de aseguramiento de la calidad.
- Las prácticas de calidad de industria de que se trate.
- Los requisitos contractuales y legales.
- Planeación.
- Prácticas de auditoria.
- Costos y Beneficios.

Desarrollar y poner en práctica un programa de aseguramiento de la calidad requerirá tiempo y costará dinero. Es difícil dar un estimado de costos, ya que éstos dependerán del tamaño de la organización y de la complejidad de sus operaciones, de si existe o no alguna forma de programa y cuántos procedimientos deben redactarse y ponerse en marcha.

Al principio se producirán desembolsos relacionados con el tiempo que necesite el grupo de trabajo para establecer los requisitos; el tiempo necesario para redactar los procedimientos y los costos que ocasionen las sesiones de conciencia. Hasta cierto punto, estos costos pueden considerarse como gastos de capital, amortizándolos contra los ahorros importantes obtenidos en áreas tales como: revisiones a los documentos de ingeniería, reproceso de artículos defectuosos, reducción de los desechos y otros.

1.3.1. Impacto en la organización del control total de la calidad.

El control total de la calidad incluye a profundidad no solo las actividades de control de calidad, sino con mayor importancia las actividades de calidad multi-funcionales interdependientes a través de toda la organización;

”*El impacto a través de la organización del control total de la calidad implica la implementación administrativa y técnica de las actividades de calidad orientadas hacia el cliente como responsabilidad primordial de la gerencia general y de las operaciones de línea principal de mercadotecnia, ingeniería, producción, relaciones industriales, finanzas y servicios así como la función de control de calidad en si.”.*

La importancia de este impacto en toda la organización es que, para algunas organizaciones, mucha de la demanda de mejoras en la calidad queda fuera del trabajo tradicional del control de calidad. Los programas realmente efectivos de control total de la calidad son aquellos que se apegan a ciertos procedimientos y funciones establecidos dentro de la organización ya que esta forma se puede enfrentar al mercado competitivo actual.

Una contribución esencial de los programas de calidad total es el establecimiento de disciplinas de calidad orientadas hacia el cliente tanto en las funciones de mercadotecnia a ingeniería como en producción, de esta forma, cada empleado de una organización desde la gerencia general hasta el trabajador directo de línea estarán personalmente involucrados con el control de calidad. Esto es vital para el establecimiento de las actitudes básicas requeridas para un enfoque positivo hacia el logro de la calidad en el negocio.

La utilización del control de calidad total en una organización toma el nombre de Administración Total de la Calidad, la misma que abarca todas las etapas del ciclo de vida de un producto (es decir desde su concepción hasta la entrega al cliente final y los servicios posteriores).

1.3.2. Bases para el control total de la calidad

Requiere de formas efectivas de integrar los esfuerzos de un gran número de personas con un gran número de maquinas y enormes cantidades de información. Por tanto, implica preguntas sistemáticas de proporciones importantes, y el enfoque de sistemas es inherente al Control Total de la Calidad.

El término “Sistema” puede ser interpretado de distintas maneras, e históricamente ha pasado de ser un conjunto de operaciones a nivel de oficina para llegar a conformarse un programa de computadora. En el control de calidad, este término ha sido interpretado igualmente de distintas maneras; procedimientos de planta que generan problemas hasta un estante de “manuales” operativos y manuales que cubren todas las rutinas de prueba e inspección.

La experiencia ha mostrado que estos enfoques han sido muy estrechos. El control efectivo de la calidad requiere de una fuerte coordinación de todo el papeleó relevante con las actividades de software, hardware y manuales. Requiere de la integración de las acciones de calidad de personas, las máquinas y la información en sistemas fuertes de calidad total.

Un sistema de calidad es la estructura operativa de trabajo aceptada en la organización y en la planta, documentada con procedimientos integrados técnicos y administrativos efectivos para guiar las acciones coordinadas de las personas, maquinas e información de la organización y la planta de las mejores y mas practicas maneras para asegurar la satisfacción del cliente y bajos costos de calidad.

Un sistema de calidad total claramente definido y por completo instalado, es una base poderosa para el control total de la calidad, en toda la organización, y para la administración total de la calidad. La calidad puede ser una consecuencia en lugar del resultado de objetivos y actividades cuidadosamente planeados; puede ser el producto final de acciones individuales, algunas sin relación, a través del proceso mercadotecnia-ingeniería-producción-servicio-calidad. Puede estar basado en intenciones sinceras, pero sin la guía de objetivos firmes y cuantitativos de la calidad para el cliente implementados por claros programas en la organización.

Como contraste, los sistemas fuertes de calidad proporcionan una base administrativa y de ingeniería para el control efectivo orientado a la prevención, que trata firmemente con los niveles actuales de complejidad humana, de maquinaria y de información que caracterizan las operaciones de la organización y la planta de hoy.

Las nuevas tecnologías de ingeniería de sistemas y administración de sistemas son bases importantes para el establecimiento y la operación continua en la administración de los sistemas de calidad. Para que esto sea así se tienen impactos técnicos y administrativos fundamentales en el trabajo de control de calidad.

CAPITULO II: GESTIÓN DE CALIDAD DEL SOFTWARE

2.1. La calidad del software

La calidad del software es un término que ha sido descrito y discutido por muchos autores a lo largo de la historia de la calidad del software. A continuación se revisarán algunas de estas definiciones:

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se esperan de todo software desarrollado profesionalmente"[5]

"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario."[6]

Según Deming[7], calidad es "Conformidad con los requisitos y confianza en el funcionamiento"

De acuerdo con estas definiciones, se puede observar que primordialmente, la calidad de un producto de software se evalúa en torno a los requerimientos del usuario, expresados en el ERS, que éste cumple (requerimientos explícitos), así como el grado con el que dicho producto satisface las necesidades que el cliente no es capaz de expresar (requerimientos implícitos). Si no existe un acuerdo sobre los requisitos implícitos y explícitos

[5] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002. Capitulo VIII, Pág. 135.

[6] IEEE, Std. 610-1990: Compilación de glosarios computacionales del estándar IEEE, IEEE, New York, Enero 18 1991.

[7] Dr. Edwards Deming, estadista estadounidense, planteó en los años 40 teorías para alcanzar la calidad para la armada de su país. Estas teorías no fueron muy bien recibidas en su patria, pero triunfaron finalmente en Japón logrando un gran desarrollo industrial de este país.

que serán abarcados por el software, la calidad del software desarrollado quedará en entredicho.

Para evaluar la calidad del software, se debe diferenciar entre la calidad del PRODUCTO software y la calidad del PROCESO de desarrollo.

- La calidad del producto software se diferencia de la calidad de otros productos de fabricación industrial, ya que el software tiene ciertas características especiales.
- El software es un producto mental No restringido por las leyes de la Física o por los límites de los procesos de fabricación. Es algo abstracto, y su calidad también lo es.
- Se desarrolla, no se fabrica. El costo está fundamentalmente en el proceso de diseño, no en la producción. Y los errores se introducen también en el diseño, no en la producción.
- El software no se deteriora con el tiempo, no es susceptible a los efectos del entorno, y su curva de fallos es muy diferente de la del hardware.
- Es artesanal en gran medida ya que en su mayoría, el software, se construye *"a la medida"* en lugar de ser construido ensamblando componentes de calidad existentes lo que causa, en muchos casos, carencia de Calidad en el Software.
- El mantenimiento del software es mucho más complejo que el mantenimiento del hardware. Cuando un componente hardware se deteriora se sustituye por una pieza de repuesto, pero cada fallo en el software implica un error en el diseño o en el proceso mediante el cual se tradujo el diseño.

2.2. Gestión de la Calidad del Software

La gestión de la calidad es una actividad esencial en cualquier empresa de producción para asegurar la calidad de sus productos, y la competitividad frente a la oferta del mercado.

Las tareas principales que deben desempeñarse para controlar la Calidad del Software y de su proceso de desarrollo son:

- Asegurar la utilización de métodos y herramientas de desarrollo.
- Realizar revisiones técnicas formales.
- Estrategia de pruebas.
- Auditorias de configuración.
- Control de cambios
- Documentación e informes de cada paso.
- Mediciones.

La gestión de la calidad del software actúa sobre 4 pilares que componen el proceso de desarrollo de software:

1. **Procesos del ciclo de vida:** El ciclo de vida del software es el período de tiempo que comienza con la concepción de la idea de un software y que termina con la vida útil del mismo.
2. **Técnicas:** (¿Cómo?) Es el conjunto de operaciones que determinan la manera en la que se realizará el software a lo largo de su ciclo de vida.
3. **Organización:** (¿Quién?) Es el grupo de trabajo con el que se cuenta para el desarrollo de un proyecto de software y la especificación de las actividades que cada integrante realizará.
4. **Infraestructura:** (¿Con qué?) Especificación de los recursos que se utilizarán para el desarrollo de un proyecto de software.

2.2.1. Técnicas para la gestión de calidad de software

Las técnicas utilizadas para realizar la Gestión de la Calidad pueden ser aplicadas completamente en la gestión de calidad de software. Es así que las técnicas del JIT (Just In Time), TQM (Total Quallity Management) y QFD (Quality Function Deployment) [8] son las aplicables al proceso del software.

2.2.1.1. Just In Time (JIT)

El propósito del JIT es lograr que se entregue un producto acorde con los requerimientos o necesidades del cliente o usuario. Los cuatro objetivos fundamentales de JIT se traducen al software de la siguiente manera.

1. **Poner en evidencia los problemas fundamentales:** Dentro del ámbito del software los problemas frecuentes no son a nivel de errores o defectos del producto como tal, si no también, que estos problemas pueden detectarse también dentro de las fases de la ingeniería del software.
2. **Eliminar despilfarros**: En el proceso de la ingeniería del software, es fundamental implementar un cronograma de actividades que permita llevar a cabo a cabalidad los procesos, con la finalidad de entregar un producto de calidad.
3. **En busca de la simplicidad:** Una de las habilidades que el grupo de desarrollo debe buscar, es la sencillez en la que los procesos que se desean automatizar puedan plasmarse en una solución informática
4. **Establecer sistemas para identificar los problemas:** Con la finalidad de llegar a la satisfacción del cliente, se debe enfatizar en las

[8] Ver Capitulo I Sección 1.2. Técnicas de Cesión de la Calidad.

revisiones de los procesos de la ingeniería del software y su concordancia con la ERS.

2.2.1.2. Quality Function Deployment (QFD)

Las ventajas principales del QFD se centran en la realización de una planeación detallada del proyecto, asegurando de esta manera que las siguientes etapas del proyecto puedan ser realizadas con mayor eficacia y en un menor tiempo (Fig. 2.1).

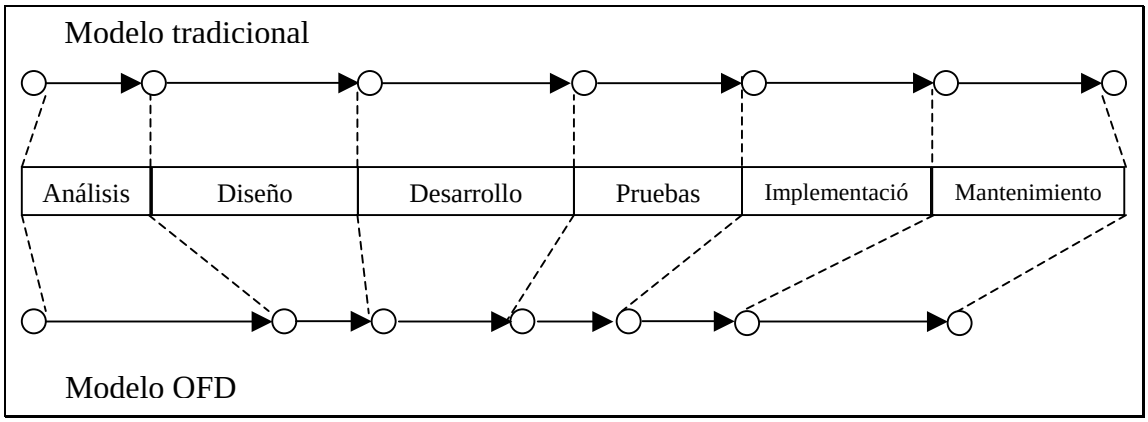


Fig. 2.1 Ciclo de vida de un proyecto de software; Modelo tradicional vs. Modelo QFD

De acuerdo a la metodología del QFD *(Capitulo I, Sección 1.2.3.2, Pág. 21)*, la aplicación de esta metodología se logra mediante el cumplimiento de cuatro etapas:

1. **Etapa de organización y plantación**: consiste en la realización del estudio de los objetivos y alcances del proyecto así como de los aspectos relacionados con la situación actual de la organización.
2. **Etapa descriptiva:** Es la descripción de los requerimientos en función de las necesidades del cliente, reflejados en la Especificación de Requerimientos de Software (ERS)

3. **Etapa de progreso:** Abarca las fases de diseño, desarrollo y pruebas de la Ingeniería de software.

   QFD propone el mejoramiento continuo a través de retroalimentación, la misma que es conceptualizada por el proceso de Configuración de Software *(Capitulo II, Sección 2.7, Pág. 96)*.

4. **Etapa de implantación**: Consiste en realizar un plan de implantación el cual consta en el que se detalla los recurso necesarios (personal, tecnología y tiempo) para poner en funcionamiento el sistema.

2.2.1.3. Administración de la Calidad Total (TQM)

Dentro del área informática, TQM puede ser aplicado dentro del ciclo de vida del software (Fig.2.2) considerando que la satisfacción tanto del cliente interno (participantes en cada fase) como del cliente externo (usuario final del producto de software) conducirá a que el producto final presente características de calidad.

El desarrollo de software se puede comparar con una empresa cualquiera ya que para la culminación de un producto de software se debe atravesar por un proceso en el que se entrega y recibe materia prima, expresada en información y con esta se inicia una nueva etapa.

Los proveedores en la producción de software son las personas que están en contacto de los procesos que se requiere automatizar, mientras que los usuarios son las personas que interaccionan con el producto final.

Partiendo de esta comparación se puede asegurar entonces que la técnica del TQM es aplicable al software.

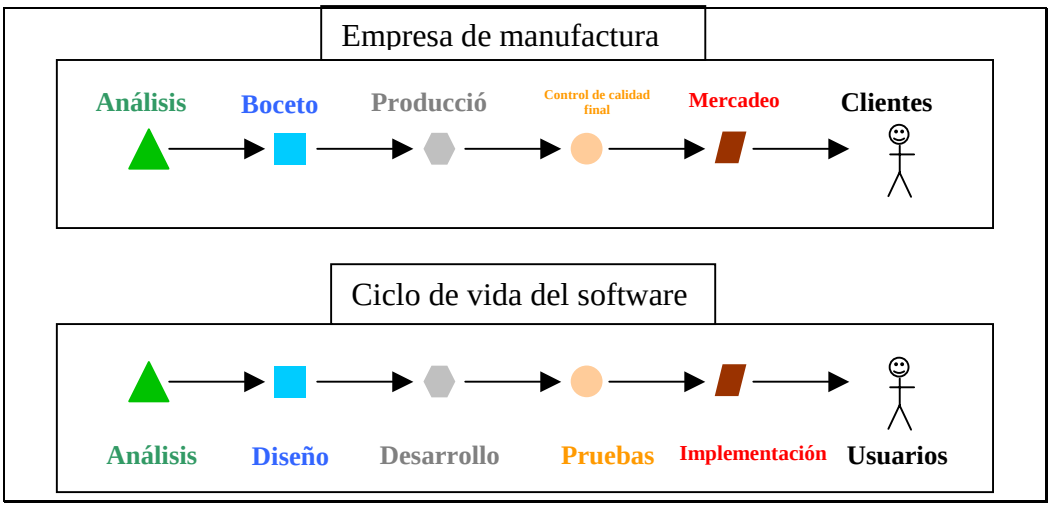


Fig. 2.2 Comparación del ciclo de vida del software con una empresa de manufactura

2.3. Gestión de proyectos de Software

Realizar un proyecto de software consiste en desarrollar aplicaciones que contribuyan a un mejor desarrollo de las funciones en una organización, lo cual implica un proceso de gestión para la creación de un producto de software, que encierra un conjunto de actividades, una de las cuales es la estimación.

Aunque la estimación, es mas un arte, que una ciencia, es una actividad importante que no debe llevarse a cabo de forma descuidada. Existen técnicas útiles para la estimación de costos, tiempo y recursos, así como:

- Estimar en base a resultados de proyecto similares realizados con anterioridad.
- Utilización de técnicas de descomposición para la estimación de costos y esfuerzo de proyecto.
- Utilización de modelos empíricos para la utilización.

Al estimar se toma en cuenta no solo del procedimiento técnico a utilizar en el proyecto, sino también en los recursos, costos y planificación. El

tamaño del proyecto es otro factor importante que puede afectar la precisión de las estimaciones. A medida que el tamaño aumenta, crece rápidamente la interdependencia entre varios elementos del Software.

"La gestión de proyectos implica la planificación supervisión y control del personal, el proceso y de los eventos que ocurren mientras evoluciona el software desde la fase preliminar hasta la implementación operacional."[9]

Es importante, para gestionar un proyecto considerar los aspectos del personal, producto, proceso y proyecto (denominado como las cuatro P).

2.3.1. Personal

El personal debe estar organizado para poder cumplir efectivamente el desarrollo del software, por tanto se debe prestar gran atención a las actividades de reclutamiento, selección, entrenamiento, diseño de la organización y el desarrollo del espíritu de equipo.

Se puede clasificar al personal que participa en el proyecto de software en cinco categorías:

1. **Gestores superiores:** Define los aspectos del negocio que a menudo tienen una significativa influencia en el proyecto.
2. **Gestores técnicos del proyecto:** son los encargados de planificar, motivar, organizar y controlar a los profesionales que desarrollan el producto de software.
3. **Profesionales:** proporcionan las capacidades técnicas necesarias para la ingeniería de un producto o aplicación.
4. **Clientes:** especifican los requisitos para la ingeniería de software.

[9] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo III, Pág. 37

5. **Usuarios finales:** son los encargados de interactuar con el software una vez que este sea puesto en producción.

2.3.2. Producto

La planificación del proyecto de software seria incompleta sino existe previamente una determinación de objetivos y una limitación del ámbito del producto, esto debe ser definido conjuntamente por el desarrollador de software y el cliente.

Para limitar el ámbito del proyecto se debe considerar ciertas características como son:

- **Contexto:** Representa la forma en la que el software a construir encaja dentro de la organización o como se relaciona con sistemas o módulos previamente instalados.
- **Objetivos de información:** Define las entradas requeridas por el sistema.
- **Función y rendimiento:** representa la forma en el que el sistema opera las entradas para transfórmalas en una salida.

2.3.3. Proceso

Dentro del proceso de software existen fases genéricas que son aplicables a todo tipo de software, sin embargo es importante escoger el ciclo de vida adecuado de acuerdo a la clase del proyecto.

Es responsabilidad del grupo de trabajo del proyecto el determinar las actividades estructurales del proyecto y las funciones que se deben realizar en el proyecto. Para esto se utiliza la matriz de la Tabla. 2.1, en la que se colocan en la primera fila las actividades estructurales, mientras

que en la primera columna se colocan las tareas concernientes al la ingeniería de software.

El trabajo del administrador del proyecto consiste en estimar los recursos necesarios para cada celda de la matriz, determinar fechas de inicio y finalización para las tareas asociadas en las mismas y los productos a esperados como consecuencia de cada celda.

Tabla 2.1 Matriz de Administración de Proyectos

Actividades del proyecto / Actividades estructurales	Comunicación con el cliente	Planificación	Análisis de Riesgos	Fases de la Ingeniería del software.
Tareas de Ingeniería				
Funciones del producto				
Actividades automatizables del proyecto de software.				

2.3.4. Proyecto

Dentro de la gestión del proyecto de software existen varios parámetros que permiten evaluar si va por el camino correcto, por lo que John Reel[10] propone diez señales que indican que un proyecto de sistemas de información esta en peligro:

1. El personal de software no comprende las necesidades del cliente.
2. El ámbito del producto esta definido pobremente.
3. Los cambios están mal realizados.
4. La tecnología elegida cambia.
5. Las necesidades del negocio cambian o están mal definidas.

[10] John S Reel: Dr. Reel posee BS and Ph.D. Trabajo en un proyecto de investigación y desarrollo dedicada a la tecnología de seguridad de información. Publico un articulo ((*Critical Success Factors in Software Projects*, May/June 1999) en la revista de software de la IEEE.

6. las fechas de entrega no son realistas.
7. Los usuarios se resisten.
8. Se pierden los patrocinadores.
9. El equipo del proyecto carece del personal con las habilidades apropiadas necesarias.
10. Los gestores evitan buenas prácticas.

Reel sugiere también cinco prácticas para evitar caer en los diez puntos anteriores:

1. **Empezar con el pie derecho:** consiste en entender claramente cuales son las expectativas del la organización respecto al software que se desarrollara, esto toma mas fuerza cuando se realiza una selección adecuada del personal involucrado.
2. **Mantenerse:** El gestor de proyecto debe esforzarse por mantener al personal motivado para realizar su trabajo, así como mantener un ambiente de seguridad laboral y colaboración del grupo de trabajo.
3. **Seguimiento del progreso:** El progreso de un proyecto de software se mide a lo largo de todo el proyecto, es por eso que el administrador debe determinar las métricas apropiadas para evaluar la evolución del mismo.
4. **Toma de decisiones inteligentes:** El administrador del proyecto debe identificar y eliminar riegos evidentes.
5. **Realizar análisis (posterior):** Es aconsejable realizar una retroalimentación de lo obtenido en el producto final del proyecto.

2.4. Factores para la calidad del software

Debido a que el software es un producto intangible, los factores que influyen en la calidad de un producto de software son perspectivas subjetivas respecto al software. Es así que se puede evaluar la calidad del software considerando los siguientes puntos de vista:

- Usuario
- Experto de software
- Desarrollador

Los factores que cada uno de estos considera para determinar la calidad de un producto de software se presentan en la Tabla 2.2.

Tabla 2.2. Factores de la Calidad del Software

Puntos de vista	Factores
Usuario	Facilidad de utilización Interfaces amigables Operación correcta y completa Operación rápida Existencia de ayudas
Experto de software	Operación / Plataforma Rendimiento Expansión Integración con otros módulos
Desarrollador	Reutilizable Utilización de estándares Distribución cliente servidor Facilidad de acoplamiento con otros módulos Documentación

- **Facilidad de utilización:** Los procesos deben ser claramente especificados en la operación del sistema, es decir que los procesos

que se realizan en el sistema debe estar estrechamente relacionados con los que se realizan en el mundo real.

- **Interfaces amigables:** El programa debe estar desarrollado de tal forma que las funciones que se presentan en el sistema deben manejarse de manera que el usuario realice el menor esfuerzo para memorizar operaciones del sistema.
- **Operación correcta y completa:** El sistema debe hacer lo que debe hacer de manera correcta.
- **Operación rápida:** Las respuestas deben entregarse de manera oportuna.
- **Existencia de ayudas:** Las ayudas para la utilización del sistema son esenciales en el entrenamiento y posterior operación del sistema.
- **Operación / Plataforma:** Este factor se refiere a la operación del sistema en las plataformas especificadas por el usuario.
- **Rendimiento:** Quizá el factor más importante sea el que evalúa los recursos utilizados por el sistema para su operación y la carga operativa que este representa para el computador en el que se va a ejecutar.
- **Expansión Integración con otros módulos:** Muchas veces un sistema necesita acoplarse con otros módulos ya desarrollados o por desarrollar, por lo que el sistema debe ofrecer facilidades para que esta integración se lleve a cabo.
- **Reutilizable:** El código fuente, análisis y diseños deben ser reutilizables de manera que se pueda obtener resultados similares en situaciones similares.

- **Utilización de estándares:** Los estándares son un factor importante en la evaluación de la calidad de cualquier producto, además de ofrecer un mayor entendimiento sobre el funcionamiento del sistema a distintas personas
- **Distribución cliente servidor:** Actualmente los sistemas tienden a distribuirse en clientes y servidores, no todos los sistemas requieren esta opción, pero debe contemplarse esta opción al momento de construir el sistema.
- **Facilidad de acoplamiento con otros módulos:** El diseño del sistema debe permitir que nuevos módulos compartan la información proporcionada por el sistema e interactúen de manera correcta y confiable.
- **Documentación:** La documentación de los procesos realizados en cada fase del desarrollo de software es importante para lograr resultados similares en condiciones similares.

Estos factores han sido considerados por varios autores para la evaluación de la calidad en el software, a continuación se presentan algunos de ellos.

2.4.1. Factores del modelo de McCall[11]

El modelo de McCall organiza los factores en tres ejes o puntos de vista desde los cuales el usuario puede contemplar la calidad de un producto:

- Operación del producto
- Revisión del producto
- Transición del producto

[11] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 324.

El modelo de McCall se basa en 11 factores de calidad, que se organizan en torno a los tres ejes:

Tabla 2.3. Factores de la calidad de McCall

Punto de vista	Factores
Operación de producto	Facilidad de uso Integridad Corrección Fiabilidad Eficiencia
Revisión del producto	Facilidad de mantenimiento Facilidad de prueba Flexibilidad
Transición del Producto	Facilidad de reutilización Interoperabilidad Portabilidad

La representación más común de los tres pilares de McCall se muestra en la Fig. 2.3.

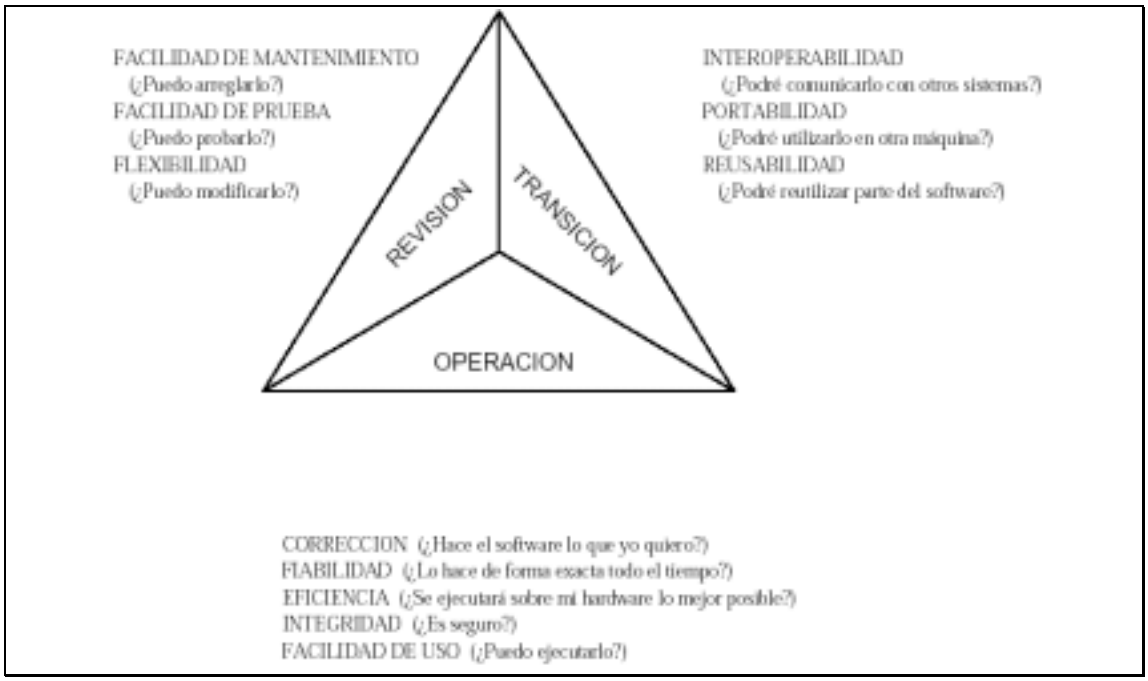


Fig. 2.3. Pilares de Mc.Call

Los factores de Mc.Call se definen como sigue (Tabla 2.4):

1. **Corrección**: Hasta qué punto un programa cumple sus especificaciones y logra los objetivos del usuario. Por ejemplo, si un programa debe ser capaz de sumar dos números y en lugar de sumar los multiplica, es un programa incorrecto. Es quizás el factor más importante, aunque puede no servir de nada sin los demás factores.
2. **Fiabilidad**: Hasta qué punto se puede esperar que un programa lleve a cabo su función con la exactitud requerida (sin errores).
3. **Eficiencia**: Se refiere a la cantidad de código y de recursos informáticos (CPU, memoria) que precisa un programa para desempeñar su función.
4. **Integridad**: Hasta qué punto se controlan los accesos ilegales a programas o datos. Un programa que permite el acceso de personas no autorizadas a ciertos datos carece de integridad.
5. **Facilidad de uso:** El esfuerzo necesario para aprender a operar con el sistema, preparar la entrada de datos e interpretar las salidas (resultados) del mismo.
6. **Facilidad de mantenimiento**: El esfuerzo necesario para localizar y corregir defectos en un programa que aparecen durante su funcionamiento.
7. **Facilidad de prueba**: El esfuerzo necesario para probar un programa y asegurarse que realiza correctamente su función.
8. **Flexibilidad**: El esfuerzo necesario para modificar un programa que ya esta en funcionamiento cuando cambian sus especificaciones.

Tabla. 2.4 Criterios de calidad Mc.Call

Factores	Criterios
Facilidad de uso	Facilidad de operación Facilidad de comunicación Facilidad de aprendizaje
Integridad	Control de accesos Facilidad de auditoria
Corrección	Completitud Consistencia Trazabilidad
Fiabilidad	Precisión Consistencia Tolerancia a fallos Modularidad Simplicidad
Eficiencia	Eficiencia en ejecución Eficiencia en almacenamiento
Facilidad de mantenimiento	Modularidad Simplicidad Consistencia Concisión Auto descripción
Facilidad de prueba	Modularidad Simplicidad Auto descripción Instrumentación
Flexibilidad	Auto descripción Capacidad de expansión Generalidad Modularidad
Reusabilidad	Auto descripción Generalidad Modularidad Independencia entre sistema y software Independencia del hardware
Interoperabilidad	Modularidad Compatibilidad de comunicaciones Compatibilidad de datos
Portabilidad	Auto descripción Modularidad Independencia entre sistema y software Independencia del hardware

9. **Portabilidad**: El esfuerzo necesario para transferir el programa de un entorno hardware/ software a otro entorno diferente.
10. **Facilidad de Reutilización**: Hasta qué punto se puede transferir un módulo o programa del presente sistema a otra aplicación en relación

al empaquetamiento y alcance de las funciones que realiza el mismo, y con qué esfuerzo.

11. **Interoperabilidad**: El costo y esfuerzo necesario para hacer que el software pueda operar conjuntamente con otros sistemas o aplicaciones (software externos).

Cada uno de estos factores se descompone, a su vez, en criterios. En el modelo de Mc.Call se definen un total de 23 criterios o métricas, con el siguiente significado:

1. **Facilidad de operación:** Determina la facilidad de operación del software.
2. **Facilidad de comunicación:** Proporciona al usuario entradas y salidas fácilmente asimilables.
3. **Facilidad de aprendizaje:** Facilitan la familiarización inicial del usuario con el software y la transición desde el modo actual de operación.
4. **Control de accesos:** Proporciona control de acceso al software y los datos que maneja.
5. **Facilidad de auditoria:** Facilita el registro y la auditoria de los accesos al software.
6. **Eficiencia en ejecución:** Minimiza el tiempo de procesamiento.
7. **Eficiencia en almacenamiento:** Minimiza el espacio de almacenamiento necesario.
8. **Precisión:** Proporciona el grado de precisión requerido en los cálculos y los resultados.
9. **Consistencia:** Proporciona uniformidad en las técnicas y notaciones de diseño e implementación utilizadas.

10. **Tolerancia a fallos:** Posibilita la continuidad del funcionamiento bajo condiciones no usuales.
11. **Modularidad:** Proporciona una estructura de módulos altamente independientes.
12. **Simplicidad:** Posibilita la implementación de funciones de la forma más comprensible posible.
13. **Completitud:** Proporciona la implementación completa de todas las funciones requeridas.
14. **Trazabilidad (Rastreabilidad):** Proporciona una traza desde los requisitos a la implementación con respecto a un entorno operativo concreto.
15. **Auto descripción:** Proporciona explicaciones sobre la implementación de las funciones.
16. **Capacidad de expansión:** Posibilita la expansión del software en cuanto a capacidades funcionales y datos.
17. **Generalidad:** Proporciona amplitud a las funciones implementadas.
18. **Instrumentación:** Posibilita la observación del comportamiento del software durante su ejecución (para facilitar las mediciones del uso o la identificación de errores).
19. **Independencia entre sistema y software:** Determina su independencia del entorno operativo.
20. **Independencia del hardware:** Determina su independencia del hardware.
21. **Compatibilidad de comunicaciones:** Posibilita el uso de protocolos de comunicación e interfaces estándar.

22. **Compatibilidad de datos:** Posibilita el uso de representaciones de datos estándar.
23. **Concisión:** Posibilita la implementación de una función con la menor cantidad de código posible.

2.4.2. Factores de calidad ISO 9126[12]

Este estándar ha sido desarrollado como una forma de definir atributos clave para la calidad del software, los que se definen a continuación:

- **Funcionalidad:** Grado de satisfacción de los requerimientos en función de los atributos de corrección, interoperabilidad, conformidad, completitud y seguridad.
- **Confiabilidad:** Cantidad del tiempo que el software está dispuesto para su uso, referido por su atributos de tolerancia a fallos, madurez[13], y facilidad de recuperación.
- **Usabilidad:** Grado de facilidad en el uso del software, reflejado por los atributos de facilidad de aprendizaje, facilidad de comprensión y operabilidad.
- **Eficiencia:** Grado de optimización en el uso de los recursos del sistema, determinado por los atributos de tiempo de uso y recursos utilizados.
- **Facilidad de mantenimiento:** Facilidad con que una modificación puede ser efectuada considerando los atributos de facilidad de cambio, facilidad de análisis, estabilidad y facilidad de prueba.

[12] ISO 9126: Trata sobre la definición de las características de la calidad que se utilizaran en la evaluación de productos de software.
[13] Madurez: Se refiere al grado de preparación del software, en otras palabras, expresa el grado en que los resultados de un proyecto de software son repetibles.

- **Portabilidad:** La facilidad con que el software puede ser llevado de un entorno a otro, considerando los atributos de facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio.

2.4.3. Factores de Hewlett Packard – FURPS [14]

Hewlett Packard ha definido 5 factores principales para la calidad del software a los que se les ha denominado con el acrónimo FURPS (Funcionalidad, Facilidad de uso (Usabilidad), Fiabilidad, Rendimiento y Capacidad de soporte):

- **Funcionalidad:** Consiste en la evaluación de las características, capacidades de un programa, además de las seguridades globales del sistema.
- **Usabilidad:** Consiste en la evaluación de características de un programa respecto a su relación con el operador, características como estética, consistencia y documentación general.
- **Fiabilidad:** Consiste en medir la frecuencia y gravedad de los fallos, la exactitud de las salidas (respuestas) del sistema, tiempo medio de fallos (TMDF) y capacidad de recuperación de un fallo.
- **Rendimiento:** Consiste en la evaluación de la velocidad de procesamiento, tiempo de respuesta, consumo de recursos y eficacia.
- **Capacidad de Soporte**: Es conocido también como mantenimiento, y combina la capacidad de ampliar el programa, adaptabilidad y servicios, así como la capacidad de hacer pruebas, compatibilidad, capacidad de configuración de software, facilidad de instalación de un sistema y la facilidad con que se pueden localizar problemas.

2.5. Estándares de Calidad del Software

Los estándares es un conjunto de criterios que permiten establecer un estudio comparativo entre lo idealmente esperado y lo obtenido. Dentro del software ocurre de manera similar, en tanto que se debe establecer una

[14] Furps: Factores de calidad desarrollado por Hewllet Pakard proveniente del ingles Functionality, Usability, Reliability, Performance, and Supportability.

comparación entre los requerimientos iniciales y el producto final del software, dicha concordancia permite entregar cierto grado de calidad al producto de software. Lo más relevante dentro de la formulación de estándares es que éstos se conviertan en criterios legítimos, es decir que sean reconocidos por toda la comunidad como validos y valiosos

2.5.1. Familia de estándar de calidad ISO 9000

Son un conjunto de normas internacionales editadas y revisadas periódicamente por la Organización Internacional de Normalización (ISO) y La Comisión Internacional de la Electrotécnica (IEC) sobre la garantía, gestión y control de calidad de procesos. Debido a que es una norma internacional, permite ahorrar tiempo y dinero evitando certificar la calidad en algún otro estándar local o particular de una empresa.

Inicialmente se contaba con 3 normas, las normas ISO 9001, 9002 y 9003, que especifican los requerimientos necesarios para establecer la confianza en la calidad de un proveedor. La aplicación de cada una depende del alcance de la relación cliente proveedor. La ISO 9001, siendo la más extensa de las 3, abarca desde el diseño del producto o servicio hasta su entrega y soporte al cliente. La ISO 9003 siendo la menos extensa, abarca únicamente las actividades de inspección y prueba del producto antes de entregarse al cliente. Sin embargo, estas 3 normas fueron resumidas en la norma ISO 9001 – 2000.

La ISO 9000 no normaliza el sistema de gestión de calidad ya que esto depende del tipo de sector tamaño de la empresa organización interna, sino que normaliza las verificaciones que se van a realizar sobre el sistema de calidad.

Es recomendable iniciar aplicando la norma ISO 9004 (Gestión de Calidad y Elementos del Sistema de Calidad) ya que es una directriz para el establecimiento o ampliación de un sistema de calidad. Una vez implantado un sistema de calidad es aconsejable iniciar con los modelos de garantía de calidad de la ISO 9001 -2000.

Uno de los problemas con el estándar ISO 9001 es que no es especifico para una industria, es decir que esta expresado en términos generales y puede ser interpretado por desarrolladores de diversos productos

2.5.1.1. El Estándar de Calidad ISO 9001-2000

Dentro de la familia de estándares de ISO 9000 existen varios estándares que se aplican a la industria. ISO 9001-2000 y se llama (Sistemas de Calidad Modelo para el Aseguramiento de Calidad, el Diseño, desarrollo producción instalación y servicio) es un estándar que describe el sistema de calidad utilizado para mantener el desarrollo de un producto que implique diseño.

La ISO 9001-2000 cuenta con un conjunto de estándares de apoyo:

- ISO 9000-3 (Guías para la aplicación de la norma ISO 9001 al desarrollo suministro y mantenimiento del software) este documento interpreta el ISO 9001-2000 para el desarrollador de software, es decir que no cambia los requerimientos de la ISO 9001-2000 sino que los amplia y aclara.
- ISO 9004-2 (Administración de la calidad y elementos del sistema de calidad – parte 2 –) este documento proporciona las directrices para el servicio de las facilidades del software como soporte a usuarios.
    a. ISO 12207 Procesos del Ciclo de Vida del software *(Ver Anexo B)*

b. ISO/IEC 2126 Características de la calidad de un producto de software
c. ISO/IEC 11129 Productos de software evaluación y pruebas
d. ISO/IEC 14102 Guía para la evaluación y selección de herramientas Case. *(Ver Anexo B)*

La Norma ISO 9001-2000 especifica los requisitos que debe cumplir un sistema de calidad cuando es necesario demostrar la capacidad de un proveedor para diseñar y suministrar productos conformes. *(Ver Anexo B)*

2.5.1.2. Estándar de calidad ISO 9000-3

Después de la reunión de la serie de normas ISO 9001, 9002 y 9003 en ISO 9001:2000, se revisa la ISO 9000-3 (Guías para la Aplicación de ISO 9001 al Desarrollo, Provisión y Mantenimiento de Software). La primera edición de ISO/IEC 90003 cancela y reemplaza a la ISO 9000-3:1997, que ha sido actualizada de conformidad con ISO 9001:2000. Esta normativa guía ha sido precursora para esquemas como TICK-IT[15], ITQMS[16] y otros bajo el cual una empresa en el sector de "software" pueda certificar ISO 9001. La revisión ISO 9000-3 2002 incluye referencia a los siguientes esquemas / normativas:

- ISO 12207 "Information Technology" - "Software Life Cycle Processes" *(Ver Anexo B)*
- ISO 15504 "Information Technology" - "Software Process Assessment"
- ISO 15939 "Information Technology" - "Software Measurement Process Framework"

[15] Tick-IT: Para que una organización obtenga la certificación TickIT, debe demostrar que reúne los requisitos de ISO 9001 con respecto al desarrollo del software.
[16] ITQMS: Esquema certificado para organizaciones de la industria del software similar a ISO 9001.

- ISO 9126 "Software Engineering" - "Product Quality - Quality Model and Metrics", véase también Libro 2 de la serie TL 9000 - 2001.

a) Particularización de los requisitos según ISO 9000-3

En este punto se explican los requisitos más importantes de ISO mencionados anteriormente.

1. Responsabilidades gerenciales

Dentro de las actividades que se deben cumplir a nivel gerencial se pueden destacar la implantación de políticas de calidad y la estructura organizacional.

a. Responsabilidad gerencial del proveedor:

Se considera proveedor a la persona u organización encargada de la producción de software, además, que su responsabilidad es definir las políticas y sus objetivos referentes a la calidad para lo cual debe realizar las actividades de documentación detallada del producto de software.

Se debe realizar los siguientes puntos:

- Establecimiento de responsabilidades, autoridades y relaciones entre todo el personal que dirige, realiza y verifica cualquier trabajo relacionado con la calidad.
- La provisión de los medios adecuados para verificación y la asignación de personal calificado para ello.
- La designación del representante de la dirección con autoridad y responsabilidad suficiente para asegurar que se apliquen los requisitos de la norma.

- La propia gerencia debe revisar a intervalos apropiados el sistema de calidad adoptado. Normalmente esto implica efectuar la evaluación de los resultados de las auditorias internas.

Todas las actividades de verificación y validación, incluyendo las de liberación del producto, su implantación y su período de puesta en marcha, deberán estar documentadas (incluyendo como mínimo: escenarios de prueba, casos probados, tipos de pruebas utilizadas, usuarios involucrados en aprobación). En las actividades de verificación que se realicen conjuntamente con el comprador (las mencionadas en el Plan de Desarrollo del Proyecto).

b. Responsabilidad gerencial del comprador

El comprador deberá colaborar con el proveedor para proporcionar a tiempo toda la información necesaria y resolver las situaciones pendientes de arreglo. El responsable designado por el Organismo, deberá contar con la autoridad requerida para:

- Definir y aprobar los requerimientos al proveedor
- Responder a sus preguntas
- Aprobar sus propuestas
- Redondear acuerdos con el proveedor
- Contar con los elementos técnicos que le permitan catalogar si la organización del proveedor le permite cumplir con los acuerdos o requerimientos que se le realicen
- Definir los criterios y procedimientos de aceptación

- Resolver sobre el accionar frente a aquellos productos resultantes de alguna fase que a su entender considere “no-conformes”
- Evaluar independientemente la integridad de cada uno de los módulos que componen el sistema

c. Revisiones conjuntas

Se acordara como realizar y documentar:

- Los resultados de las verificaciones
- Los resultados de los ensayos de aceptación
- El estudio de la conformidad del “software” con los requisitos iniciales de especificación

2. Sistema de calidad

Se trata de un proceso integrado a lo largo de todo el ciclo de vida, que permite asegurar que la calidad se construye a medida que avanza el desarrollo y no que se descubre al final del proceso.

Este proceso deberá estar establecido y documentado. El proveedor debe asegurar y demostrar su aplicación efectiva ante instancias de certificación.

Para cada desarrollo de software se debe contar con el plan de calidad independiente.

a. Auditorias internas del sistema

El proveedor debe contar con un sistema completo de auditorias internas planificadas (plan anual de auditorias) y registradas, que permitan verificar el efectivo cumplimiento de la actividades adoptadas, y para determinar la efectividad del sistema.

Se debe realizar el seguimiento de las auditorias y la responsabilidad del personal ejecutivo encargado del área auditada sobre las posibles acciones correctivas.

b. Sistema de Calidad en las actividades del Ciclo de Vida

Todo desarrollo de “software” deberá estar organizado de acuerdo a algún modelo de ciclo de vida. Algunos de los más conocidos en software son: Cascada (1970), Transformación (1975), Espiral (1989), Pileta (1990), entre otros. La aplicación de ISO 9000 es independiente del modelo de ciclo de vida seleccionado.

3. Revisión de contratos (calidad en la comercialización)

Se establecerán los procedimientos que permitan asegurar que el objeto del contrato está definido y documentado, además de permitir la identificación de riesgos o contingencias. Adicionalmente asegura que el comprador y el proveedor tienen la capacidad de cumplir las obligaciones contractuales y que la terminología esté acordada por ambas partes.

Será necesario contar con procedimientos para las modificaciones y revisiones de los contratos, al igual que para el establecimiento de canales de comunicación con los clientes.

4. Especificación de requisitos del comprador

Recurriendo a la Ingeniería de Software: todo sistema complejo de ingeniería, debe ser especificado. Entendiendo por “especificación” a la declaración del acuerdo entre el proveedor de un servicio (el implementador), y el consumidor del mismo (el usuario), sobre lo qué debe ser provisto por la implementación.

Una especificación se utilizará de manera correcta cuando contenga los siguientes puntos:

- El proveedor deberá disponer, de un conjunto completo y no ambiguo[17] de requerimientos funcionales, los cuales sean factibles de validar al momento de aceptación del producto. Los mismos deben referirse como mínimo a: requerimientos sobre la funcionalidad del software, sobre su comportamiento frente a errores o excepciones, sobre seguridad, confiabilidad de procesos, protección de datos y su integridad, interfaces entre sistemas, hardware, software de base, restricciones de diseño, comunicaciones, desempeño, entre otros.
- La documentación de las especificaciones también estará sujeta al control sobre documentación y al control sobre la configuración de software.

a. Ingeniería de requerimientos

1. Narrativa: documentación de las reuniones de valoracion mediante actas formalmente aprobadas por los usuarios
2. Listas de eventos extractadas de las narrativas anteriores

b. **Herramientas de representación**

Estas herramientas de representación se utilizan como un marco documental para especificar requerimientos, como por ejemplo IEEE 830[18] Sección 3 *(Ver Anexo A)*.

[17] Ambiguo: Que puede entenderse de varios modos o admitir distintas interpretaciones y dar por consiguiente motivo a duda, incertidumbre o confusión., Diccionario Enciclopédico Siglo XXI.
[18] IEEE 830: Estándar para la especificación de requerimientos de software.

5. Planificación del desarrollo

El objetivo del plan de desarrollo es definir un proceso o metodología para transformar la especificación de los requisitos del usuario en un producto de “software”. Dicho plan deberá cubrir lo siguiente:

- La declaración de objetivos y definición del proyecto
- La organización de los recursos
- Las fases de desarrollo
- Los elementos de entrada y los de salida a cada fase
- El calendario del proyecto
- Los controles sobre el progreso del trabajo
- Las actividades de revisión, verificación y validación del diseño
- Los métodos y herramientas de desarrollo (reglas, convenciones, herramientas, gestión de la configuración)

Los otros posibles planes relacionados al plan de desarrollo, serán el plan de calidad, el plan de gestión de la configuración, el plan de integración y el plan de pruebas.

Para planificación es aconsejable el uso de paquetes que permiten diagramación de redes Pert, Gantt, con lo cual se facilita la gestión de los diferentes componentes de un proyecto.

6. Planificación de Calidad

El plan de calidad debe contener:

- Los objetivos de calidad expresados en términos mensurables (cuando sea posible)

- Planificación detallada de ensayos, verificaciones y validaciones; calendarios, recursos y autoridades para la aprobación
- Lista de responsables de revisar y ensayar, gestionar la configuración y el control de cambios y acciones correctivas a los defectos detectados

7. Diseño y realización

Estas actividades son las que transforman la especificación de los requisitos en un producto de “software”.

El diseño Constituye una de las etapas fundamentales en la producción del “software”, su función es la de traducir las expectativas del cliente (dadas como una descripción cualitativa del producto) a especificaciones técnicas de materiales, de procesos y de productos.

Estas especificaciones técnicas deben ser de fácil interpretación para la “realización” del producto y deben poder verificarse objetivamente. Un producto diseñado inadecuadamente puede dañar la imagen de la organización o impedir su crecimiento; en particular si se liberan productos inseguros, no confiables o difíciles de reparar y mantener.

Se debe tomar en cuenta lo siguiente:

- Entradas y salidas de la etapa
- Reglas de diseño e interfaces internas entre componentes
- Metodología sistemática de diseño
- Modularización del producto
- Grado de reusabilidad de sus componentes.

Realización es la función encargada de llevar a cabo los procesos que conducen al logro de los productos de calidad requerida.

Al igual que todos los procesos que pueden afectar a la calidad, los procesos de realización deberán ser planificados, aprobados, monitoreados y controlados.

Además de estas actividades y las usuales del desarrollo de software, deberán ser considerados aspectos tales como:

- Reglas consistentes y adecuadas de programación
- Convención para las denominaciones y codificación usada

Revisiones para asegurar que se cumplen los requisitos y que se lleva a cabo la metodología planificada. Estas revisiones deben ser exámenes formales, completos y sistemáticos sobre los requisitos del diseño, las cuales serán evaluadas a través de métricas, con la finalidad de evaluar si cumple con los requisitos iniciales, permitiendo identificar problemas actuales o potenciales, y proponer soluciones.

2.5.2. CMM (Capability Maturity Model)

Es una aplicación de sentido común de los conceptos de gestión de procesos y mejora de la calidad al desarrollo y mantenimiento del software. Su objetivo es describir el manejo y la práctica de la ingeniería de software en un producto, estructurándolos en un esquema de madurez. La madurez del proceso de desarrollo del software ayuda a predecir la capacidad de alcanzar metas en un proyecto.

2.5.2.1. Estructura

Este modelo está constituido por cinco niveles de madurez, los que a su vez están compuestos por un grupo de áreas claves del proceso (KPA Key Process Áreas), organizadas de acuerdo a sus características.

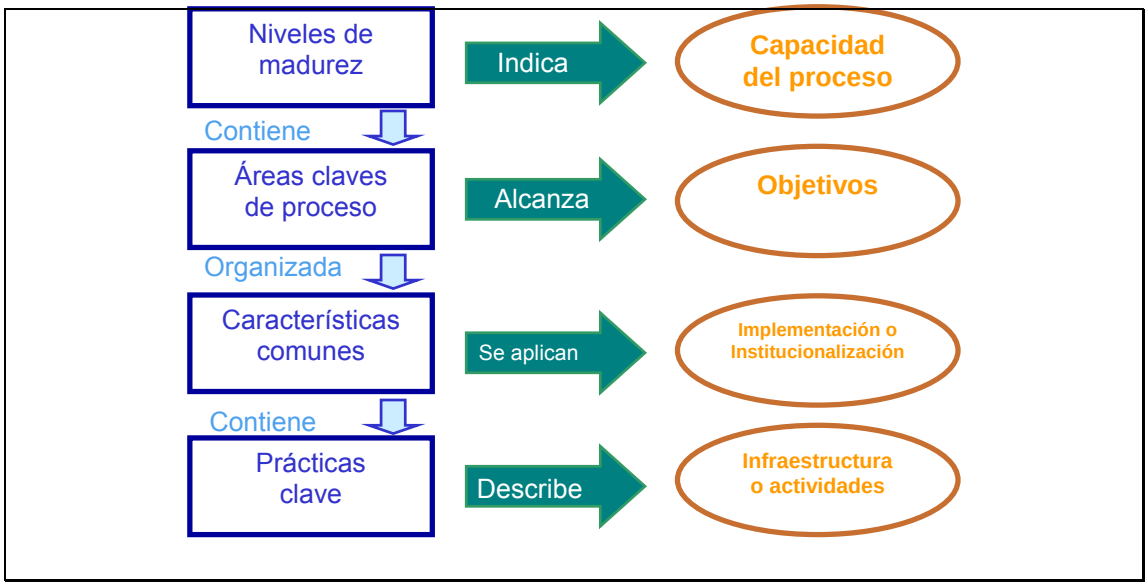


Fig.2.4 Niveles de madurez del CMM

2.5.2.2. Niveles de madurez.

Estos niveles definen una escala para medir cada proceso del software.

- **NIVEL 1. (Inicial):** En este nivel se definen pocos procesos y el cumplimiento de los objetivos del desarrollo (Costo, tiempo y recursos) depende de esfuerzos individuales lo que provoca que existan falta de documentación e inestabilidad en el desarrollo y el mantenimiento.
- **NIVEL 2. (Repetible):** Se establece en este nivel un proceso de administración de proyectos básico para realizar un seguimiento de costos, planificación y funcionalidad, sin embargo no existe una disciplina de procesos establecida para el cumplimiento de objetivos,

sino que se basa en la experiencia en proyectos anteriores. Existen aún cajas negras pero ya son revisadas y definidas.

- **NIVEL 3. (Definido):** El proceso del software de las actividades de gestión y de la ingeniería se documenta, se estandariza y se integra dentro de un proceso de software de toda la organización. Existe un proceso de software bien definido que se basa en el entendimiento de las actividades, roles y responsabilidades.
- **NIVEL 4. (Gestionado):** El proceso de software es medido y opera dentro de límites cuantificables. Se cumplen con planes y programas de mejoramiento además de diferenciar los procesos principales de los de soporte.
- **NIVEL 5. (Optimización):** Se practica el mejoramiento continuo de los procesos junto con innovaciones tecnológicas.

2.5.2.3. Áreas clave del proceso (Key Area Process KAP)

Cada nivel de madurez se descompone en un conjunto de áreas clave de proceso que indican el área en la que se debe centrar la organización para desarrollar sus procesos de software. Cada área clave de proceso identifica los puntos que deben ser considerados para alcanzar un nivel de madurez.

Existen 18 KPA´s repartidas entre los niveles de madurez con excepción del *Nivel 1*, esto se debe a las características del mismo. *(Ver Anexo C)*

2.5.2.4. Características Comunes

Las Áreas Clave de Procesos son organizadas en características comunes que son atributos que indican si la implantación de un Área Clave de Proceso es efectiva, repetible y duradera. *(Ver Anexo C)*

2.5.2.5. Prácticas clave

Cada área clave de proceso está descrita en términos de prácticas clave que contribuyen con la satisfacción de los objetivos. Las prácticas clave describen la infraestructura y actividades que mayor contribuyen a una efectiva implantación e institucionalización de las áreas clave de procesos.

Las prácticas clave describen lo que debe realizarse, pero no deben ser interpretados basados en la manera en la que los objetivos son alcanzados. Las prácticas alternativas deben alcanzar los objetivos de las áreas de procesos clave. Las prácticas clave deben ser interpretadas con un juicio razonable de forma que los objetivos de las áreas clave de procesos son alcanzados efectivamente. *(Ver Anexo C)*

2.5.3. SPICE (Software Process Improvement Capability dEtermination)

Este modelo es similar al CMM ya que es utilizado para la evaluación y mejora de procesos del software y determinación de la capacidad, es aplicable a cualquier organización o empresa que quiera mejorar la capacidad de cualquiera de sus procesos de software, cabe recalcar que es independiente del ciclo de vida, metodología, tecnología que la empresa disponga.

SPICE es un marco para métodos de evaluación no es un método o modelo en sí.

2.5.3.1. Componentes de SPICE

Los componentes del SPICE se presentan en la Fig. 2.5 y se explican a continuación:

- **PI: Conceptos y guía de introducción**

Explica los requisitos que contienen el estándar y la valoración de los requerimientos, a la construcción y selección de herramientas de apoyo, y a la construcción de procesos extendidos.

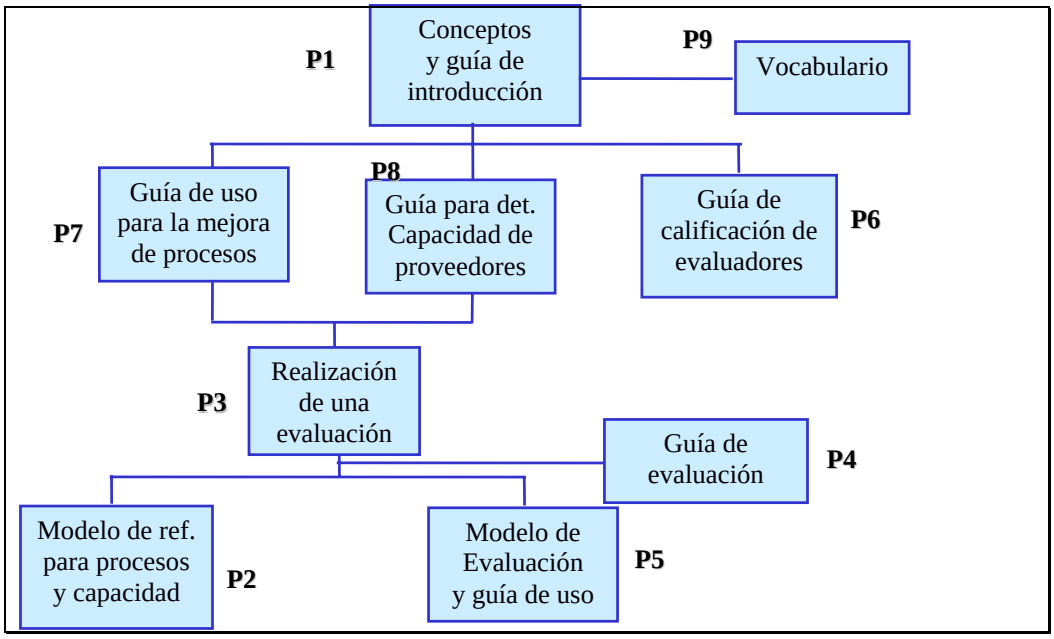


Fig. 2.5 Modelo del SPICE

- **P2: Modelo de Referencia**

Describe los procesos que una empresa debe realizar para comprar, suministrar, desarrollar, operar, mantener y soportar el software así como los atributos que caracterizan la capacidad de estos procesos.

Proporciona una base para medir la capacidad de los procesos, en función del grado de consecución de sus atributos. Tiene dos dimensiones procesos y capacidad.

1. Dimensiones de Procesos

Contiene los procesos que se van a ser evaluados, corresponden a los procesos del ciclo de vida del software.

Se agrupan en categorías en función del tipo de actividad al que se aplican:

- CUS: Cliente proveedor
- ENG: Ingeniería
- SUP: Soporte
- MAN: Gestión
- ORG: Organización

a. Dimensión de procesos CUS

Esta Categoría esta formada por los procesos que afectan al cliente directamente y soportan el desarrollo y la transición del software al cliente y permiten la correcta operación y uso del producto y/o servicio del software, esta dimensión tiene ocho actividades relacionadas:

- CUS1: Adquisición de productos de software y/o servicios.
- CUS2: Establecimiento de contratos.
- CUS3: Identificar las necesidades del cliente
- CUS4: Realizar auditorias y revisiones conjuntas.
- CUS5: Entrega e instalación de software.
- CUS6: Mantenimiento del software.
- CUS7: Proporcionar servicios al cliente.
- CUS8: Valorar la satisfacción al cliente.

b. Dimensión de procesos ENG

Están formados por procesos que directamente especifican, implementan o mantienen el producto de software, su relación con el sistema y su documentación. Esta dimensión tiene 7 actividades relacionadas:

- ENG 1: Análisis y diseño de requerimientos del sistema
- ENG 2: Análisis de requerimientos del software
- ENG 3: Diseño de software
- ENG 4: Construcción del software
- ENG 5: Integración y pruebas del software
- ENG 6: Integración y pruebas del sistema
- ENG 7: Mantenimiento del software y del sistema

c. Dimensión de procesos SUP

Esta formada pro procesos que dan soporte a cualquiera del resto de procesos en distintos puntos del ciclo de vida del software. Estas dimensiones tienen 5 actividades relacionadas:

- SUP 1: Documentación
- SUP 2: Gestión de la configuración del software
- SUP 3: Garantía de la calidad del software
- SUP 4: Resolución de problemas
- SUB 5: Realizar revisiones conjuntas

d. Dimensión de procesos MAN

Formada por procesos utilizados en la gestión de cualquier tipo de proceso en ciclo de vida en el software. Esta dimensión tiene 4 actividades relacionadas:

- MAN 1: Gestionar procesos
- MAN 2: Gestionar proyectos
- MAN 3: Gestionar la calidad
- MAN 4: Gestionar los riesgos

e. Dimensión de procesos ORG

Esta formada por procesos que establecen los objetivos del negocio de la organización. Esta dimensión tiene 7 actividades relacionadas:

- ORG 1: Alineamiento de la organización
- ORG 2: Establecimiento de procesos
- ORG 3: Evaluación de proceso
- ORG 4: Mejora del proceso
- ORG 5: Gestión de recursos humanos
- ORG 6: Infraestructura
- ORG 7: Reutilización

2. Dimensión de capacidad

Define una escala de medida para determinar la capacidad de cualquier proceso. Consta de seis niveles de capacidad.

- 0: No realizada, no hay productos de trabajo identificables.
- 1: Realizado informalmente, planificación y seguimiento dependientes del conocimiento individual. Productos de trabajo identificables.
- 2: Planificada, verificada de acuerdo a los procedimientos especificados
- 3: Bien definida, procesos bien definidos y documentados

- 4: Controlada, cuantitativamente, medidas detalladas de realización, predicción, etc. Productos de trabajo evaluado cuantitativamente
- 5: Mejorada continuamente, objetivos cuantitativos de eficiencia basados en objetivos de negocio.

SPICCE utiliza dos tipos de prácticas, prácticas base y prácticas genéricas.

a. **Prácticas base:** Cada proceso tiene un conjunto de prácticas base asociadas, que describen las actividades esenciales de un proceso específico. La realización de las prácticas base indica el grado de alcance de la finalidad del proceso.
b. **Practicas genéricas:** Cada atributo de proceso tiene un conjunto de prácticas de gestión asociadas, son las que implementan un proceso de una manera general. La realización de las prácticas genéricas indica la consecución del atributo en esa instancia del proceso.

2.5.3.2. Evaluación de atributos

Los atributos de un proceso se evalúan con N (Not), P (Partially), L (Largely) y F (Fully), siendo:

- N (No alcanzado - 0% a 15%): Refleja poca o ninguna evidencia de la consecución del atributo.
- P (Parcialmente alcanzado - 16% a 50%): Evidencia de un enfoque sistemático y de la consecución del atributo. aunque algunos aspectos de la consecución pueden ser impredecibles
- L (Ampliamente alcanzado - 51% a 85%): Evidencia de un enfoque sistemático y de una consecución significativa del atributo. La realización del proceso puede variar en algunas áreas.

- F (Totalmente alcanzado - 86% a 100%): Evidencia de un enfoque completo y sistemático y de la consecución plena del atributo

2.6. Aseguramiento de la Calidad del Software

Reifer define la Garantía de Calidad como *"un conjunto de actividades de planificación, estimación y supervisión de las actividades de desarrollo, que se realizan de forma independiente al equipo de desarrollo, de tal forma que los productos software resultantes cumplen los requisitos establecidos"*[19].

Bersoff la define como *"un conjunto de procedimientos, técnicas y herramientas, aplicados por profesionales, durante el ciclo de desarrollo de un producto, para asegurar que el producto satisface o excede los estándares o niveles de calidad preestablecidos".*

El Aseguramiento de la calidad del software es un conjunto de actividades que permiten planificar, dirigir, controlar y evaluar el procesos del ciclo de vida del software para garantizar que *¨…el proceso y los productos del desarrollo satisfacen aceptablemente estándares de calidad."* [20]

Existen tres áreas que están bajo la responsabilidad del grupo de Garantía de Calidad:

TABLA 2.5 Áreas que están bajo la responsabilidad del grupo de Garantía de Calidad

	AREAS DE PROCESOS		
PROCESOS	**Ingeniería**	**Gestión**	**Calidad**
	Análisis de requisitos	Estimación	Prevención
	Diseño	Planificación	Detección y corrección
	Validación y verificación	Control y seguimiento	
	Mantenimiento	Gestión de riesgos	
		Relaciones con clientes	

[19] Reifer, D. J., Knudson, R. W., Smith, J., Final report: Software Quality Survey, American Society for Quality Control, Aerospace Industries Association of America, 1988.
[20] Ramón Mollineda, Tanja Vos. www.iti.upv.es/actualidadtic/2003/07/2003-07-calidad.pdf

2.6.1. Áreas del aseguramiento de la calidad del software

Las áreas del aseguramiento de calidad están dirigidas a sectorizar las responsabilidades en el proceso de aseguramiento del software y eventualmente proporcionar una guía para la asignación de responsabilidades de cada área de acuerdo al ámbito en el que se desenvuelve

2.6.1.1. Procesos de Ingeniería

Se encarga de los procesos relacionados directamente con las fases de la Ingeniería del software, las cuales son:

a. Gestión de Requisitos

Consiste en el levantamiento de requisitos, su especificación en un formato preestablecido, y en la utilización de técnicas como prototipos, entrevistas, etc., para alcanzar con exactitud lo que el cliente requiere, la revisión periódica de la consistencia entre requisitos y otros contenidos del desarrollo (diseño, código, manual de usuarios), y la gestión de cambios en los requisitos durante todo el proyecto.

Una gestión de requisitos mal planificada es una de las causas más frecuentes de que los proyectos se retrasen, sobrepasen sus presupuestos o tengan menos funcionalidad de la esperada.

Una gestión correcta y completa de requisitos debe permitir su uso como base para estimar, planificar, diseñar, implementar y verificar y validar el software.

b. Diseño

El Diseño es el proceso de definición de la arquitectura del sistema, de las estructuras de datos y de los algoritmos a emplear, antes de realizar la construcción del software.

Para alcanzar diseños de calidad se debe basar en el conocimiento de estilos como: estructurado u orientado a objetos y conceptos de modularidad, abstracción básica del diseño entre otros.

c. Implementación

Cuando se llega a la implementación dentro de un proceso correcto de software, la mayoría del trabajo creativo ya ha sido realizado, sin embargo, errores sutiles dentro de este proceso pueden provocar aumentar los costos del desarrollo del proyecto así como el tiempo.

La estandarización de las prácticas de implementación del software simplifican notablemente los esfuerzos de trabajo en grupo, en especial, aquellos orientados al mantenimiento del propio software o a la reutilización de código en futuros proyectos.

d. Mantenimiento

De acuerdo a IEEE 1219, el mantenimiento de software es el conjunto de actividades de modificación de un producto de software después de entregado, para corregir fallos, mejorar su rendimiento u otros atributos, o adaptar el producto a un entorno modificado.

En muchos casos, el mantenimiento es el proceso más largo del ciclo de vida. El mantenimiento de software es difícil de realizar y gestionar. Sin embargo, este proceso se simplifica notablemente si los procesos

primarios previos de ingeniería han sido correctamente realizados y documentados.

e. Verificación y Validación

El proceso de validación y de verificación (V&V) es el que permite revisar las actividades realizadas con la finalidad de encontrar el grado de cumplimiento de los objetivos:

- Verificación comprueba la consistencia del software con respecto a especificaciones y requisitos.
- Validación comprueba si lo que se ha especificado (e implementado) es lo que el usuario realmente desea,

Las tareas de V&V no solo se aplican a productos de software, sino también a otros productos resultantes del proceso del desarrollo.

Las pruebas realizadas en el proceso del software no deben ser utilizadas para determinar la ausencia de errores, ya que las pruebas pueden demostrar la presencia de errores. Una vez que se corrijan los errores localizados, la calidad del software mejorará.

Cuando se realiza un cambio en el software, se debe examinar el impacto del cambio sobre el sistema y considerar las actividades de V&V necesarias para garantizar calidad en el software.

2.6.1.2. Procesos de Gestión

Son los procesos relacionados con la administración del proyecto y los recursos entregados al mismo y gestionar riesgos e incertidumbres asociados al desarrollo.

a. Estimación

El proceso de estimación puede definirse a partir de tres pasos básicos: primero, estimar el tamaño del proyecto a partir de un análisis preliminar de requisitos; luego estimar el esfuerzo total (en unidades de tiempo) que requiere el desarrollo de un proyecto de tal tamaño; por último, estimar el tiempo de desarrollo del proyecto en función del esfuerzo estimado y del personal con el que se cuente para su realización.

La diferencia entre un procedimiento de calidad y otro improvisado es que el primero define metodologías para hacer estimaciones objetivas y contrastadas dando lugar a estimaciones precisas, mientras que en el segundo las estimaciones son resultado de análisis subjetivos que conducen a resultados poco confiables y de baja calidad.

b. Gestión de Riesgos

En el proceso de análisis del proyecto, surgen ciertos temas en los que se debe poner mayor atención los cuales se consideran riesgos para el proyecto. Al no atender adecuadamente estos puntos, el éxito del proyecto se verá afectado.

Una de las líneas esenciales de la gestión moderna de software es la gestión dinámica de riesgos. Este proceso periódico consiste en identificar y analizar cada riesgo, estimar su probabilidad de ocurrencia y su posible impacto en el cronograma, y definir un plan de gestión del mismo, el cual es un grupo de acciones orientadas a prevenir el riesgo o a corregir sus consecuencias, en función del proceso que resulte menos costoso. Una gestión global incluye además el mantenimiento de listas actualizadas de riesgos ordenados por peligrosidad, de forma que sea

posible centrarse en aquellas incertidumbres potencialmente más destructivas.

Un procedimiento de calidad para el desarrollo de software debe incluir una metodología de gestión de riesgos, así como un registro de riesgos y errores frecuentes en la organización que ayuden a evitar omisiones importantes.

c. Planificación

La planificación consta de dos partes: la división del proyecto en tareas y la asignación de recursos a tareas, es decir, ordenar las tareas en el tiempo, asignándoles recursos humanos y materiales para su realización.

El tiempo asignado a una tarea depende de múltiples factores: tamaño y complejidad de la tarea (productos de la estimación), grado de conocimiento o de incertidumbre que se tiene sobre ella (análisis de riesgos), y de la preparación y experiencia del personal que debe realizarla.

Es recomendable dentro de un procedimiento de calidad la existencia de una metodología con directrices para realizar planes de desarrollo, relacionada con las metodologías de elaboración de estimaciones y de gestión de riesgos.

d. Control y Seguimiento

Las actividades de control y seguimiento consisten en verificar que el progreso del proyecto se ajusta al plan y a los estándares, es decir, que se están cumpliendo los plazos, costos, y los objetivos de calidad.

Las tareas de control y seguimiento deben ser también planificadas. Sin ellas no es posible gestionar un proyecto ni sus riesgos, y no hay forma

de evaluar el cumplimiento de los planes. Un control efectivo permite detectar anticipadamente problemas en el cronograma, cuando aún hay tiempo suficiente para actuar sobre él.

e. Medición de Estadísticas

Una de las claves del progreso a largo plazo de una organización de software es la medición de datos para analizar la calidad del mismo y su productividad.

El procedimiento de calidad de desarrollo de software de una organización, debe definir qué mediciones realizar, con qué objetivo y la periodicidad, y cómo van a ser colectadas. Es usual disponer de un software que soporte la recolección automática o semiautomática de estas mediciones, y su uso de acuerdo a los fines para los que han sido definidas.

f. Gestión de Relaciones con los Clientes

El conocimiento y aplicación de buenas prácticas en las relaciones con clientes producen beneficios directos para el desarrollo de un software. Buenas relaciones con los clientes disminuyen el tiempo real y percibido de desarrollo, pues eliminan fuentes importantes de errores y riesgos para el proyecto, y propician una cooperación más activa y comprometida por parte de clientes y usuarios.

2.6.1.3. Procesos de Calidad

Debe asegurar que se optimiza la utilización del hardware y software en los productos desarrollados, que son económicos (se desarrollan con el menor costo posible), eficientes (sacan el máximo beneficio posible a los

recursos utilizados) y efectivos (alcanzan el resultado deseado con la menor cantidad posible de recursos, tiempo y esfuerzo).

Es importante destacar que los costos en los que se incurre para lograr la calidad en el software, son altos y, según Joseph Juran[21], el costo de asegurar la calidad crece de forma geométrica a medida que se pretende llegar a la perfección, por lo que no es recomendable tratar de llegar a la perfección, sino tratar de mantener los costos de aseguramiento de calidad mientras el costo de prevenir y detectar problemas sea menor que el costo de corregirlos.

a. Prácticas de prevención

Los estándares son uno de los medios más efectivos para garantizar la calidad del software. Prácticamente para cada producto a elaborar (manual de usuario, Interfaz, código, análisis de requisitos, entre otros.) o proceso a realizar (análisis de riesgos, diseño, planificación, etc.) deben existir estándares o normas organizacionales que definan directrices sobre cómo hacerlo. Los estándares tienen dos beneficios principales:

1. Evitan improvisaciones, olvidos y errores al definir qué hacer y cómo hacerlo.
2. Proponen una manera uniforme de hacer que facilitan comparaciones entre proyectos y colaboraciones entre equipos de trabajo diferentes.

Otro grupo de técnicas orientadas a prevenir errores y omisiones es el de métodos formales, que hace referencia a una variedad de técnicas de modelación matemáticas aplicables al diseño de sistemas informáticos.

[21] Joseph Juran: Precursor de la calidad en Japón. Se le considera el padre de la calidad. Se le reconoce como quien agrego recalco el aspecto humano en el campo de la calidad es de aquí donde surge los orígenes estadísticos de la Calidad Total.

Los métodos formales pueden ser usados para especificar y modelar el comportamiento de un sistema y para verificar matemáticamente que el diseño y la implementación del sistema satisfagan sus especificaciones. Estas técnicas pueden ser aplicadas prácticamente a todos los niveles del ciclo de vida del software.

Aplicar métodos formales solo a las partes más críticas de un sistema es una estrategia útil y muy efectiva. La verificación formal completa debe aplicarse únicamente en sistemas críticos que requieran la máxima fiabilidad.

b. Prácticas de detección y corrección

Para realizar la detección de errores se requiere de la elaboración de pruebas del sistema, de manera que permita encontrar fallas de calidad en un sistema.

Las actividades de pruebas pueden clasificarse en estáticas o dinámicas. Las técnicas de pruebas estáticas detectan errores sin ejecutar el programa. Las pruebas dinámicas implican la ejecución de programas. A su vez, las técnicas dinámicas pueden subdividirse en dos estrategias generales:

- **Pruebas de comportamiento:** (caja negra, basado en datos, entrada/ salida, basado en requisitos), en la que el grupo o persona a cargo es completamente ajeno al código fuente del programa, y está únicamente interesado en casos en los que el programa no se comporta como se espera.

- **Pruebas estructuradas:** (caja blanca, basado en lógica, basado en código) en la que el grupo o persona a cargo examina la estructura interna del programa con el objetivo de derivar casos de prueba.

c. Evaluación y mejora de proceso

La mejora de procesos de software (SPI, Software Process Improvement) se orienta a reducir costos y riesgos de los procesos, acortar el tiempo del proceso de desarrollo, y a incrementar la calidad del producto.

Los modelos de madurez de proceso de desarrollo de software, como SPICE o CMM, no han tratado adecuadamente el proceso de pruebas. Para solucionar estos inconvenientes, existen modelos especializados para medir la madurez y mejorar el proceso de pruebas como TIM (Test Improvement Model), TOM (Test Organisation Maturity Model), TPI (Test Process Improvement Model) y TMM (Testing Maturity Model).

d. Servicios de Calidad del Software

Persigue juzgar la efectividad del proceso de software de una organización e identificar aquellas áreas susceptibles de ser mejoradas y pretende ser la herramienta que describa el camino a seguir para incrementar gradualmente la madurez del proceso de software. Esta relacionado con el CMM (Capability Maturity Model).

El proceso de aseguramiento de la calidad va de la mano con el de control de calidad, ya que el grupo de aseguramiento de la calidad está encargado de definir los mecanismos de control de la calidad, sin ser este grupo el responsable de aplicar estos mecanismos. Mientras que el grupo de aseguramiento de la calidad es responsable de utilizar los resultados del

control de calidad para evaluar y mejorar el proceso de desarrollo, para conseguir productos de más calidad.

2.6.2. Tareas de la garantía de la calidad

Las principales tareas del grupo de garantía de calidad, por lo tanto, son:

2.6.2.1. Planificación de la calidad

Consiste en seleccionar, clasificar y ponderar las propiedades de calidad que se van a establecer como requisitos, con respecto al producto y con respecto al proceso. Se elegirán también los mecanismos de control de calidad a utilizar para medir y evaluar estas características y se determinarán las metas a alcanzar.

2.6.2.2. Supervisión de la calidad

Consiste en supervisar y corregir, si es necesario, el trabajo que se está realizando (según los resultados obtenidos con las actividades de control de calidad), con el objetivo de llegar a satisfacer los requisitos establecidos.

2.6.2.3. Construcción de la calidad

Actividades constructivas son aquellas que sirven para “construir” la calidad, es decir, son actividades preventivas cuyo objetivo es evitar la introducción de errores mediante la puesta en práctica de ciertos principios, métodos, formalismos y herramientas.

2.6.3. Establecimiento de un plan de SQA para un proyecto de software

El plan de garantía de calidad es elaborado en la etapa de planificación de la garantía de la calidad y este plan permite definir las actividades que

deben realizar el grupo de control de calidad y el de aseguramiento de la calidad a lo largo del proyecto de software. El plan de SQA identifica:

- Evaluaciones a realizar
- Auditorias y revisiones
- Estándares que se pueden aplicar al proyecto
- Procedimientos para información y seguimiento de errores
- Documentación producida por el grupo de SQA
- Realimentación de la información proporcionada al equipo del proyecto del software.

2.6.3.1. Actividades del proceso de SQA

SQA se define como un conjunto de actividades planificadas y sistematizadas, que permite evaluar la calidad y la adherencia de los problemas de software a los estándares, procesos y procedimientos. La conformidad con los estándares y procedimientos es evaluada a través del monitoreo de procesos, evaluación del producto.

El monitoreo de procesos y la evaluación del producto corresponde a las actividades del SQA responsable de verificar que el plan del proyecto, los procedimientos y estándares son seguidos correctamente durante el desarrollo y el control de los procesos.

La auditoria es una técnica, que analiza con detenimiento los procesos y productos sobre la base de su adherencia a los procedimientos y estándares. Su propósito es garantizar que se encuentra con un proceso de control adecuado, que la documentación mantenida y que los informes de los desarrolladores reflejen el estado de las actividades que

desempeñan. Y su producto un informe de guías y recomendaciones para los procesos de calidad de desarrollo dirigido a la administración superior.

Si bien estas tareas son las principales actividades SQA, son aun demasiado generales con relación a una implantación de esta área de prácticas.

Por ello es necesario establecer y definir las tareas que permiten materializar el monitoreo del proceso, la evaluación del producto y las auditorias al interior de una organización, que son:

1. **Estándares:** Los estándares son los cimientos de cualquier sistema de calidad de software, pues proveen la base para la evaluación y medición de las actividades y de los productos de trabajo durante el ciclo de vida del software. Por tanto ellos establecen el marco de trabajo para el desarrollo de software constituyéndose en un valor crítico de este último.

   Su aplicación otorga uniformidad, consistencia, rigurosidad y fortaleza a los métodos y a las actividades de desarrollo de software. Es mas los estándares entregan métodos y practicas comunes que permiten concretar una tarea repetidas veces en una misma forma.

   Las áreas contenidas en el estándar varían de una organización a otra según sus necesidades. Lo importante es no estandarizar todo. Un conjunto de estándares que cubra cada unos de los aspectos de las actividades organizacionales pierde la adhesión de sus usuarios sólo por su envergadura.

Por tanto, la estandarización puede ser aplicada a cualquier o a todas las áreas del desarrollo de software y mantenimiento. Sin embargo debería cubrir:

- Ciclo de vida del software
- Documentación
- Código Fuente
- Criterios para denominar los ítems de configuración
- Procedimientos y protocolos

2. **Revisiones:** Las revisiones constituyen la primera forma de monitorear y evaluar la calidad de los productos de trabajo, y además, proveen mayor visibilidad al desarrollo.

   Las revisiones son una metodología definida, estructurada y disciplinada para la detección e identificación de defectos en los productos de trabajo durante el ciclo de vida del software. Cuenta con sus etapas: planificación, orientación, preparación, inspección y seguimiento, las cuales son llevadas a cabo por un equipo con tareas y responsabilidades definidas, con documentación específica por un periodo determinado.

3. **Pruebas:** La prueba es la última actividad de evaluación del producto que permite detectar defectos y establecer el nivel de satisfacción de los requerimientos. Sus actividades incluyen la planificación, diseño, ejecución y reporte sobre los diferentes niveles de prueba existentes durante el proyecto.

   Esto incluye a los planes, especificaciones, procedimientos e informes. Por lo tanto, SQA debe garantizar que:

- Los procedimientos de prueba verifican los requerimientos según el plan
- La revisión del software evaluada sea la actual
- Los procedimientos sean utilizados
- Cualquier problema detectado durante esta actividad, sea registrado e informado oportunamente.
- Los informes entregados corresponden a la realidad y sean completos.
- La corrección de los errores sea realizada antes de la entrega del producto final.

4. **Análisis de defectos:** Los defectos ocurren a lo largo de todo del ciclo de vida del software sin excepción. Por ello resulta natural concentrar esfuerzos en su detección y corrección. No obstante a que la corrección de defectos es importante, más lo es su prevención. Esto solo puede alcanzarse a partir del registro y seguimiento de los defectos. Es entonces, el análisis de defectos, la actividad responsable de corregirlos y así disminuir los defectos en proyectos futuros.

   En términos generales, el análisis de defectos es el puente que une las tareas orientadas al control de la calidad (detectar defectos en los productos de trabajo) con aquellas orientadas al aseguramiento de la calidad (detectar debilidades en los procesos y procedimientos). Un análisis de defectos efectivo permite, a partir de la detección de fallas y de las mediciones del producto, identificar relaciones causa – efecto, las cuales van entregando pautas sobre posibles mejoras al proceso de desarrollo.

SQA debe responsabilizarse de crear un método de identificación de defectos de definiciones del proceso y de realizar los cambios necesarios para mejorar su eficacia y eficiencia.

5. Gestión de Configuración en SQA

El proceso de la Gestión de Configuración permite establecer y mantener la integridad de los productos a través de todo e ciclo de vida del software, además de mantener un control adecuado de los cambios producidos en los diversos puntos de la configuración. Para ello, cuenta con cuatro actividades principales: identificación de la configuración, control de cambios, contabilidad y auditorias de la configuración.

La identificación de la configuración proporciona un método único y específico para identificar cada instancia (release, versión, entre otros) de un producto de software. El control de cambios asegura que cada modificación sobre alguna instancia del producto sea conocida, autorizada y documentada.

Como actividad continua durante todo el ciclo de vida del software SQA es explicado en cada etapa del desarrollo con diferentes propósitos:

a) Planificación

Durante la etapa de planificación, SQA debe participar de la elaboración del plan de proyecto. Es su responsabilidad producir el Plan de SQA y verificar que los procesos, procedimientos y estándares identificados en el plan de proyectos son apropiados, claros, específicos y auditables.

El contenido del plan de SQA debe identificar, evaluaciones, auditorias y revisiones, estándares, procedimientos de seguimiento y reporte de errores, y la documentación por producir.

b) Especificación de requerimientos

SQA debe corroborar que en la especificación estén expresados todos los requerimientos funcionales, técnicos, operacionales y de interfaz, de manera tal que puedan ser verificados en el producto final.

c) Diseño

En la fase de diseño, dentro de las actividades de SQA se incluyen:

- La adherencia del diseño y su documentación a los estándares definidos en el plan de proyecto.
- La presencia de todo módulo en el diseño
- El ingreso del diseño a la configuración del software, tras su aprobación.

d) Implementación

A SQA le corresponde auditar:

- Los resultados de las actividades de diseño y codificación
- El estado de todos los entregables
- Las actividades de gestión de configuración y de la biblioteca del software
- Los informes sobre desviaciones y las correctivas

e) Integración y prueba

Con relación a la integración y a la prueba a SQA le corresponde garantizar la concordancia de las pruebas con el plan y los

procedimientos definidos, así como también que toda desviación haya sido informada y corregida. Además debe certificar que las actividades de prueba se han completado satisfactoriamente y que el software y su documentación se encuentran listos para la entrega del producto final.

En la fase de aceptación SQA es responsable de realizar la última auditoria de configuración del software, con el objetivo de determinar que los de liberables están listos par ala entrega.

f) Mantenimiento

Durante la operación pueden presentarse correcciones o mejoras que originen pequeños “ciclos de desarrollo”. En tal caso, se repetirán las actividades de SQA descrita con anterioridad.

2.7. Gestión de la Configuración del Software

A lo largo del ciclo de vida del software se produce información que forma parte del sistema construido, la misma que se denomina Configuración del Software, a medida que se avanza por las fases de la ingeniería de software se cuenta con mayor número de elementos de Configuración de Software (ECS).

Los ECS están siempre sujetos a cambio, lo cuales pueden producir confusión e incertidumbre, mas aun cuando no ha existido un correcto análisis.

La Gestión de la configuración del software “Es el arte de coordinar el desarrollo de software para minimizar la confusión. La gestión de configuración de software es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de

programación. La meta es maximizar la productividad minimizando los errores".

La gestión de la configuración del software es una actividad de auto protección que se aplica en el proceso del software. Como el cambio se puede producir en cualquier instante, las actividades del GCS sirven para:

- Identificar el cambio.
- Controlar el cambio.
- Garantizar que el cambio se implemente correctamente.
- Informar del cambio al equipo.

Es importante distinguir entre el mantenimiento del software y la gestión de configuración del software, puesto que el mantenimiento es un conjunto de actividades de la ingeniería del software que se producen después de que el software se ha entregado al cliente y este en funcionamiento, mientras que la gestión de configuración del software es un conjunto de actividades de seguimiento y control que comienza cuando se inicial el proyecto de ingeniería y termina cuando el software queda fuera e servicio.

2.7.1. Administración de la Configuración del Software

La gestión de la configuración del software requiere un grupo de trabajo que deberá adaptarse a políticas directrices y procedimientos que permitirán continuar con una adecuada administración.

2.7.1.1. Organización

El grupo de gestión de configuración del software debe contar con un personal que cumpla los siguientes roles:

1. **Líder de configuración**: Es el encargado de identificar los elementos de configuración.

2. **Miembros de control de cambios**: Es responsable de proveer o solicitar requerimientos de cambio.
3. **Desarrollador**: Es el responsable de revisar los cambios y resolver conflictos con los mismos
4. **Auditor**: Es el responsable de la selección y evaluación de las versiones para liberarlas y asegurar la consistencia y terminación del mismo.

2.7.1.2. Responsabilidades de Gestión de Configuración del Software

Cada uno de los roles especificados para la gestión de configuración del software deben cumplir con ciertas tareas asignadas a cada uno.

1. **Líder de configuración:**
   - Es el encargado del cambio
   - Preparación del cambio
   - Datos de los cambios
   - Instalación y mantenimientos de herramientas de administración
   - Preparación de informes
2. **Miembro de control de cambios:**
   - Aprobación del cambio
   - Registro del cambio
3. **Desarrollador:**
   - Crear versiones
   - Chequear los cambios
   - Resolución de conflictos
   - Desarrollador de soluciones

4. **Auditor:**

- Selección de versiones
- Aprobación de versiones

2.7.1.3. Políticas, directrices y procedimientos aplicables

1. Se desarrollara un plan continuo de trabajo
2. Desarrollo de estándares y aplicación de los mismos
3. La administración de configuración del proyecto debe seguir las políticas de la organización para la configuración del software.
4. Se deben organizar reuniones a la mitad y al finalizar el proyecto.
5. Se debe conservar todas las versiones pasadas y presentes de los elementos de configuración liberados.
6. Solo el líder del proyecto o en su ausencia el administrador del departamento pueden tener acceso a al archivo maestro.

2.7.2. Actividades De La Gestión De Configuración Del Software

2.7.2.1. Línea base

Línea base es un concepto de Gestión de Configuración del Software que ayuda a controlar los cambios sin impedir seriamente los cambios justificados, consiste en una especificación o producto que se ha revisado formalmente y sobre el que se ha llegado a un acuerdo, estableciendo una base para un desarrollos posteriores que puede ser cambiado solamente a través de procedimientos formales de control de cambios.

En realidad, cada especificación o elemento objeto de cambio (ECS) [22]se organizan como objetos de configuración que han de ser catalogados en la base de datos del proyecto con un nombre único. Un objeto de

[22] Cada elemento que conforma un cambio en un producto se conoce como Elemento de Configuración de Software (ECS).

configuración tiene un nombre y unos atributos. De acuerdo a los objetos de configuración especificación de diseño, modelado de datos, componentes n, código fuente y especificación de pruebas, están definidos por separado.

2.7.2.2. Actividades de la Gestión de Configuración del Software

La gestión de configuración del software cuenta con un conjunto de actividades que permiten evitar confusiones respecto a versiones y cambios, permitiendo además llevar un control histórico de los mismos.

1. **Identificación de Objetos en la Configuración del Software**: Para controlar y gestionar los Elementos de Configuración de Software, se debe identificar cada uno de forma única y organizarlos mediante un enfoque orientado a objetos. Se puede identificar dos tipos de objetos:
    a. **Objetos compuestos**: Es una colección de objetos básicos y de otros objetos compuestos
    b. **Objetos Básicos:** Es una unidad de ECS creado por un ingeniero de software durante el análisis, diseño, codificación o pruebas.
2. **Control de versiones:** El control de versiones combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso del software.

    La gestión de configuración permite a un usuario especificar configuraciones alternativas del sistema de software mediante la selección de las versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego

especificar una configuración describiendo el conjunto de atributos deseado.

Los atributos que se menciona pueden ser tan sencillos como un número específico de versión asociado a cada objeto o tan complejos como una cadena de variables lógicas que especifiquen tipos de cambios funcionales aplicados al sistema.

3. **Control de configuración:** Se encarga de mantener un control de las versiones y los cambios efectuados.
4. **Control de versiones:** Combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso de software.

   Se puede utilizar un grafo de evolución de versiones en el que se representa la manera en la que una versión varía por efecto de atributos distintos a cada uno de ellas.
5. **Control de cambios:** Consiste en mantener o en revisar cada pequeño cambio en los elementos de configuración de software para evitar que estos produzcan un fallo mayor en el sistema final.

   Las actividades que se deben realizar para efectuar un cambio son básicamente:

   - Reconocer la necesidad del cambio
   - Autorización de cambio
   - Realización de cambio
   - Pruebas del cambio

2.7.3. Registro de estado de configuración

El líder de configuración debe actualizar el resumen de configuración continuamente, este resumen debe entregar información respecto a los siguientes puntos:

- Actividades realizadas
- Responsables
- Fecha
- Consecuencias

2.7.4. Auditorias y revisiones de configuración

Es necesario realizar auditorias aleatorias, que deben ser programadas por el administrador de proyectos de forma que el líder de configuración realice revisiones continuas. El líder de gestión de configuración debe revisar el estado de la gestión de configuración e informar los procedimientos detallados propuestos a seguir en el código y en la integración.

2.8. Métricas Técnicas del Software

Las métricas para el software se han propuesto como una valoración cuantitativa de los procesos involucrados en el desarrollo de software. Estas métricas permiten valorar también el grado de calidad con el que cuenta el software, convirtiéndose en una herramienta del ingeniero de software para detectar y corregir problemas “sobre la marcha”.

Es importante definir ciertos términos tales como[23]:

- **Medida:** proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto.

[23] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo IV, Pág. 54.

- **Medición:** es el acto de determinar medidas
- **Métrica:** Medida cuantitativa del grado en que un sistema componente o proceso posee un atributo dado.
- **Indicador:** es una métrica o una combinación de métrica que proporciona una visión profunda del proceso del software.

A pesar que en la actualidad los desarrolladores de software y los directores de proyectos de software no dan importancia a la medición de los productos de software, hay que considerar que en los últimos años se han obtenido grandes avances dentro de este campo y con estos avances se ha logrado que mayor cantidad de personas se inclinen más hacia su utilización.

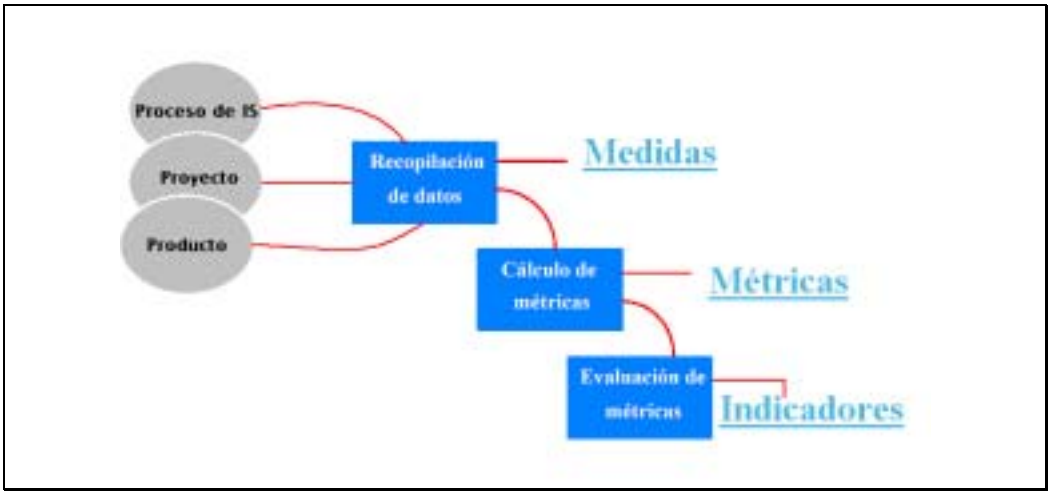


Fig.2.6 Medición y Métricas

El proceso de medición de la fases de la ingeniería de software es importante debido a que da una pauta al desarrollador sobre la manera en la que se mejora en el desarrollo o si es que este mejoramiento existe.

Si el proceso de desarrollo del software puede ser mejorado esto representara avances significativos en el producto final, por tanto la medición

se utiliza para establecer una línea base[24] del proceso desde donde se pueden evaluar las mejoras.

Una vez terminado el Diseño es posible establecer ciertas preguntas:

- ¿Qué requisitos del usuario son más susceptibles al cambio?
- ¿Qué módulos del sistema son más propensos al error?
- ¿Cuántos errores pueden esperar cuando comience la prueba?

Al aplicar las preguntas mencionadas anteriormente se debe recopilar los resultados que serán la base para el desarrollo de la guía técnica que se presentará en el Capitulo III.

2.8.1. Métricas clásicas del software

Las métricas clásicas del software son aquellas que se utilizan en los métodos orientados al proceso y funciones. Estas métricas están divididas por cada etapa de desarrollo del software.

2.8.1.1. Métricas del modelo de análisis

Debido a que en la fase de análisis es en la que se define requisitos y se establece fundamentos para el diseño, e importante contar con un conjunto de métricas que permitan evaluar la calidad del modelo de análisis utilizado.

Existen algunas métricas que pueden ser utilizadas para evaluar la calidad del proceso de análisis tales como:

- Métricas basadas en puntos función
- Métrica de Bang
- Métricas de la calidad de la especificación

[24] Línea Base: Una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado un acuerdo, y que de ahí en adelante sirve como ase para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambio (IEEE-612)

2.8.1.1.1. Métricas basadas en puntos de función.

Puede ser utilizada como una herramienta para predecir el tamaño de un sistema que se va a obtener del modelo de análisis. Para el cálculo de puntos de función existen los siguientes puntos:

i. Número de entradas de usuario
ii. Número de salidas de usuario
iii. Número de consulta de usuario
iv. Número de archivos
v. Número de interfaces externas.

$$PF = cuenta - total * (0.65 + 0.01)*\sum[Fi]$$

Ecuación 2.1

Donde cuenta-total es la suma de todas las entradas de Puntos de Función obtenidas (Tabla 2.6) y Fi (i=1 a 14) son los valores de ajuste de complejidad[25].

Tabla 2.6. Factores de Punto de Función

Parámetros	Cuenta	Factor de ponderación			Subtotal
		Simple	Medio	Complejo	
Número de entradas de usuario		3	4	6	
Número de salidas de usuario		4	5	7	
Número de peticiones de usuario		3	4	6	
Número de archivos		7	10	15	
Número de Interfases externas		5	7	10	
Total					

[25] Arthur. L. J., Measuring Programmer Productivity and Software Quality, Wile-Interscience, 1985.

Este factor de ajuste de complejidad se determina entregando valores entre 0 (siendo el menos significativo) y 5 (significativo) a las siguientes preguntas:

1. Requiere el sistema copias de seguridad y de recuperación fiables.
2. Se requiere comunicación de datos
3. Existen funciones de procesamiento distribuido
4. Es critico el rendimiento
5. Se ejecutara el sistema e un entorno operativo existente y fuertemente utilizado
6. Requiere el sistema de datos entrada de datos interactiva
7. Requiere la entrada de datos interactiva que las transacciones de datos se lleven a cabo sobre múltiples pantallas y operaciones
8. Se actualiza los archivos maestros en forma interactiva.
9. Son complejas las entradas, salidas los archivos o las peticiones
10. Es complejo el procesamiento interno
11. Se ha diseñado el código para ser reutilizado
12. Están incluidas en el diseño la conversión y la instalación
13. Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones
14. Se ha diseñado la aplicación para facilitar los cambios y para hacer fácilmente utilizada por el usuario.

Para la determinación del tamaño de un sistema, se propone encontrar la relación Puntos de Función y el esfuerzo utilizando como nexo las líneas de código (Tabla 2.7.), es decir, calcular los Puntos de Función para estimar el número de líneas de código y usar este resultado para estimar el esfuerzo.

La Tabla 2.7. proporciona estimaciones del número medio de líneas de código requerido para construir un punto de función en diferentes lenguajes y niveles de programación.

La estimación de recursos necesarios para un proyecto de software se puede determinar utilizando el total de número de líneas de código determinadas en la tabla anterior (Tabla 2.7.).

Tabla 2.7. Líneas de Código / Puntos de Función

RELACION / LENGUAJE	NIVEL	LDC / PF
C	2.5	128
Decision Support Default	9	36
Ansi Basic	5	64
C++	6	53
Java	6	53
SQL	25	13
PL/I	4	80
4th Generation Default	16	20
Ansi Cobol 74	3	107
Visual 4.0	11.00	29
Visual Basic 1	7.00	46
Visual Basic 2	7.50	43
Visual Basic 3	8.00	40
Visual Basic 4	9.00	36
Visual Basic 5	11.00	29
Visual Basic DOS	8.00	40
Visual C++	9.50	34
Basic Assembly	1	320

Dentro del alcance del software las métricas del punto de función ayudan a determinar la eficacia y eficiencia que puede llegar a tener un producto. Es así que, basado en un análisis de lo

requerido vs. lo obtenido, se logra definir que tan acertado fue el análisis de un proyecto de software o en su defecto cuales fueron las debilidades del proceso de análisis.

Tabla 2.8. Categoría de un proyecto en función de sus líneas de código

Categoría	Personas	Duración	Líneas de código
Trivial	1	0 – 4 semanas	< 1k
Pequeño	1	1 – 6 meses	1k – 3k
Media	2 – 5	0,5 – 2 años	3k – 50k
Grande	5 – 20	2 – 3 años	50k – 100k
Muy grande	100 – 1000	4 – 5 años	100k – 1M
Gigante	1000-5000	5 – 10 años	> 1M

El usuario juega un papel importante en el proceso de levantamiento de información (Fig. 2.7) por cuanto él domina el ámbito que será automatizado previo a un exhaustivo análisis que permita establecer la factibilidad de automatización.

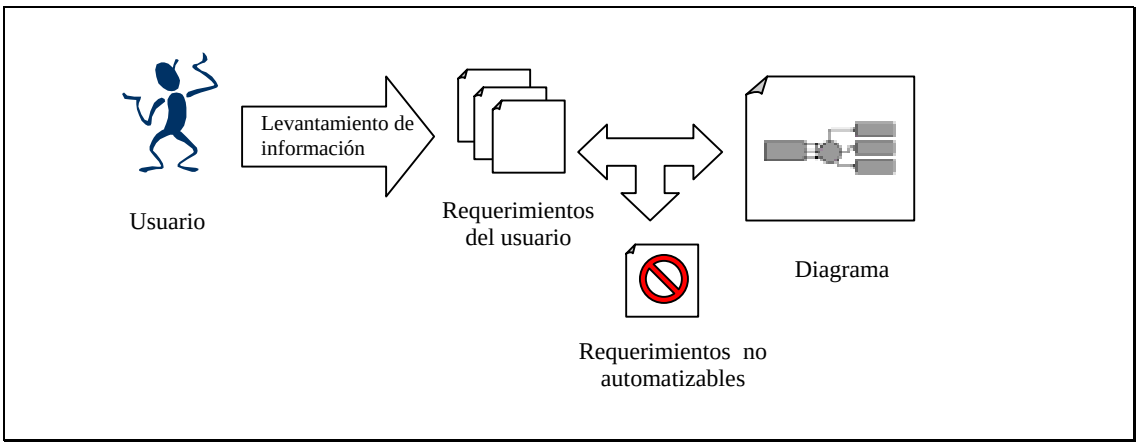


Fig. 2.7 Proceso de Especificación de Requerimientos

Es claro, entonces, suponer que el esfuerzo que se aplica al análisis de las necesidades es proporcional al éxito con que se obtiene la solución para dicha necesidad.

2.8.1.1.2. Métricas Bang

Al igual que la métrica de punto de función, la métrica Bang puede emplearse para desarrollar una identificación del tamaño del software a implementar como consecuencia del modelo de análisis. Las primitivas se determinan evaluando el modelo de análisis por medio del desarrollo de conteos para los siguientes elementos:

- Primitivas funcionales (PFu)
- Elementos de datos (ED)
- Objetos (OB)
- Relaciones (RE)
- Estados (ES)
- Transiciones (TR)

Es claro que esta métrica se enfoca de manera aparentemente híbrida al dominio de funciones y datos, por lo que se puede determinar cual es el dominio en el que cae aplicando la relación del Cuadro 2.1 propuesta por De Marco[26].

Cuadro 2.1. Dominio de la aplicación propuesta por De Marco

RE/Pfu < 0,7	Dominio de Función
0, 8 <= RE/Pfu <= 1,4	Híbrido
RE <=1,5	Dominio de Datos

2.8.1.1.3. Métricas de la calidad de la especificación

Proponen una lista de características que pueden emplearse para valorar la calidad del modelo de análisis y la correspondiente especificación de requisitos: especificidad (ausencia de

[26] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 330

ambigüedad), compleción, corrección, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización.

Debido a la gran cantidad de requisitos posibles, se asume una medición cuantitativa, definiendo un valor de n_r requisitos funcionales de una especificación de requerimientos tal como se ilustra en la Ecuación 2.2.

$$n_r = n_f + n_{nf}$$

Ecuación 2.2

Donde n_f es el número de requisitos funcionales y n_{nf} es el número de requisitos no funcionales.

Para determinar la especificidad de los requisitos, existe la métrica basada en la consistencia de la interpretación de los revisores para cada requisito, por medio de la utilización de la siguiente ecuación:

$$Q_1 = n_{ui} / n_r$$

Ecuación 2.3.

Donde n_{ui} es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. Cuanto mas cerca esté Q de 1, menor será la ambigüedad [27]de la especificación.

La compleción[28] de los requisitos funcionales puede determinarse calculando la siguiente relación:

[27] Ambiguo: Que puede entenderse de varios modos o admitir distintas interpretaciones y dar por consiguiente motivo a duda, incertidumbre o confusión., Diccionario Enciclopédico Siglo XXI.

$$Q_2 = u_n / (n_i \times n_s)$$

Ecuación 2.4.

Donde U_n es el número de requisitos únicos de función, n_i es el número de entradas (estímulos) definidos o implicados por la especificación y n_s es el número de estados especificados.

La relación Q_2 mide el porcentaje de funciones necesarias que se han especificado para un sistema. Esta métrica no evalúa los requisitos no funcionales, para lo cual se requiere utilizar la siguiente relación:

$$Q_3 = n_c / (n_c + n_{nv})$$

Ecuación 2.5.

Donde n_c es el número de requisitos que se han validado como correctos y n_{nv} el número de requisitos que no se han validado.

2.8.1.2. Métricas del modelo de diseño

Estas métricas son las menos utilizadas en el proceso de desarrollo de software, pero no por ello dejan de ser importantes. A pesar de existir pocas métricas probadas para esta etapa, se pueden citar las siguientes:

[28] Complecion: Indicador de que una especificación de requerimientos esta completa, es decir que se incluya todas las características del comportamiento del software que se va a construir y que se especifique todas las entradas que debe soportar y las salidas que debe entregar.

2.8.1.2.1. Métricas de diseño arquitectónico

Estas métricas se concentran en las características de la arquitectura del programa y la eficiencia de sus módulos. Los autores Glass y Card[29] definen tres medidas para el diseño arquitectónico.

1. **La complejidad estructural, S(i),** de un módulo i se define de la siguiente manera:

$$S(i) = f^2_{out}(i)$$

Ecuación 2.6

Donde $f_{out}(i)$ es la expansión del módulo i; es decir, el número de módulos invocados directamente por el módulo i.

Mientras el valor de S(i) aumenta, aumenta la complejidad del diseño del sistema, concluyendo en mayor complejidad del sistema.

2. **La complejidad de datos, D(i)** proporciona una indicación de la complejidad en la interfaz interna de un módulo i; representa la cantidad de información procesada por el módulo y es proporcional a la complejidad del sistema. Se define como:

$$D(i) = v(i)/[fout(i)+1]$$

Ecuación 2.7

Donde v(i) es el número de variables de entrada y salida del módulo.

[29] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 332

3. **La complejidad del sistema** Se define como la suma de las complejidades estructural y de datos.

$$C(i) = S(i) + D(i)$$

Ecuación 2.8

En general, a medida que aumentan los valores de complejidad, la complejidad arquitectónica o global del sistema también aumenta, lo que indica una mayor probabilidad de que aumente el esfuerzo para la integración y las pruebas.

El esfuerzo requerido para las actividades de integración y pruebas es un indicativo de calidad en la medida en que un software de calidad pretende facilitar las dichas actividades; Un gran esfuerzo en la integración y pruebas implica directamente una deficiencia en la calidad global del producto de software.

Existe otra métrica de diseño arquitectónico de alto nivel propuesta por Henry y Kafura [30] utiliza la expansión y la concentración.

$$MHK = longitud(i) \times \left[f_{in}(i) + f_{out}(i) \right]^2$$

Ecuación 2.9

Donde la longitud *(i)* es el número de sentencias en lenguaje de programación en el módulo *(i)* y es la concentración[31] del módulo i.

[30] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 333

2.8.1.2.2. Métricas de diseño a nivel de componentes

Se concentran en las características internas de los componentes e incluyen las medidas de la cohesión, acoplamiento y complejidad del módulo estas tres medidas pueden ayudar al desarrollador del software a determinar la calidad a nivel de los componentes. Estas métricas se pueden aplicar una vez que se ha desarrollado un diseño procedimental e incluso pueden ser desarrolladas en el momento de contar con el código fuente.

1. **Métricas de cohesión:** Se utilizan cinco conceptos y medidas: porciones de datos, símbolos léxicos de datos, señales de unión, señales de superunión, pegajosidad[32].

2. **Métricas de acoplamiento:** El acoplamiento de módulo proporciona un indicador de la conectividad de un módulo con otros módulos, datos globales y el entorno exterior.

   Según Dhama[33], existe una métrica que combina el acoplamiento de flujo de datos y de control, acoplamiento global y acoplamiento de entorno.

   Las medidas utilizadas son las siguientes:

   a. **Acoplamiento de Flujo de datos y de control**

      De= número de parámetros de datos de entrada

      Ce= número de parámetros de control de entrada

[31] Concentración: Dado un módulo i, se define concentración como el número de llamadas a dicho módulo.
[32] Pegajosidad: Es una proporción del número de porciones de datos con los que un módulo se relaciona.
[33] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 334

Ds= número de parámetros de datos de salida

Cs= número de parámetros de control de salida

b. Acoplamiento global

Gd= número de variables globales usadas como datos

Gc= número de variables globales usadas como control

c. Acoplamiento de entorno

W = número de módulos llamados (expansión)

R = número de módulos que llaman al módulo en cuestión (concentración)

Con estas medidas, se tiene el indicador de acoplamiento de módulo, así:

$$m_c = \frac{k}{M}$$

Ecuación 2.10

Donde:

$$M = De + a \times Ce + Ds + b \times Cs + Gd + c \times Gc + w + r$$

K = 1, a = b = c = 2. Estos valores pueden ajustarse a medida que se van verificando estos datos experimentalmente.

Mientras mayor es el valor de m_c , menor es el acoplamiento del módulo, por lo que se limita la portabilidad del módulo y se dificulta las tareas de mantenimiento del mismo.

2.8.1.2.3. Métricas de Complejidad

Se utilizan para determinar la complejidad del flujo de control del programa. Según Thomas McCabe pueden emplearse para predecir la información crítica sobre la fiabilidad y mantenimiento de sistemas de análisis automático de código fuente. En la etapa de pruebas y mantenimiento se ayuda a resaltar áreas de inestabilidad potencial. La métrica propuesta por McCabe es la complejidad ciclomática[34]

2.8.1.2.4. Métricas de diseño de interfase

Sugiere la conveniencia de la representación (CR) como una valiosa métrica de diseño para interfaces hombre – maquina. Una GUI (Interfaz Grafica de Usuario) típica usa entidades de representación para ayudara al usuario a completar tareas.

Para una representación específica se puede asignar costos a cada secuencia de acciones de acuerdo a la siguiente relación:

$$\text{Costos} = \sum [\text{frecuencia de transición (k)} * \text{costo de transición (k)}]$$

Ecuación 2.11

Donde k es transición especifica de una entidad de representación a la siguiente cuando se realiza una tarea. Esta suma se da con todas las transiciones de una tarea en particular o un conjunto de tareas requeridas para conseguir alguna función de la aplicación. La conveniencia de la representación se define como:

[34] Complejidad Cilomatica: Métrica de software que proporciona una medición cuantitativa de la complejidad lógica del programa.

CR = 100 * [(costo de la representación óptima CR) / (costo de la representación propuesta)]

Ecuación 2.12

Donde CR es = 100 para una representación óptima.

2.8.1.3. Métricas del Código Fuente

Halstead propone las primeras leyes analíticas para el software, estas leyes se obtienen a partir de un conjunto de medidas primitivas que resultan de la estimación o generación del código[35].

El entendimiento de estas medidas primitivas se halla sujeto a dos definiciones:

- **Operadores:** Conjunto de las construcciones de flujo de control y operaciones matemáticas.
- **Operandos:** Conjunto de todas las variables y constantes del programa.[36]

Las medidas primitivas se listan a continuación:

- n_1: número de operadores diferentes que aparecen en el programa.
- n_2: número de operandos diferentes que aparecen en el programa.
- N_1: número total de veces que aparece el operador.
- N_2: número total de veces que aparece el operando.

Las medidas propuestas por Halstead tienen por objetivo determinar la longitud y el volumen de un programa. Las fórmulas propuestas por el autor son:

[35] Martin, James. SOFTWARE MAINTENANCE, THE PROBLEM AND ITS SOLUTIONS.1983. Part II, Pág.53-SS
[36] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 336.

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

Ecuación 2.13

Donde N es la longitud del programa

$$V = N \log_2 (n_1 + n_2)$$

Ecuación 2.14

Donde V es el volumen de información en bits.

Debe existir un volumen minino para el algoritmo (L), como se muestra en la ecuación 2.15, definido como una relación entre el volumen mínimo y el real de un programa; por tanto L debe ser menor a 1.

$$L = 2/ n_1 * n_2 / N_2$$

Ecuación 2.15

Al existir un volumen demasiado alto, se corre el riesgo de ganar dispersión [37] de los elementos del sistema, lo que ocasionaría un incremento en el esfuerzo necesario para expansión, acoplamiento y mantenimiento del sistema.

2.8.1.4. Métricas en las pruebas

En general las métricas en las pruebas, deben estar basadas en las aplicadas en el análisis, diseño y código de los programas, para ser comprobados en los casos de prueba.

Así las métricas basadas en la función pueden emplearse para predecir el esfuerzo global de las pruebas. Se gana mucho, el basarse en proyectos anteriores, tomando en cuenta los errores encontrados, esfuerzo y tiempo

[37] Dispersión: Dado un módulo i, se define dispersión como el número de llamadas que dicho módulo realiza a otros módulos.

para realizar esas pruebas, de esta manera se puede predecir de estas características, que tan bien estructurado esta el proyecto actual.

Otra forma de examinar las pruebas es hacerlo en la interfaz hombre-máquina del sistema desarrollado, tomando en cuenta ciertas características como:

- Número de objetos de datos, que se mueven a través de la interfaz.
- Número de datos que se introducen en cada interfaz.
- Número de transiciones contenidas en la interfaz hombre-máquina.

Cuando se desea obtener información acerca de la facilidad o dificultad para realizar las pruebas de integración y la necesidad de software especializado de prueba, se debe utilizar las métricas de diseño de alto nivel.

Para calcular el porcentaje del esfuerzo de cada módulo, se debe conocer primero el nivel del programa, que está dado por la siguiente relación:

$$\text{Nivel de programa} = 1/\ (\ n_1/2)*(\ N_2/\ n_2\ )$$

Ecuación 2.16

Donde:

n_1: Es el número de operadores no repetidos presentes en el programa.

n_2: Es el número de operando no repetidos presentes en el programa.

N_2: Es el número de veces que aparece el operando.

Una vez encontrado el nivel de programa, se tiene el valor del volumen (o se puede calcular con la Ecuación 2.13), se tiene la siguiente expresión:

$$\text{Esfuerzo} = \text{volumen} / \text{nivel de programa}$$

Ecuación 2.17

Porcentaje de esfuerzo de pruebas (k) = e (k) / e (i)

Ecuación 2.18

Donde:

- k es el módulo
- e (k) es el esfuerzo de cada módulo
- e(i) es la suma del esfuerzo total

2.8.1.5. Métricas de mantenimiento

Dentro de los costos de mantenimiento se puede tener costos tangibles, es decir aquellos que representan una suma de dinero; y costos intangibles, es decir aquellos que tienen que ver con la insatisfacción del cliente ante una reparación o modificación no satisfecha a tiempo, disminución de la calidad del Software ante errores que hay que solucionar, modificación de tareas del equipo de trabajo.

El costo de mantenimiento incide en la reducción de la productividad, sobre todo cuanto se inicia el mantenimiento en programas antiguos; medida la productividad en líneas de código por persona - mes, o en puntos de función por persona - mes.

Para realizar el mantenimiento se podrían presentar los siguientes inconvenientes:

- Dificultad en seguir la evolución del Software por mala o inadecuada documentación.
- No se encuentra un proceso bien delimitado del ciclo de desarrollo.

- Dificultad en entender un programa desarrollado, código sin una debida información adicional del proceso que desarrolla cada línea o módulo de código.
- La persona desarrolladora de la versión inicial no está presente en los cambios del Software.
- Ciclo de desarrollo de Software no documentado o documentado en forma parcial.
- Software que no permite cambios.
- Mantenimiento sea considerado como una actividad innecesaria.

Las métricas de mantenimiento de Software suelen referirse a características del Software que más frecuentemente afectan a este y en modelos de predicción de la cantidad de trabajos de mantenimiento. Basados en el modelo COCOMO se ha establecido una relación para predecir el número de personas-mes dedicadas anualmente al mantenimiento, pero antes se debe calcular el Tráfico de Cambios Anual (TCA):

$$TCA = KLDC \text{ para un sistema en mantenimiento} / \ IC$$

Ecuación 2.19

Donde IC es el número de instrucciones de código fuente que se modifican o añaden durante un año de mantenimiento.

Conocido el valor de TCA, se tiene la siguiente expresión, para predecir el número de personas-mes necesarias para el mantenimiento:

$$E.mant = TCA * 2{,}4 * KLDC^{1.05}$$

Ecuación 2.20

Donde:

E.mant es número de personas / mes

TCA es Tráfico de Cambios Anual

KLDC Kilo Líneas de Código

Es complejo evaluar una actividad de mantenimiento debido a que no se registran datos importantes. Swanson presenta una lista resumida de medidas potenciales:

1. Número medio de fallos de procesamiento.
2. Total de personas / hora por cada categoría de mantenimiento.
3. Número medio de cambios por programa, por lenguaje y por tipo de mantenimiento.
4. Número medio de personas / hora, por línea añadida o eliminada.
5. Número medio de personas / hora por lenguaje.
6. Tiempo total medio por formulario de petición de mantenimiento.
7. Porcentaje de petición de mantenimiento por tipo.

Estas siete medidas ayudan a tomar una decisión sobre la técnica de desarrollo de Software, selección de lenguaje de previsión de esfuerzo de mantenimiento, etc.

La capacidad de mantenimiento es el grado en que un sistema de software permite un cambio para aumentar su vida útil.

Existen dos puntos de vista de la capacidad de mantenimiento:

- Interno: Depende únicamente de las características del sistema de software.
- Externo: Depende también de la persona que hace el cambio, la documentación de apoyo y las herramientas.

2.8.2. Métricas Orientadas Objetos

El paradigma de Orientación a Objetos posee varias características como son el encapsulamiento, herencia, ocultamiento entre otras, que lo diferencia notablemente de los paradigmas tradicionales por lo que las métricas orientadas a datos o funciones pueden no ser útiles en Orientación a Objetos.

Se propone por tanto un conjunto de métricas que pretenden complementar, en ciertos casos, o definir, en otros, medidas para lograr evaluar un software realizado bajo el paradigma de Orientación a Objetos.

Bernard define cinco características que regulan las métricas orientadas a objetos:

- **Localización:** Es un indicador de la forma en la que se concentra la información en un programa en orientación a objetos esta concentración se origina a partir de la encapsulación de datos de datos y procesos de clases u objetos.
- **Encapsulación:** Engloba las responsabilidades de una clase, incluyendo sus atributos, operaciones y los estados de la clase dentro de atributos.
- **Ocultación de información:** Consiste en la supresión de detalles operacionales de un componente de un programa.
- **Herencia:** Cuando mas profunda se encuentra una clase en la jerarquía, mayor es la probabilidad de heredar mas métodos, lo cual conduce a que exista mayor cantidad de métodos heredados en una clase y conduciendo a una mayor complejidad de la misma.

- **Abstracción:** Es un mecanismo que permite al diseñador concentrase en los detalles esenciales de un componente de programa prestando poca atención a los detalles de bajo nivel[38].

Dentro de las métricas Orientadas a Objetos, una característica fundamental es como se puede caracterizar a un sistema orientado a objetos para que implemente efectivamente los requerimientos del cliente, cabe aclarar que cuando un programa crece en tamaño y en complejidad se debe tener una visión mas objetiva.

Para lo cual de debe tomar en cuenta los puntos que se describen a continuación:

- Descripción de los objetos en términos de sus propiedades y de sus relaciones.
- Idea básica: describir un grupo de objetos similares en términos de clases, que son instanciadas para crear objetos individuales.

Según Whitmire[39] se debe tomar en cuenta ciertas características que pueden medirse para evaluar un diseño orientado a objetos:

- **Tamaño*:*** Se define en cuanto a instancias: población, volumen, longitud, funcionalidad.
- **La población:** Se mide realizando un recuento de las entidades OO, clases y operaciones.
- **Longitud*:*** Es la medida de un cadena de elementos de diseño interconectados (eje. árbol de herencia).

[38] GRADY, Booch. ANALISIS Y DISEÑO ORIENTADO A OBJETOS CON APLICACIONES, 1996, Pag. 47.
[39] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XXIV, Pág. 423

Las métricas de funcionalidad proporcionan una indicación indirecta del valor agregado que se entrega al cliente dentro de una aplicación OO.

- **Complejidad:** Es la interrelación entre las clases de un diseño orientado a objetos.
- **Acoplamiento:** Es una medida de interconexión entre objetos, indica la medida en que una clase utiliza métodos o atributos de otra, un bajo acoplamiento refleja una limitada interdependencia de las clases.
- **Suficiencia:** Es el grado de que un componente de diseño posee características de abstracción.
- **Integridad:** Evalúa la capacidad que un objeto puede ser reutilizado.
- **Cohesión:** Un componente OO debe ser diseñado de manera que posea todas las operaciones trabajando conjuntamente para alcanzar un propósito único y bien definido.
- **Originalidad:** Esta orientado tanto a las operaciones como a las clases; es como el grado de una operación no puede ser construida fuera de una secuencia de otras operaciones contenidas dentro de una clase.
- **Similitud:** Indica el grado en que dos o mas clases son similares en términos de estructuras.
- **Volatilidad:** Es un componente OO que mide la probabilidad de que un cambio ocurra.

2.8.2.1. Métricas de análisis Orientadas a Objetos

Entre las actividades que debe realizar el jefe de proyecto en la etapa de análisis esta la planificación y una de las primeras actividades de planificación es la estimación. Dentro del modelo evolutivo de los

procesos, la planificación debe ser revisada en cada interacción del software

Uno de los aspectos que el jefe de proyecto debe tomar en cuenta es el tamaño de implementación del software, ya que el tamaño es directamente proporcional al esfuerzo y a la duración.

2.8.2.1.1. Número de escenario (NE)

El número de escenarios o caso de uso es directamente proporcional al número de clases requeridas para cumplir los requisitos. El número de escenarios (NE) es un indicador del tamaño de un programa.

2.8.2.1.2. Número de clases clave (NCC)

Una clase clave es la representación del dominio del negocio por lo que es difícil de ser reutilizada. Por esta razón, valores altos para el NCC indican un gran esfuerzo al momento de desarrollo.

Lorenz y Kidd sugieren que entre el 20 y 40 por ciento de las clases de un sistema Orientado a Objetos son claves (Clases Base) el resto son consideradas de soporte (GUI).

Es así que se estima que por cada Clase Base[40] se requerirá de 10 a 15 días por una persona[41], adicionalmente se requiere de un factor que representa el tipo de GUI que será utilizado por cada clase (Tabla 2.28).

[40] Clase que engloban el dominio del negocio y que por tanto son difíciles de reutilizar.
[41] La cantidad de días requeridos dependerá de la experiencia del grupo de desarrollo.

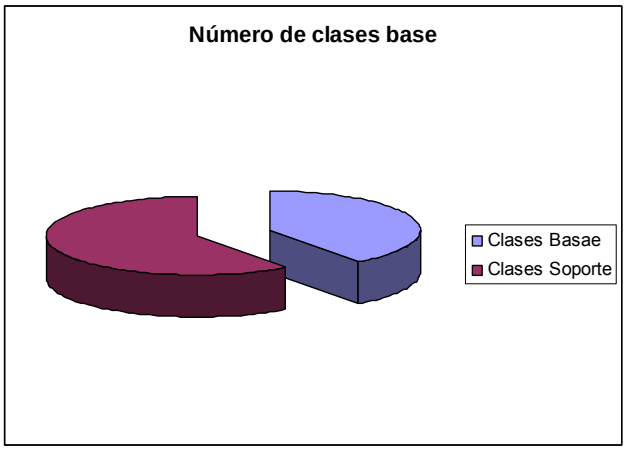


Fig. No. 2.8. Proporción de las clases base

$$\text{Tiempo requerido para Clases base = TRCB = } 15 * \sum (n * f)$$

Ecuación 2.21

Donde n es el número de clases que utilizarán una determinada GUI y f es el factor de dicha GUI.

Por otra parte, para calcular el tiempo necesario para la elaboración de las clases soportes se indica que se requiere 5 días y los valores del factor correspondiente al GUI utilizado.

$$\text{Tiempo requerido para Clases soporte = TRCS = } 5 * \sum (\mathrm{n} * \mathrm{f})$$

Ecuación 2.22

Donde en es el número de Clases Soporte[42] que se utilizarán con determinada GUI y f es el factor de dicha GUI (Tabla 2.28).

[42] Clases destinadas a complementar el funcionamiento de las clases base.

Tabla 2.9. Tabla valores por tipo de interfaz Lorenz y Kidd

Tipo de Interfaz	Multiplicador
Interfaz no gráfica (no IGU)	2,0
Interfaz de usuario basada en texto	2,25
Interfaz gráfica de usuario (IGU)	2,5
Interfaz gráfica de usuario compleja	3,0

Para la determinación del tiempo total en días por una persona, basta sumar los tiempos estimados de construcción de las clases base y las clases soporte.

$$\text{Total de días} = TD = TRCB + TRCS$$

Ecuación 2.23

La determinación del tiempo total de duración del sistema, se requiere dividir el número total de días para el número de personas que intervendrán en el desarrollo del mismo (Ecuación 2.23).

$$\text{Total días del proyecto} = TDP = TD / \text{Total de Personas}$$

Ecuación 2.24

El ingeniero en sistemas se encargará de coordinar y gestionar los aspectos relacionados con la administración de los recursos y las presentaciones parciales del sistema. Los costos indirectos son los relacionados con la depreciación de las maquinas utilizadas durante la realización del sistema. Los costos directos son los relacionados con los servicios básicos requeridos para el desarrollo del sistema.

Y los suministros finalmente son los suministros de oficina que se requieren para la realización del sistema.

2.8.2.1.3. Número de subsistemas (NSUB)

Proporciona una idea general de la asignación de recursos, de la planificación y del esfuerzo global de la integración.

2.8.2.2.1. Colección de Métricas de Diseño Orientado a Objetos (MDDO)

Esta colección consta de un grupo de indicadores que permiten medir la de forma cuantitativa el diseño orientado a Objetos y fueron propuestas por Harrison, Consseil y Nithi[43].

1. Factor de acoplamiento

Este factor es una relación entre el número máximo de acoplamientos del sistema y el número de acoplamientos reales (descartando aquellos de las relaciones de herencia), convirtiéndose en un indicador de la comunicación entre las clases. Se define como:

$$FA = \frac{\left[\sum_{i=1}^{TC}\sum_{j=1}^{TC} es\_cliente(C_i, C_J)\right]}{\left(TC^2 - TC\right)}$$

Ecuación 2.25.

Donde i y j van desde 1 hasta TC (Total de clases de sistema), *es_cliente* es una función cuyo valor es 1 si existe una relación entre las clases clientes C_c y la clase servidor C_s, caso contrario es 0. Donde Cc es cada clase C_i y C_s es cada clase C_j, se evalúa progresivamente cada par de clases para logra conocer el valor de *es_cliente*.

[43] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XXIV, Pág. 427

Si FA tiende a crecer, la complejidad del software, su comprensión, mantenimiento y reutilización se verán afectados.

2. Factor de herencia de métodos (FHM)

Este factor se constituye en un indicador del grado en que la arquitectura de clases hace uso de la herencia para métodos y atributos y esta dado por:

$$FHM = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

Ecuación 2.26

Donde las sumatoria va desde i=1 hasta el número Total de Clases (TC) en la arquitectura, C_i es una clase dentro de la arquitectura y $M_a(C_i)$ es el número de métodos que pueden ser invocados en relación con C_i, M_d es el número de métodos declarados en la clase C_i. $M_i(C_i)$ es el número de métodos heredados y no redefinidos en C_i (Ecuación 2.27).

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

Ecuación 2.27

2.8.2.2.2. Métricas CK

Estas métricas fueron propuestas por Chidamber y Kemerer[44] y constan de 6 métricas:

1. Métodos ponderados por clase

[44] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XXIV, Pág. 425

2. Árbol de profundidad de herencia
3. Número de descendientes
4. Acoplamiento entre clase objeto
5. Respuesta a una clase
6. Carencia de cohesión en los métodos.

1. Método Ponderado por Clases (Weight Methods per Class WMC)

WMC Mide la complejidad de una clase, por lo que si todos los métodos son considerados de igual complejidad, entonces WMC es el número de métodos definidos en la clase.

$$WMC = \sum_{i=1}^{n} C_{\bar{i}}$$

ecuación 2.28

Donde C_i es la complejidad del método i.

Cuando la complejidad asignada a cada método es igual a 1, esta medida se transforma en un contador de métodos, siendo considerada en este caso como una medida de tamaño de la clase y no como una de complejidad.

2. Árbol de profundidad de herencia

Se define como el máximo nivel de jerarquía de herencia, es decir la cuenta directa de los niveles de jerarquía de la herencia.

Mientras el valor del árbol de profundidad de herencia aumenta, los niveles mas bajos heredan mayor cantidad de métodos y por tanto la complejidad del sistema aumenta.

3. Número de dependientes

Representa el número de sub clases subordinadas a una clase en la jerarquía, es un nivel de reutilización y del nivel de pruebas requerido, a medida que el número de descendientes crece, la reutilización se incrementa, pero esto puede ocasionar que algunos descendientes de una clase no sean los más apropiados.

4. Respuesta Para una Clase (Response for a Class - RFC)

Es la medida de la complejidad de una clase a través del número de métodos y de su comunicación con otras.

Cuanto mayor es el RFC más complejidad tiene el sistema ya que es posible invocar más métodos como respuesta a un mensaje.

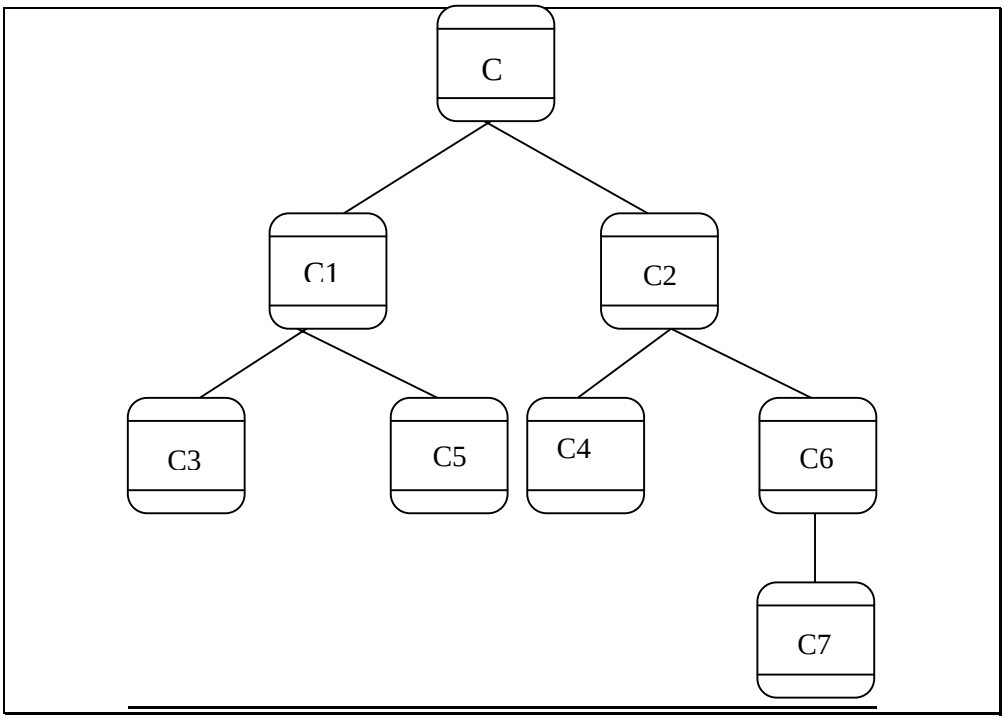


Fig. 2.9. Jerarquía de clases.

5. Carencia de cohesión en los métodos (CCM)

Es el número de métodos locales que acceden a atributos comunes, si no existen atributos comunes a los que se acceden entonces el valor de este indicador es 0.

Esta es una medida que indica la calidad de la abstracción. Si los valores de estas métricas son altos, es un indicador que las clases deben descomponerse en dos o más clases distintas.

2.8.2.2.3. Métricas de Lorenz y Kidd

Estas métricas se dividen en cuatro categorías que son[45]:

1. **Tamaño:** Se centran en el recuento de atributos y operaciones para cada clase y los valores promedio para el sistema como un todo.
2. **Herencia:** Se centran en la reutilización de operaciones en las jerarquías de clases.
3. **Valore Internos:** Examinan la cohesión y los aspectos referentes al código
4. **Valores Externos:** Examina el acoplamiento y la reutilización.

1. Tamaño de Clase (TC)

El tamaño general puede medirse tomando en cuenta las siguientes medidas:

- El total de operaciones de una clase. Esto incluye operaciones propias y heredadas.

[45] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XXIV, Pág. 426.

- El número de atributos de la clase, atributos propios y heredados.

Cuando los valores de TC crecen, se reduce la reutilización de la clase y se dificultan las operaciones de implementación y pruebas.

2. Número de operaciones redefinidas para una sub- clase (NOR)

Redefinir implica reemplazar una operación heredada de una súper clase para uso exclusivo de una sub-clase, por esto un valor alto del NOR implica un problema en el diseño, lo que causa una jerarquía poco útil de clases y un sistema que resulta difícil del probar y modificar.

3. Número de operaciones añadidas a por una sub-clase (NOA)

Las sub-clases se especializan al incrementar operaciones y atributos a la clase, al incrementar el valor de NOA la sub-clase se aleja de la abstracción definida Mientras mas grade es el árbol jerárquico de clases menor debe ser el valor de NOA.

4. Índice de Especialización (IES)

Es un indicador sobre la especialización de sub-clases

$$IES= NOR * NIVEL) / M_{Total}$$

Ecuación 2.29

Donde nivel corresponde al nivel de la jerarquía de clases y M_{total} es el número total de métodos en la clase.

Mientras mas elevado es el valor de IE mas probable será que las jerarquía de clases tenga clases que no corresponda a la abstracción de la súper-clase.

2.8.2.2.4. Acoplamiento entre objetos (Copling between objects- CBO)

Representa el número de clases a las cuales una clase esta ligada sin tener con ella relaciones de herencia.

Evalúa la relación existente entre clases, descartando las relaciones de herencia, permitiendo valorar el esfuerzo requerido para mantenimiento y pruebas.

Mientras menor sea el acoplamiento entre objetos (mas independiente), es mas fácil reutilizarlo y las pruebas deberán ser menos rigurosas.

Tabla 2.10. Relación entre acoplamiento y modularidad, encapsulamiento y complejidad.

Acoplamiento / Característica	Alto	Bajo
Modularidad	Deficiente	Optimizada
Encapsulamiento	Difícil	Facilita
Complejidad	Alta	Bajo

Cuanto menor sea el acoplamiento, la complejidad se reduce, se optimiza la modularidad y el encapsulamiento se facilita.

2.8.2.2.5. Complejidad Ciclomática

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Esta se calcula en base a la prueba del camino básico [46] desarrollada por Tom McCabe[47]. Los casos de prueba obtenidos en el camino básico garantizan que durante la prueba se ejecuta por los menos una vez cada sentencia del programa.

Cuando la complejidad ciclomática se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición.

El cálculo de la complejidad V(G), se puede realizar de tres formas, el número de regiones del grafo de flujo coincide con la complejidad ciclomática.

$$V(G) = A - N + 2$$

Ecuación 2.30

[46] Técnica utilizada para las Pruebas de caja blanca – Pruebas que consisten en evaluar al menos una vez cada sentencia, bucle y decisión lógica de un sistema – que permite encontrar casos de prueba para obtener un conjunto básico de caminos de ejecución; es decir que se asegura que se ejecuta cada sentencia al menos una vez.
[47] Pressman, Rogger. INGENIERÍA DE SOFTWARE. UN ENFOQUE PRÁCTICO. 5ª ED. 2002, Capitulo XIX, Pág. 335

Donde A es el número de aristas del grafo de flujo y N es el número de nodos.

$$V(G) = P + 1$$

Ecuación 2.31

Donde P es el número de nodos predicado contenidos en el grafo de flujo G.

2.8.2.2.6. Métricas de encapsulamiento

El encapsulamiento es una de las principales actividades al diseñar clases, estas deben estar libres de redundancias, es por esto que se proponen métricas para controlar esta característica.

1. **Proporción de métodos ocultos:** Se expresa como una proporción entre los métodos protegidos y privados y la totalidad de métodos en una clase, mientras mayor sea esta proporción, el esfuerzo necesario para corregir defectos disminuye porque los método públicos y privados no pueden ser accedido por clases externas lo que implica una menor dependencia entre clases originando una mayor facilidad para el mantenimiento.

$$PMO = \frac{\sum_{i=1}^{TC} \sum_{m=1}^{M_d(C_i)} (1 - V(M_{mi}))}{\sum_{i=1}^{TC} M_d(c_i)}$$

Ecuación 2.32

Donde

$$V(M_{mi}) = \frac{\sum_{j=1}^{TC} es\_visible(M_{mi}, C_j)}{TC}$$

Ecuación 2.33

$es\_visible(M_{mi}, C_j)$ es una función cuyo valor es 1 si cada clase (C_j) puede llamar a M_{mi}, y 0 en caso contrario, la sumatoria va desde j = 1 hasta el número total de clase (TC).

$M_d(C_i)$ es el número de métodos definidos en la clase C_i sin contar con los métodos heredados. $V(M_i)$ es el porcentaje del total de clases desde las cuales el método M_{mi} es visible.

2. Proporción de atributos ocultos

Se expresa como la proporción entre la suma de los grados de invisibilidad de los atributos en todas las clases y el número total de atributos definidos en el sistema. El grado de invisibilidad de un atributo es el porcentaje sobre el número total de clases desde las cuales un atributo no es visible. Es decir, AHF es la proporción entre los atributos definidos como protegidos o privados y el número total de atributos.

$$\frac{\sum_{i=1}^{TC}\sum_{m=1}^{Ad(Ci)}\left(1-V(Ami)\right)}{\sum_{i=1}^{TC} Ad(Ci)}$$

Ecuación 2.34

Donde

$$V(Ami) = \frac{\sum_{j=1}^{TC} es\_visible(Ami, Ci)}{TC}$$

Ecuación 2.35

es_visible(m_i,C_j) es una función cuyo valor es 1 si C_i puede llamar A m_i

$A_d(C_i)$ es el número de atributos definidos en la clase C_i (no heredados), mientras que $V(A_{mi})$ es la visibilidad: porcentaje del total de clases desde las cuales el atributo A_{mi} es visible.

2.8.2.2.7. Métricas de herencia

Las subclases heredan de la superclase actividades y atributos, pero esta característica no debe interferir con otras como la abstracción, es por esta razón que debe controlarse y medirse esta característica.

1. **Proporción de métodos heredados:** Es la proporción entre la suma de todos los métodos heredados en todas las clases y el número total de métodos, es un indicador del nivel de reutilización.

   Si existe un gran número de métodos heredados, se complica la reutilización de una clase, por lo que el valor de la proporción deseada es de menos del 40% del total de clases.

$$\mathrm{PMH} = \frac{\sum_{i=1}^{TC} M_i(c_i)}{\sum_{i=1}^{TC} M_a(c_i)}$$

Ecuación 2.36

Donde $M_a(c_i) = M_d(c_i) + M_i(c_i)$

$M_a(c_i)$ es el número de métodos disponible.

$M_d(c_i)$ es el número de métodos definidos.

$M_i(c_i)$ es el número de métodos heredados.

TC es el número total de clases.

2. **Proporción de atributos heredados:** Es la proporción entre el número de atributos heredados y el número total de atributos.

$$\mathrm{PAH} = \frac{\sum_{i=1}^{TC} A_i(c_i)}{\sum_{i=1}^{TC} A_a(c_i)}$$

Ecuación 2.37

Donde $A_a(c_i) = A_d(c_i) + A_i(c_i)$ y

$A_a(c_i)$ es el número de atributos disponible.

$A_d(c_i)$ es el número de atributos definidos.

$A_i(c_i)$ es el número de atributos heredados.

TC es el número total de clases.

Es un indicador de la capacidad de reutilización de un sistema, y se recomienda, al igual que la métrica anterior, debe ser menor que el 40% del total de atributos del sistema. Esta medición se realiza en la etapa de diseño e implementación de desarrollo del software

2.8.2.2. Métricas de operaciones

Las métricas de operaciones se enfocan a la conectividad de los módulos del sistema. Se proponen 2 métricas para dicha evaluación.

2.8.2.3.1. Tamaño medio de operación

El número de mensajes enviados por una operación representan una alternativa al número de líneas de código. Un gran número de mensajes enviado por una operación implica una asignación incorrecta de responsabilidades en el diseño.

2.8.2.3.2. Número de parámetros entre operaciones

Mientras mas parámetros existan en una operación, la colaboración entre objetos se vuelve mas complicada, es por eso que esta medida debe mantenerse tan baja como sea posible.

2.8.2.3. Métricas de pruebas OO

Las métricas de diseño y desarrollo influyen de manera directa en el esfuerzo requerido para la realización de pruebas de un sistema. Es por esto que se han organizado en categorías que reflejan características de diseño.

2.8.2.4.1. Carencia de cohesión en métodos (CCM)

Mientras más alto sea este valor, es necesario probar más estados para asegurar que los métodos no produzcan efectos secundarios.

2.8.2.4.2. Porcentaje público y protegido (PPP)

Los atributos protegidos, son una especialización u son privados a subclases específicas, por lo que esta métrica indica el porcentaje de atributos de una clase que son públicos.

2.8.2.4.3. Acceso público a datos miembro (APD)

Indica el número de clases que pueden acceder a los atributos de otras clases, una violación de encapsulación[48].

2.8.2.4.4. Número de clases raíz (NCR)

Esta métrica se relaciona con la jerarquía de clases desarrolladas en el diseño y consiste en proponer un conjunto de pruebas para cada clase raíz de la jerarquía. A medida que el valor de NCR se incrementa, el esfuerzo de comprobación también se incrementa.

2.8.2.4.5. Número de padres directos (NPD)

Se utiliza en el contexto orientado a objetos y por tanto el NPD es una indicación de herencia múltiple[49]. Cuando NPD es mayor a 1, indica que la clase hereda sus atributos y operaciones de más de una clase raíz.

2.8.2.4.6. Número de descendientes (NDD) y árbol de profundidad de herencia

Se indica que los métodos de la superclase tendrán que ser probado nuevamente para cada subclase.

2.9. Planeación y Control de la Calidad del Software

2.9.1. Planeación de la Calidad del Software

La planificación de la calidad facilita el modo para adaptar la planificación del sistema de gestión de la calidad para un proyecto específico. La planificación de la calidad del software a nivel de proyectos debe dirigirse a lo siguiente:

a) Inclusión de los planes de desarrollo, o referencia a los mismos.

[48] Encapsulación: Es el empaquetamiento de una colección de elementos.
[49] Herencia Múltiple: Se produce al heredar un grupo de atributos y operaciones de dos o mas clases

b) Los requisitos de calidad relacionados con los productos y / o procesos.
c) Los sistemas de gestión de la calidad adaptando y / o identificando los procesos e instrucciones específicos apropiados para el ámbito del manual de calidad y algunas exclusiones expuestas (ISO 9001:2000).
d) Los procesos específicos de proyectos e instrucciones, tales como especificación de pruebas del software, detallando los planes, diseños, casos de pruebas y procesos para la unidad, integración, sistemas y pruebas de aceptación.
e) Los métodos, modelos, herramientas, lenguajes de programación, y componentes reutilizables que serán utilizados en los proyectos.
f) Los criterios para determinar el inicio y la finalización de cada fase o etapa del proyecto.
g) Los tipos de análisis y otras verificaciones y actividades de validación que deberán ser ejecutadas.
h) Los procesos de gestión de la configuración que serán realizados.
i) Las actividades de seguimiento y las medidas que se llevaran a cabo.
j) Las personas responsables de aprobar los procesos de salida para su uso posterior.
k) La formación necesaria para el uso de herramientas y técnicas.
l) Los registros que deben efectuarse.
m) La Administración de cambios.

La planificación de calidad permite limitar el proyecto de software de tal manera que sea sencillo identificar los controles y validaciones que deben ser aplicados en cada proceso y etapa del desarrollo de software.

El plan de calidad debe contener:

- Los objetivos de calidad expresados en términos mensurables (cuando sea posible)
- Planificación detallada de ensayos, verificaciones y validaciones; calendarios, recursos y autoridades para la aprobación
- Lista de responsables de revisar y ensayar, gestionar la configuración y el control de cambios y acciones correctivas a los defectos detectados

Los objetivos del aseguramiento de la calidad son reducir, eliminar y prevenir las deficiencias de calidad de los productos a obtener, además de alcanzar una razonable confianza en que las prestaciones y servicios esperados por el cliente o el usuario queden satisfechas.

Para conseguir estos objetivos, es necesario desarrollar un plan de aseguramiento de calidad específico que se aplicará durante la planificación del proyecto de acuerdo a la estrategia de desarrollo adoptada en la gestión del proyecto. En el plan de aseguramiento de calidad se reflejan las actividades de calidad a realizar, estándares a aplicar, productos a revisar, procedimientos a seguir en la obtención de los distintos productos durante el desarrollo y la normativa para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección.

2.9.1.1. Estudio de Viabilidad del Sistema (EVS)

En este proceso el grupo de aseguramiento de calidad inicia el estudio de los sistemas de información definidos en cada alternativa de solución

propuesta, con el fin de identificar las condiciones en que se van a desarrollar y/ o a implantar, así como las características que deben reunir en cuanto a operación, mantenimiento y portabilidad, para satisfacer las necesidades del cliente y los requisitos especificados.

El plan de aseguramiento de calidad debe cubrir todas las necesidades establecidas de modo que, aquellas normas impuestas por los usuarios o clientes que difieran de las existentes en el sistema de calidad, deben quedar también reflejadas en el plan.

El esquema de la Fig. 2.5 muestra la correspondencia entre las actividades del proceso EVS y las de la interfaz de Aseguramiento de la Calidad (IAC).

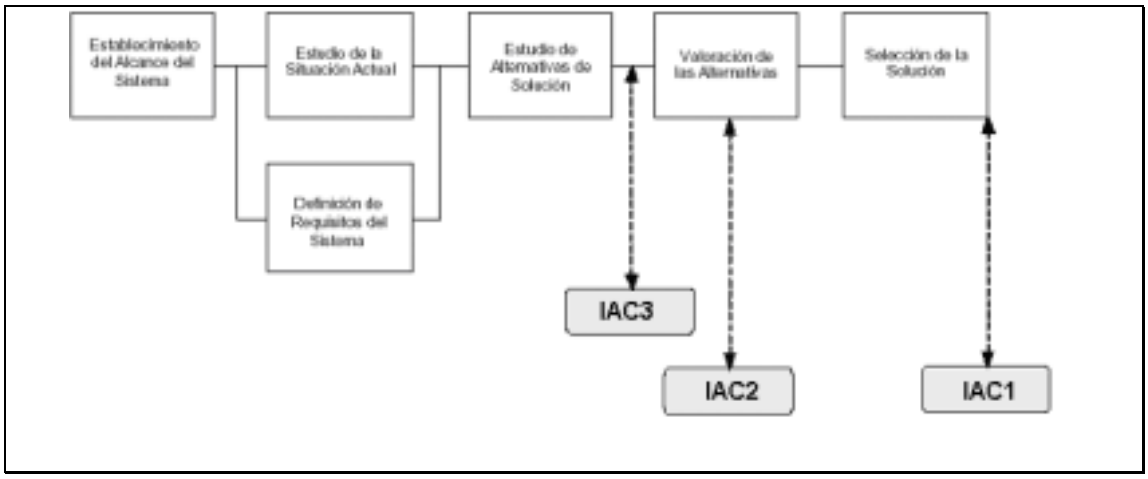


Fig. 2.10 Estudio de viabilidad en el aseguramiento de la calidad

2.9.1.1.1. Identificación de las propiedades de Calidad para el Sistema

1. **Constitución del Equipo de Aseguramiento de Calidad**

   Se constituye el equipo de trabajo inicial necesario para determinar y valorar la conveniencia de establecer un plan de aseguramiento

de calidad para las alternativas de solución propuestas y se fija un plan de acción.

2. **Determinación de los Sistemas de Información objeto de Aseguramiento de Calidad**

   Para cada alternativa de solución propuesta se determina qué sistemas de información van a estar afectados por un plan de aseguramiento de calidad. Para ello se pueden considerar varios criterios relativos a las características que reúna el sistema, como pueden ser:

   - Un período de vida largo.
   - Sujeto a cambios constantes.
   - Equipo de trabajo heterogéneo en el desarrollo del sistema.
   - Operación continúa.
   - Alta disponibilidad.
   - Fuerte interacción con otros sistemas.
   - Portabilidad.

3. **Identificación de las Propiedades de Calidad**

Una vez identificados para cada alternativa propuesta los sistemas de información, se definen para cada uno de ellos aquellas propiedades que permitan evaluar la calidad en cuanto a las características de operación, facilidad de mantenimiento y adaptabilidad a nuevos entornos. Algunas de estas propiedades pueden ser, por ejemplo, la facilidad de uso, eficiencia, seguridad, portabilidad, integridad y fiabilidad.

Tabla 2.11. Identificación de las propiedades de Calidad para el Sistema

TAREAS	PRODUCTOS	TÉCNICAS Y PRACTICAS	PARTICIPANTES
Construcción del equipo de aseguramiento de la calidad	Equipo de Aseguramiento de calidad Plan de acción	Sesiones de trabajo Planificación	Grupo de Aseguramiento de la Calidad Jefe de Proyecto
Determinación de los sistemas de información objeto de Aseguramiento de Calidad.	Sistemas de Información objeto de aseguramiento de calidad	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad Jefe de Proyecto
Identificación de las propiedades de la calidad.	Sistemas de Información objeto de aseguramiento de calidad Propiedades de calidad	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad Jefe de Proyecto

2.9.1.1.2. Establecimiento del Plan de Aseguramiento de Calidad

1. **Necesidad del Plan de Aseguramiento de Calidad para las Alternativas Propuestas**

   Se determina la necesidad de incluir un plan de aseguramiento de calidad en las alternativas propuestas teniendo en cuenta el análisis de riesgos y el enfoque del plan de trabajo establecido en la actividad Valoración de las Alternativas, así como las propiedades de calidad establecidas, para cada sistema de información objeto de aseguramiento de calidad, en la tarea Identificación de las Propiedades de Calidad

   - **Alcance del Plan de Aseguramiento de Calidad**

   Si en la organización existe un Sistema de Calidad, se realiza una valoración de las normas generales establecidas en el mismo, con el fin de adaptarlo al Plan de Aseguramiento de Calidad específico de cada sistema de información implicado, con el que se deben

cubrir las propiedades de calidad identificadas anteriormente. El contenido del plan de aseguramiento de calidad se fijará de acuerdo a los estándares del sistema de calidad e incluirán aspectos tales como:

- Propósito y alcance del plan en términos de propiedades de calidad.
- Objetivos.
- Actividades y tareas relacionadas con el aseguramiento de calidad a realizar a lo largo del desarrollo del software y responsabilidades.
- Productos mínimos exigibles de Ingeniería del Software.
- Estándares, prácticas y normas aplicables durante el desarrollo del software.
- Tipos de revisiones, verificaciones y validaciones que se van a llevar a cabo, así como los responsables de su realización.

Tabla 2.12. Establecimiento del Plan de Aseguramiento de Calidad

Tarea	Productos	Técnicas y prácticas	Participantes
Necesidad del Plan de Aseguramiento de Calidad para las Alternativas Propuestas	• Sistemas de Información objeto de aseguramiento de calidad • Necesidad de un plan de aseguramiento de calidad	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad
Alcance del Plan de Aseguramiento de Calidad	Plan de aseguramiento de calidad de cada alternativa	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad
Impacto en el Costo del Sistema	• Valoración de alternativas • Costo del plan de aseguramiento de calidad	Análisis costo / beneficio	• Grupo de Aseguramiento de la Calidad • Jefe de proyecto

- Criterios para la aceptación o rechazo de cada producto resultante de un proceso.
- Procedimientos para implementar acciones correctivas o preventivas y realizar su seguimiento, identificando responsables.
- Métodos para la salvaguarda y mantenimiento de la documentación obtenida en las actividades de aseguramiento de calidad.

1. **Impacto en el Costo del Sistema**

Una vez identificada la necesidad de un plan de aseguramiento de calidad y definido su alcance, se establece el costo adicional asociado a cada sistema de información, con el fin de aportar esta información al costo total del sistema y determinar su viabilidad económica.

2.9.1.1.3. Adecuación del plan de aseguramiento de calidad a la solución

1. **Ajuste del Plan de Aseguramiento de Calidad**

Una vez estimado los aspectos económicos, técnicos, entre otros, que surjan como consecuencia de incluir un plan de aseguramiento de calidad en la solución seleccionada, puede ser necesario modificar el enfoque de algunas de las propiedades de calidad y el modo en que se solucionen, incluyendo estas variaciones en el plan de aseguramiento de calidad.

2. **Aprobación del Plan de Aseguramiento de Calidad**

Se registra la aprobación del Plan de Aseguramiento de Calidad asociado a cada sistema de información que conforma la solución seleccionada. En caso de rechazo se incluirá el motivo.

Tabla 2.13 Adecuación del plan de aseguramiento de calidad a la solución

Tarea	**Productos**	**Técnicas y prácticas**	**Participantes**
Ajuste del Plan de aseguramiento de Calidad	Plan de aseguramiento de calidad de la alternativa elegida	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad Jefe de Proyecto
Aprobación del Plan de Aseguramiento de Calidad	Registro de aprobación / rechazo		Grupo de Aseguramiento de la Calidad

2.9.1.2. Análisis del Sistema de Información (ASI)

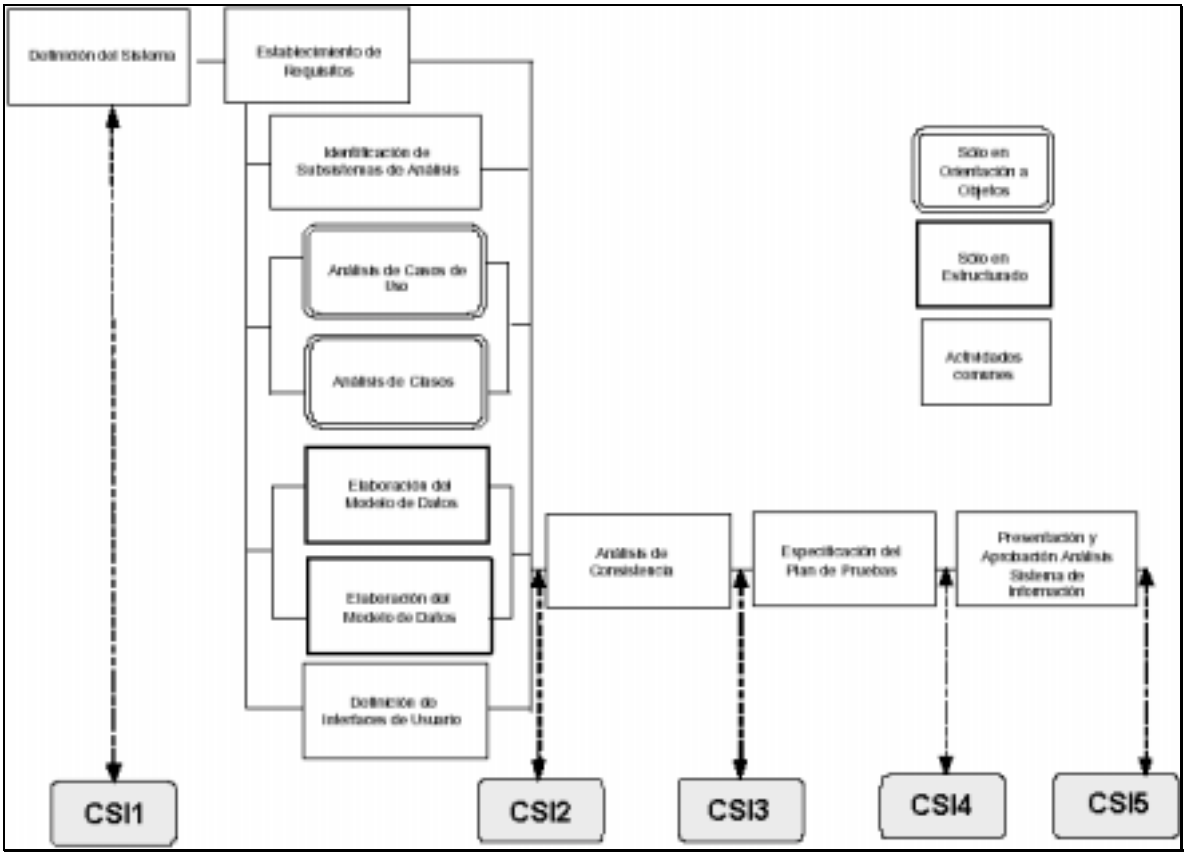


Fig. 2.11 Análisis del Sistema de Información

En este proceso se define de forma detallada el plan de aseguramiento de calidad para un sistema de información, a partir de la especificación

resultante del proceso Estudio de Viabilidad del Sistema (EVS), también se detallan los estándares y normas a cumplir, las revisiones a llevar a cabo y sobre qué productos, así como los procedimientos y mecanismos necesarios para resolver los problemas que surjan, definiendo las acciones preventivas o correctivas a aplicar e identifican los responsables en cada caso.

En el proceso Análisis del Sistema de Información (ASI), el grupo de aseguramiento de calidad se involucra directamente en la revisión de los siguientes productos:

- **Especificación de requerimientos:** Observando la forma en la que ha desarrollado y determinando el grado en que se facilita su comprensión y seguimiento.
- **Modelos resultantes del análisis:** Asegurando que se han verificado y validado y que se ha realizado el seguimiento de requisitos.
- **Plan de pruebas:** Comprobando que se han tenido en cuenta en su definición los criterios establecidos en el Plan de Aseguramiento de Calidad, con el fin de facilitar en los procesos Diseño del Sistema de Información (DSI), Construcción del Sistema de Información (CSI) e Implantación y Aceptación del Sistema (IAS) la revisión de los distintos niveles de prueba.

En la Fig. 2.11 se representa la relación existente entre las actividades del ASI y la construcción del sistema de información CSI.

2.9.1.2.1. Especificación Inicial del Plan de Aseguramiento de Calidad

1. Definición del Plan de Aseguramiento de Calidad para el Sistema de Información

Se especifican de forma clara y detallada:

- Estándares y normas a aplicar durante el proceso de desarrollo.
- Procedimientos para informar, hacer seguimiento y resolver errores, identificando los responsables de llevarlos a cabo.
- Mecanismos de modificación de productos estableciendo LA manera en la que se van a gestionar dichas modificaciones y el modo de comunicarlo a los implicados.
- Modo de valorar las propiedades de calidad.

TABLA 2.14 Especificación Inicial del Plan de Aseguramiento de Calidad

TAREA	PRODUCTOS	TÉCNICAS Y PRÁCTICAS	PARTICIPANTES
Definición del Plan de Aseguramiento de Calidad para el Sistema de Información	Plan de aseguramiento de calidad Aspectos generales	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad . Jefe de Proyecto

2.9.1.2.2. Especificación Detallada del Plan de Aseguramiento de Calidad

1. **Contenido del Plan de Aseguramiento de Calidad para el Sistema de Información**

Una vez conocido el alcance del sistema de información objeto del análisis, se completa el plan de aseguramiento de calidad definido anteriormente, incluyendo:

- Actividades y tareas a realizar en cuanto al aseguramiento de calidad y su ejecución a lo largo del proceso de desarrollo. Se valora, incluyendo en su caso, la realización de auditorias.
- Descripción de cada uno de los productos obtenidos en el proceso de desarrollo

Tabla 2.15 Especificación Detallada Del Plan De Aseguramiento De Calidad

Tarea	**Productos**	**Técnicas y prácticas**	**Participantes**
Contenido del Plan de Aseguramiento de Calidad para el Sistema de Información	Plan de aseguramiento de calidad (detallado) Documento de aseguramiento de calidad (índice)	Sesiones de trabajo	Grupo de Aseguramiento de la Calidad Jefe de Proyecto

- Revisiones a realizar, su propósito y criterios que se deben seguir en la revisión.
- Cronograma para la ejecución de estas actividades, incluyendo los recursos humanos y materiales necesarios para llevarlo a cabo.

Una vez definido el plan de aseguramiento de calidad, toda la información resultante de las actividades llevadas a cabo por el grupo

de aseguramiento de calidad se incluirá en el documento de calidad que formará parte del producto software.

2.9.1.2.3. Revisión del Análisis de Consistencia

Tabla 2.16 Revisión Del Análisis De Consistencia

Tarea	***Productos***	***Técnicas y Prácticas***	***Participantes***
Revisión del Catálogo de Requisitos	Documento de aseguramiento calidad Revisión de requisitos	Revisión técnica	Grupo de Aseguramiento de la Calidad
Revisión de la Consistencia entre Productos	Documento de aseguramiento calidad Revisión de consistencia	Revisión técnica	Grupo de Aseguramiento de la Calidad

1. Revisión del Especificación de Requerimientos

Se valida que los requisitos se han especificado de una forma estructurada de acuerdo a los criterios establecidos en el plan de aseguramiento de calidad y que su contenido es preciso y completo.

2. Revisión de la Consistencia entre Productos

Se comprueba que todos los productos obtenidos se ajustan a las normas y estándares establecidos en el plan de aseguramiento de calidad y que responden a los requisitos especificados.

Se revisa que se ha realizado la verificación y validación de los productos resultantes del análisis, así como el seguimiento de requisitos.

2.9.1.2.4. Revisión del plan de pruebas

1. Revisión del Plan de Pruebas

Se revisan los criterios establecidos en el plan de aseguramiento de calidad para la elaboración del plan de pruebas, desde el punto de

vista de aseguramiento de calidad, con el fin de determinar qué métodos se van a aplicar en la ejecución de las pruebas, cuáles van a ser los criterios de aceptación, cómo se van a realizar las actividades de verificación y cómo se van a emitir los resultados.

Se revisa la existencia en el plan de pruebas de una normativa para la gestión de los resultados de las pruebas, que permita constatar que dichas pruebas se han realizado y descubrir las diferencias entre los resultados esperados y los obtenidos en el caso de incidencias.

Tabla 2.17 Revisión Del Plan De Pruebas

Tarea	Productos	Técnicas y prácticas	Participantes
Revisión del plan de pruebas	Documento de aseguramiento de calidad Revisión del plan de pruebas	Revisión técnica	Grupo de aseguramiento de la Calidad

2.9.1.2.5. Registro de la Aprobación del Análisis del Sistema

1. **Registro de la Aprobación del Análisis del Sistema de Información**

Se registra en el documento de aseguramiento de calidad, la aprobación o rechazo de los productos resultantes del Análisis del Sistema de Información, teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad en cuanto al tipo de productos a entregar, contenido, y normativa aplicada. En caso de rechazo se registran las no conformidades.

TABLA 2.18 Registro de la Aprobación del Análisis del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Registro de la aprobación del análisis del sistema de información	Documento de aseguramiento de la calidad Registro de la aprobación del análisis del sistema de información		Grupo de aseguramiento

2.9.1.3. Diseño del Sistema de Información

Las revisiones del diseño se centran en confirmar que los requisitos especificados en el proceso Análisis del Sistema de Información se han traducido en una arquitectura conforme al entorno tecnológico seleccionado.

Asimismo, se revisan los requisitos que deben cumplir los distintos niveles de pruebas (unitarias, de integración, del sistema, de implementación y aceptación) especificados en el plan de pruebas, de acuerdo a los criterios de revisión establecidos en el plan de aseguramiento de calidad. También se realiza una revisión de la identificación de los requisitos no funcionales relacionados con la documentación de usuario e implantación.

2.9.1.3.1. Revisión de la Verificación de la Arquitectura del Sistema

1. Revisión de la Consistencia entre Productos del Diseño

Se comprueba que todos los productos resultantes del diseño se ajustan a las normas y estándares establecidos en el plan de aseguramiento de calidad y se revisa que se ha realizado la verificación y validación de los mismos.

Se verifica que el diseño de la arquitectura del sistema responde a los requisitos especificados en el análisis.

2. Registro de la Aceptación de la Arquitectura del Sistema

Se verifica que los responsables de operación están de acuerdo con el diseño de la arquitectura del sistema, teniendo en cuenta el entorno tecnológico en el que va a estar operativo.

Se registra la aprobación o rechazo de los productos resultantes teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad. En caso de rechazo se registran las no conformidades.

Tabla 2.19 Revisión de la Verificación de la Arquitectura del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de la consistencia entre los productos del diseño	Documento de aseguramiento de calidad Revisión de la arquitectura del sistema	Revisión técnica	Grupo de aseguramiento de la calidad
Registro de la aceptación de la arquitectura del sistema	Documento de aseguramiento de la calidad Registro de la aceptación de la arquitectura del sistema		Grupo de aseguramiento de la calidad

2.9.1.3.2. Revisión de la Especificación Técnica del Plan de Pruebas

1. Revisión del Diseño de las Pruebas Unitarias, de Integración y del Sistema

Según los criterios de revisión establecidos en el plan de aseguramiento de calidad, se comprueba que el diseño de las pruebas unitarias, de integración y del sistema, cumplen dichos criterios, en

cuanto a la especificación de verificaciones, casos de prueba asociados a cada verificación, registro de resultados de las pruebas, informes de incidencias en la ejecución, etc. Con respecto a la especificación de los casos de prueba, se comprobará que se han tenido en cuenta las propiedades de calidad establecidas anteriormente.

Asimismo, en función de las características del sistema de información, se determina según los criterios de selección establecidos en el plan de aseguramiento de calidad sobre qué verificaciones y casos de prueba, unitarias, de integración y del sistema, se va a llevar a cabo la revisión.

TABLA 2.20 Revisión de la Especificación Técnica del Plan de Pruebas

Tareas	Productos	Técnicas y prácticas	Participante
Revisión del diseño de las pruebas unitarias, de integración y del sistema	Documento de aseguramiento de calidad Revisión del diseño de las pruebas		Grupo de aseguramiento de la calidad
Revisión del pan de pruebas	Documento de aseguramiento de calidad Revisión del plan de pruebas	Revisión técnica	Grupo de aseguramiento de la calidad

2. Revisión del Plan de Pruebas

Se verifica que en el plan de pruebas se han detallado tanto las pruebas de implantación como las de aceptación. Las verificaciones asociadas a las pruebas de implantación deben contemplar los requisitos no funcionales relacionados con las propiedades de calidad.

En las pruebas de aceptación, se revisará que las verificaciones y casos de prueba propuestos van dirigidos a la comprobación de los criterios de aceptación establecidos por el usuario.

Se revisa la planificación de las pruebas con el fin de establecer el propio plan de acción del grupo de aseguramiento de calidad.

2.9.1.3.3. Revisión de los Requisitos de Implantación

1. Revisión de los Requisitos de Documentación de Usuario

Se verifica que se han identificado los requisitos necesarios relativos a la documentación que se va a entregar a los usuarios y a operación (tipos de documentos y estructura, formato en que se desarrollarán, estándares a seguir, soporte, número de copias a editar, entre otros).

TABLA 2.21 Revisión de los Requisitos de Implantación

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de los requisitos de documentación de usuario	Documento de aseguramiento de calidad Revisión de los requisitos de documentación de usuarios	Revisión técnica	Grupo de aseguramiento de la calidad
Revisión de los requisitos de implantación		Revisión técnica	Grupo de aseguramiento de la calidad

Revisión de los Requisitos de Implantación

Se comprueba que se han identificado y detallado los requisitos necesarios para la implantación del sistema relacionados con la instalación, formación e infraestructura.

2.9.1.3.4. Registro de la Aprobación del Diseño del Sistema de Información

1. **Registro de la Aprobación del Diseño del Sistema de Información**

   Se registra la aprobación o rechazo de los productos resultantes del Diseño del Sistema de Información, teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad en cuanto al tipo de productos a entregar, contenido y normativa aplicada. En caso de rechazo se registrarán las no conformidades.

Tabla 2.22 Registro de la Aprobación del Diseño del Sistema de Información

Tareas	Productos	Técnicas y prácticas	Participante
Registro de la aprobación del diseño del sistema de información	Documento de aseguramiento de calidad Registro de la aprobación del diseño del sistema de información		Grupo de aseguramiento de la calidad

2.9.1.4. Construcción del Sistema de Información

En este proceso el grupo de aseguramiento de calidad revisa los estándares de nomenclatura y normativa aplicada en la generación del código de componentes, en la evaluación de los resultados de las pruebas, en los manuales de usuario y en el esquema de formación.

Con respecto a las pruebas, se revisa que se han llevado a cabo las pruebas unitarias, de integración y del sistema según los criterios de selección de verificaciones y casos de prueba asociados que se habrán fijado en el plan de aseguramiento de calidad.

El esquema de la Fig. 2.6 se muestra la correspondencia entre las actividades del proceso Construcción del sistema de Información (CSI) y las de la interfaz de Aseguramiento de la Calidad.

2.9.1.4.1. Revisión del Código de Componentes y Procedimientos

1. Revisión de Normas de Construcción

Se comprueba que se ha generado el código de los componentes y de los procedimientos de operación y seguridad de acuerdo con los criterios de nomenclatura y normativa vigentes en la organización y especificadas en el proceso Diseño del Sistema de Información. Si se considera oportuno se puede llevar a cabo una revisión más detallada del código por alguna propiedad de calidad relativa a la modularidad o autodefinición.

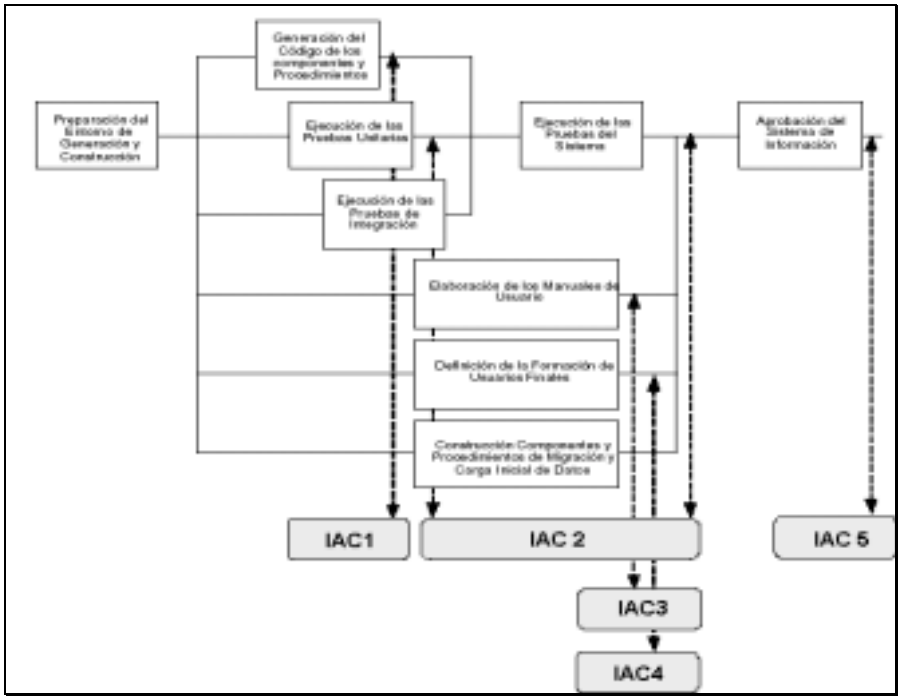


Fig. 2.12 Construcción Del Sistema De Información

2.9.1.4.2. Revisión de las Pruebas Unitarias, de Integración y del Sistema

1. Revisión de la Realización de las Pruebas de Integración

Se comprueba la realización de las pruebas de integración. Se lleva a cabo la revisión de las verificaciones y los casos de prueba que se hayan determinado aplicando los criterios de selección y de acuerdo al modo que se recogió en los criterios de revisión del plan de aseguramiento de calidad.

Para todo esto, se tendrá en cuenta la normativa establecida para la gestión de los resultados de dichas pruebas.

TABLA 2.23 Revisión del Código de Componentes y Procedimientos

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de estándares	Documento de aseguramiento de calidad (actualizando con la revisión del código de componentes y procedimientos)	Revisión técnica	Grupo de aseguramiento de la calidad

2. Revisión de la Realización de las Pruebas del Sistema

Se comprueba la realización de las pruebas del sistema. Se lleva a cabo la revisión de las verificaciones de los casos de prueba que se hayan determinado de acuerdo al modo que se recogió en los criterios de revisión del plan de aseguramiento de calidad.

TABLA 2.24 Revisión de las Pruebas Unitarias, de Integración y del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de la realización de las pruebas unitarias	Documento de aseguramiento de la calidad Revisión de la realización de las pruebas unitarias	Revisión formal	Grupo de aseguramiento de la calidad
Revisión de la realización de las pruebas de integración		Revisión formal	Grupo de aseguramiento de la calidad
Revisión de la realización de las pruebas del sistema		Revisión formal	Grupo de aseguramiento de la calidad

Para todo esto, se tendrá en cuenta la normativa establecida para la gestión de los resultados de dichas pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta.

Igualmente se revisarán las incidencias no resueltas con el fin de validar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente.

2.9.1.4.3. Revisión de los Manuales de Usuario

1. Revisión de los Manuales de Usuario

Se comprueba que los manuales de operación y de usuario se han descrito de forma clara y concisa y se ajustan a los criterios y normativa establecidos.

TABLA 2.25 Revisión de los Manuales de Usuario

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de los manuales de usuario	Documento de aseguramiento de calidad Revisión de los manuales de usuario	Revisión técnica	Grupo de aseguramiento de la calidad

2.9.1.4.4. Revisión de la Formación a Usuarios Finales

Se revisa que se han definido los esquemas de formación a los usuarios finales del sistema de información y que se han identificado los distintos perfiles de usuario en función de sus capacidades, habilidades, experiencia y responsabilidades, así como los recursos necesarios para llevarlo a cabo.

TABLA 2.26 Revisión de la Formación a Usuarios Finales

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de la formación a usuarios	Documento de aseguramiento de la calidad Revisión de la formación a usuarios finales.		Grupo de aseguramiento de la calidad

2.9.1.4.5. Registro de la Aprobación del Sistema de Información

1. Registro de la Aprobación del Sistema de Información

Se registra la aprobación o rechazo del sistema de información construido teniendo en cuenta los criterios establecidos en el plan de aseguramiento de calidad en cuanto al tipo de productos a entregar, contenido y normativa aplicada. En caso de rechazo se registrarán las no conformidades.

TABLA 2.27 Registro de la Aprobación del Sistema de Información

Tareas	Productos	Técnicas y prácticas	Participante
Registro de las aprobaciones del sistema de información	Documento de aseguramiento de calidad Registro de la aprobación del sistema de información		Grupo de aseguramiento de la calidad

2.9.1.5. Implantación y Aceptación del Sistema

El grupo de aseguramiento de calidad en este proceso es responsable de revisar la existencia de un plan de implantación que se habrá elaborado conforme a la estrategia de implantación determinada en el proceso Estudio de Viabilidad del Sistema (EVS) y teniendo en cuenta los requisitos de implantación establecidos en el proceso Diseño del Sistema de Información (DSI).

También deben comprobar que se han realizado las pruebas de implantación y de aceptación según el plan de pruebas establecido en MÉTRICA Versión 3 y la normativa acordada en el plan de aseguramiento de calidad. Revisan la totalidad de las verificaciones y casos de prueba de implantación y aceptación que se hayan especificado para el sistema y las incidencias producidas, con el fin de determinar si puede verse afectada alguna propiedad de calidad. En cualquier caso, se registra la aprobación de las pruebas de implantación y de aceptación por parte de operación y del usuario respectivamente.

En cuanto al mantenimiento, el grupo de aseguramiento de calidad debe asegurar que se le entrega el producto software al responsable de mantenimiento, con las propiedades adecuadas para que pueda asumir el

servicio de mantenimiento, una vez que el sistema se encuentre en producción.

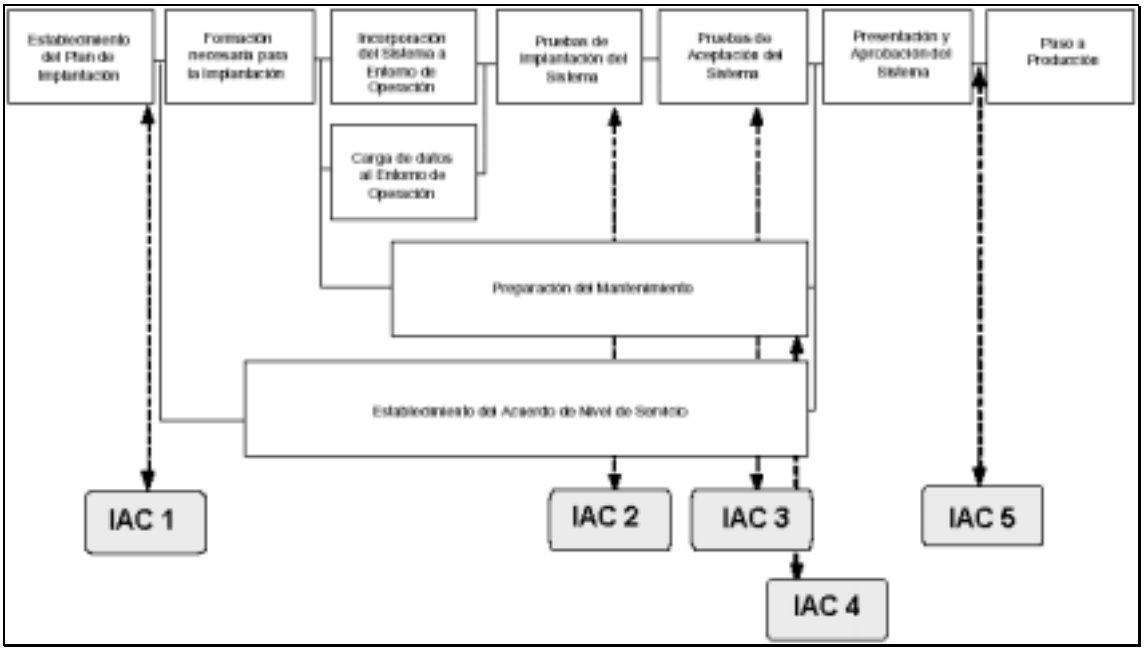


FIG 2.13 Implantación y Aceptación del Sistema

La Fig. 2.13 muestra la correspondencia entre las actividades del proceso IAS y las de la interfaz de Aseguramiento de la Calidad.

2.9.1.5.1. Revisión del Plan de Implantación del Sistema

1. Revisión del Plan de Implantación del Sistema

Se revisa que se ha elaborado un plan de implantación de acuerdo a la estrategia de implantación establecida en el proceso Estudio de Viabilidad del Sistema (EVS) y conforme a los requisitos de implantación establecidos.

Asimismo, se comprueba que se ha establecido un plan de trabajo para la implantación que permita determinar las actividades a realizar por el grupo de aseguramiento de calidad durante el proceso de implantación.

Se determinan productos de entrada y salida, técnicas, prácticas y participantes para esta actividad.

TABLA 2.28 Revisión del Plan de Implantación del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Revisión del plan de implantación del sistema	+ Documento de aseguramiento de calidad + Revisión del plan de implantación	Revisión técnica	Grupo de aseguramiento de la calidad

2.9.1.5.2. Revisión de las Pruebas de Implantación del Sistema

Para esto se tendrá en cuenta la normativa establecida para la documentación de los resultados de dichas pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta.

Igualmente se revisarán las incidencias no resueltas con el fin de valorar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente.

1. **Registro de la Aprobación de las Pruebas de Implantación del Sistema**

Se registra la aprobación o rechazo de las pruebas de implantación por parte del Revisión de las Pruebas de Implantación del Sistema

Tabla 2.29. Revisión de las pruebas de la implantación del sistema.

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de la realización de las pruebas de implantación del sistema	Documento de aseguramiento de la calidad Revisión de las pruebas de implantación	Revisión formal	Grupo de aseguramiento de la calidad
Registro de las pruebas de implantación del sistema	Documento de aseguramiento de calidad Registro de la aprobación de las pruebas de implantación por operación		Grupo de aseguramiento de la calidad

2.9.1.5.3. Revisión de las Pruebas de Aceptación del Sistema

1. Revisión de la Realización de las Pruebas de Aceptación del Sistema

Se comprueba la realización de las pruebas de aceptación. Se lleva a cabo la revisión de las verificaciones y casos de prueba que se hayan determinado para cada sistema de información implicado en la implantación del sistema, de acuerdo al modo que se recogió en los criterios de revisión de los respectivos planes de aseguramiento de calidad.

Para todo esto, se tendrá en cuenta la normativa establecida para la documentación de los resultados de dichas pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta.

Tabla 2.30 Revisión de las Pruebas de Aceptación del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de la realización de las pruebas de aceptación del sistema	Documento de aseguramiento de calidad Revisión de la realización de la realización de las pruebas de aceptación	Revisión formal	Grupo de aseguramiento de la calidad
Registro de la aprobación de las pruebas de aceptación del sistema	Documento de aseguramiento de calidad Registro de la aprobación de las pruebas de aceptación por el usuario		Grupo de aseguramiento de la calidad

Igualmente se revisarán las incidencias no resueltas con el fin de valorar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente.

2. Registro de la Aprobación de las Pruebas de Aceptación del Sistema

Se registra la aprobación o rechazo de las pruebas de aceptación por parte del responsable de usuarios finales.

2.9.1.5.4. Revisión del Plan de Mantenimiento del Sistema

1. Revisión del Plan de Mantenimiento del Sistema

Se comprueba que los productos entregados al responsable de mantenimiento son los acordados y que éste asume el mantenimiento del sistema de información.

Así mismo, se comprueba que se ha formalizado un plan de mantenimiento para el sistema de información, entre el cliente/usuario y el responsable de mantenimiento. Si se considera conveniente, se estudiará la necesidad de llevar a cabo un

seguimiento y control de la calidad en los sistemas de información, una vez se encuentren en el entorno de producción.

Tabla 2.31 Revisión del Plan de Mantenimiento del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Revisión del plan de Mantenimiento del sistema	Documento de aseguramiento de la calidad Revisión del plan de mantenimiento	Revisión técnica	Grupo de aseguramiento de la calidad

2.9.1.5.5. Registro de la Aprobación de la Implantación del Sistema

1. Registro de la Aprobación de la Implantación del Sistema

Se registra la aprobación de la implantación del sistema y se comprueba que el documento de aseguramiento de calidad forma parte del producto software.

Tabla 2.32 Registro de la Aprobación de la Implantación del Sistema

Tareas	Productos	Técnicas y prácticas	Participante
Registro de la aprobación de la implantación del sistema	Documento de aseguramiento de calidad Registro de la aprobación de la implantación del sistema		Grupo de aseguramiento de la calidad

2.9.1.6. Mantenimiento del Sistema de Información

En el proceso Implantación y Aceptación del Sistema se habrá determinado la necesidad de llevar a cabo un seguimiento y control de la calidad en los sistemas de información, una vez se encuentren en el entorno de producción.

El grupo de aseguramiento de calidad intervendrá durante el mantenimiento, efectuando revisiones de seguimiento periódicas, más o menos frecuentes según los casos, que sirvan para constatar que el mantenimiento establecido para el sistema de información se realiza de forma correcta.

En algún caso, según las implicaciones del cambio, puede ser necesario revisar puntualmente:

- El contenido del plan de pruebas de regresión.
- La ejecución de las pruebas de regresión según la normativa acordada en el plan de aseguramiento de calidad.
- Las verificaciones y casos de prueba que se hayan incluido en el plan de pruebas para los cambios producidos por una petición.
- Las incidencias detectadas con el fin de determinar si puede verse afectada alguna propiedad de calidad.

En caso de revisar la ejecución de las pruebas de regresión, se registrará la aprobación de las pruebas por el responsable de mantenimiento.

En el siguiente gráfico se aprecian las actividades de Aseguramiento de la Calidad durante el Mantenimiento del Sistema de Información.

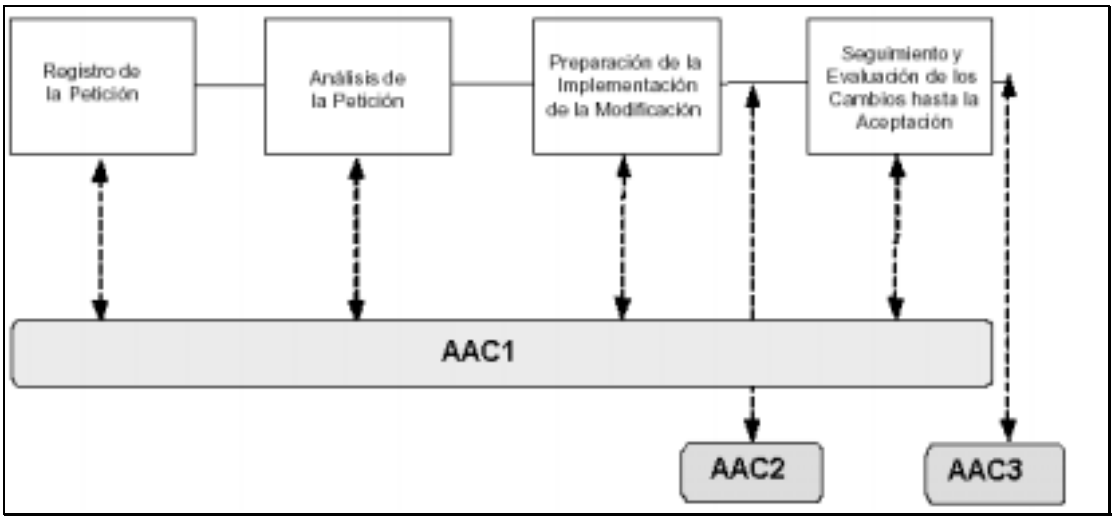


FIG 2.14 Mantenimiento del Sistema de Información

2.9.1.6.1. Revisión del Mantenimiento del Sistema de Información

TABLA 2.33 Revisión del Mantenimiento del Sistema de Información

Tareas	Productos	Técnicas y prácticas	Participante
Revisión del mantenimiento	Informe de seguimiento del mantenimiento	Revisión técnica	Grupo de aseguramiento de la calidad

1. Revisión del Mantenimiento

Se verifica que las peticiones incluidas en la especificación de requerimientos se corresponden con las previstas en la revisión del plan de mantenimiento del Documento de aseguramiento de la calidad. Se realiza una revisión periódica del catálogo de requisitos comprobando que se mantiene actualizado. Asimismo, se revisa que el usuario acepta o rechaza la solución propuesta para dar respuesta a su petición y que aprueba formalmente el cierre de la petición.

Esta tarea de la interfaz de aseguramiento de calidad se aplica a todas las actividades del proceso Mantenimiento de Sistemas de Información.

2.9.1.6.2. Revisión del Plan de Pruebas de Regresión

TABLA 2.34 Revisión del Plan de Pruebas de Regresión

Tareas	Productos	Técnicas y prácticas	Participante
Comprobación de la existencia del plan de pruebas de regresión	Informe de revisión del plan de pruebas de regresión	Revisión técnica	Grupo de aseguramiento de la calidad

1. Comprobación de la Existencia del Plan de Pruebas de Regresión

Se revisa que se ha establecido un plan de pruebas de regresión de acuerdo a los criterios establecidos en el plan de aseguramiento de calidad para la elaboración del plan de pruebas desde el punto de vista de aseguramiento de calidad, con el objetivo de determinar qué métodos se van a aplicar para la ejecución de las pruebas, cuáles van a ser los criterios de aceptación, cómo se van a realizar las actividades de verificación y cómo se van a emitir los resultados.

Se revisa la existencia de una normativa para la gestión de los resultados de las pruebas.

2.3.1.6.1. Revisión de la Realización de las Pruebas de Regresión

TABLA 2.35. Revisión de la Realización de las Pruebas de Regresión

Tareas	Productos	Técnicas y prácticas	Participante
Revisión de la realización de las pruebas de regresión	Informe de revisión de la realización de las pruebas de regresión	Revisión formal	Grupo de aseguramiento de la calidad

1. **Revisión de la Realización de las Pruebas de Regresión**

   Se comprueba que se han realizado las pruebas de regresión y se lleva a cabo la revisión de las verificaciones y casos de prueba que se hayan determinado para la correcta implantación del cambio.

   Para todo esto, se tendrá en cuenta la normativa establecida para la gestión de los resultados de dichas pruebas. En el caso de existir casos de prueba adicionales, incorporados como consecuencia de las medidas correctoras tomadas para solventar los errores detectados, el grupo de aseguramiento de calidad revisará que se han resuelto de forma correcta.

   Igualmente, se revisarán las incidencias no resueltas con el fin de valorar hasta que punto se ven comprometidas las propiedades de calidad establecidas inicialmente. Se registra la aprobación por parte del responsable de mantenimiento.

2.9.2. Control de la Calidad del Software

El control de la calidad es una serie de inspecciones, revisiones, pruebas utilizados a lo largo del proceso del software para asegurar que cada producto cumple con los requisitos y característica de calidad en el grado requerido, este control de calidad permite obtener mejoramiento continuo en el proceso de desarrollo de software.

Es importante definir, para un proceso de control de calidad exitoso, todos los productos y los factores mesurables con los que se pueda evaluar los resultados obtenidos.

Cuando un producto no posee una determinada característica de calidad se dice que tiene un DEFECTO[50]. Partiendo de esto el objeto del Control de Calidad es identificar defectos en el producto y corregirlos.

Se pueden clasificar las actividades de control de calidad en dos categorías: controles estáticos y controles dinámicos tales como se indica en la Fig. 2.15. Los primeros analizan el objeto sin necesidad de ejecutarlo mientras que los segundos requieren la ejecución del objeto que está siendo probado.

2.9.2.1. Controles estáticos manuales informales

1. **Comprobación de escritorio (Desk Checking):**

   Esta prueba pretende examinar a mano e individualmente el objeto que se acaba de desarrollar. Es el método más tradicional para analizar un programa. Debe ser cuidadoso y concienzudo para que sea efectivo. Cabe destacar que es más efectivo si se hace un intercambiando del objeto a examinar con otro compañero.

Fig.2.15. Clasificación de las actividades de control

2.9.2.2. Controles estáticos manuales disciplinados

Los controles estático manuales pasan a ser técnicas de grupo, donde su misión principal es conseguir que la responsabilidad del control de calidad no recaiga sólo sobre el propio desarrollador.

[50] Defecto: Carencia de las cualidades propias del proyecto de software, respecto a la ERS.

2.9.3. Auditorias

Una auditoria consiste en realizar una investigación para determinar:

- El grado de cumplimiento y la adecuación de los procedimientos, instrucciones, especificaciones, códigos, estándares u otros requisitos de tipo contractual establecidos.
- La efectividad y adecuación de la implementación realizada.

Se pueden considerar tres tipos de auditorias:

- **Auditoria del producto:** El objetivo es cuantificar el grado de conformidad del producto con las características requeridas. Las auditorias del producto software más comunes son la auditoria[51] Funcional y la auditoria Física[52].
- **Auditoria del proceso:** El objetivo es evaluar el proceso de desarrollo o de gestión, y evaluar su completitud y efectividad, determinando dónde se puede mejorar. En el desarrollo de software se suelen realizar dos tipos de auditorias del proceso:
- **Auditoria de proyecto:** El objetivo es evaluar la productividad y eficacia del equipo que trabaja en un proyecto así como la efectividad de los métodos y herramientas utilizados.
- **Auditoria de gestión de proyecto:** El objetivo es evaluar la efectividad de las prácticas de gestión realizadas y la organización del proyecto.
- **Auditoria del sistema de calidad:** El objetivo es evaluar la complejidad y efectividad del propio sistema de calidad de software.

[51] Auditoria Funcional: Es un estudio de la funcionalidad de los procesos y la integridad de los mismos.
[52] Auditoria Física: Es un estudio físico del software y de los requerimientos para su uso.

2.9.4. Revisiones

Se puede definir una revisión como una reunión formal en la que se presenta el estado actual de los resultados de un proyecto a un usuario, cliente, entre otros.

Con las revisiones se consigue que el peso de la evaluación técnica no recaiga a las mismas personas involucradas en la producción del software,

que por la posición que ocupan no pueden ser totalmente objetivas, sino en otras personas técnicamente competentes y objetivas.

Las revisiones son, en la actualidad, el único método de control de calidad eficaz en las fases iniciales del desarrollo a la hora de identificar desviaciones con respecto a las especificaciones de calidad.

Las revisiones redundan en una mejora directa de la calidad del objeto que se examina y provocan, indirectamente, una mejora de la calidad del proceso de desarrollo, al facilitar la comunicación entre los miembros del equipo de desarrollo. Al mismo tiempo facilitan el control del costo y el tiempo.

La diferencia más importante entre las revisiones y las auditorias, radica en que el objetivo de las revisiones es detectar defectos, es por eso que se lleva a cabo a lo largo de cada fase mientras que el objetivo de las auditorias es certificar conformidad e identificar desviaciones por eso se lo lleva a cabo en las fases finales.

2.9.2.3. Controles estáticos automáticos

La mayor parte del análisis estático automático del código, lo realizan los compiladores, que pueden detectar desde expresiones con sintaxis

incorrectas hasta incompatibilidades de tipo y otros errores de tipo semántico.

Otras técnicas de análisis estático automático de programas son:

- Análisis de Flujo
- Ejecución simbólica
- Algoritmos
- Pseudo código

1. Análisis de flujo

Se basa en una representación gráfica para lo cual se usan grafos en los que los nodos representan sentencias o segmentos de programa y los arcos posibles transiciones de control desde un segmento a otro.

2. Ejecución simbólica

El resultado de una ejecución simbólica es generalmente una expresión larga y compleja. La mejor forma de analizarla es descomponerla en una estructura de árbol, donde cada hoja representa un camino de ejecución y cada ramificación representa un punto de decisión en el programa.

El árbol resultante se puede usar también como ayuda para generar casos de prueba. Aunque esta técnica puede ser muy útil, si el programa no tiene ciclos, el árbol resultante es finito, pero si tiene ciclos el árbol es infinito. Lo que se hace es tomar un número finito de iteraciones[53].

2.9.2.4. Controles dinámicos

Dentro de la trayectoria del software, no se ha desarrollado ninguna teoría universalmente aceptada acerca de la prueba de software. Lo único que

[53] Iteraciones: Es el número de veces que se realiza.

hay es un conjunto de aproximaciones metodológicas que facilitan y hacen más eficiente el proceso de prueba.

Se llama ***PRUEBA*** del Software al proceso en el que se ejecuta un sistema con el objetivo de detectar fallos.

Se llama ***DEPURACIÓN*** al proceso en el que se localiza el defecto que es la causa de un fallo, se determina la forma de corregirlo, se evalúa el efecto de la corrección y se lleva a cabo la corrección. Por lo general, después del proceso de depuración será necesario repetir el proceso de prueba, para garantizar que el defecto quedó efectivamente corregido.

Aunque la prueba es una parte importante del Control de Calidad, es importante darse cuenta de que no es la única.

2.9.2.4.1. Tipos de pruebas

El proceso de prueba conlleva la realización de un conjunto de tareas a lo largo del ciclo de vida del sistema. De acuerdo con el estándar IEEE 1012-1986 el conjunto mínimo de pruebas que se deben realizar son:

- **La prueba modular** consiste en la prueba de cada módulo aislado del resto del sistema.
- **La prueba de integración** se realiza a medida que los diferentes módulos del sistema se integran en el mismo.
- **La prueba del sistema** se realiza cuando se han integrado todos los módulos, y su objetivo es comprobar que el sistema satisface los requisitos del usuario, tanto los funcionales como los no funcionales.

- **La prueba de aceptación** se realiza una vez que el sistema se ha implantado en su entorno real de funcionamiento, y su objetivo es demostrar al usuario que el sistema satisface sus necesidades.
- **La prueba de regresión** esta prueba se realiza al presentar una nueva versión de un producto de software y su objetivo es constatar que la introducción de cambios a una versión previa, no afectó la calidad que tenia originalmente.

CAPITULO III GUIA PARA LA GESTION DE LA CALIDAD DE SOFTWARE

3.1. Introducción

Es importante mantener una base de datos de los proyectos de software realizados o en ejecución de manera que sirva como retroalimentación para realizar mejoras en proyectos futuros, siendo necesario plantear procesos definidos para el desarrollo de software en cada una de sus etapas así como una librería de documentación de cada proyecto, el que se utilizara en el futuro como estándar para proporcionar calidad, eficiencia y eficacia en productos y procesos futuros[54].

La guía de gestión de la Calidad del Software se enfocará en tres áreas:

- Ingeniería
- Gestión
- Calidad

En donde cada una de estas áreas englobará temas relacionados con ellas, es así que el área de ingeniería comprenderá cada fase de la ingeniería de software (Análisis de requisitos, Diseño, Validación y pruebas y Mantenimiento), el Área de Gestión cubrirá la panificación, estimación y gestión de riesgos, mientras que el Área de Calidad se encargará de la planeación y control de la calidad del proyecto de software.

Es importante mencionar que la presente guía se dirige a las metodologías de desarrollo orientados a Datos y Objetos, por lo que en los apartados pertinentes se diferenciara las actividades a realizar en cada uno de ellos.

[54] Anexo C. Practicas Clave *"Enfoque al proceso de la organización", "Definición de procesos de la organización"* e *"Ingeniería de productos de software".*

3.1.1. Objetivos

Determinar un marco de gestión de calidad que permita al grupo de desarrollo obtener indicadores de calidad en cada fase de la ingeniería de software.

3.1.2. Alcance

Abarca a las fases de análisis y diseño de la ingeniería de software y comprende los documentos que se entregarán al final de cada fase, así como los que se recibirán al inicio de las mismas, métricas para ponderar la calidad de los procesos envueltos en cada fase así como los mencionados documentos.

3.1.3. Técnica

QFD es la técnica que se utiliza en esta guía ya que se presta mayor atención a la recolección de requerimientos del cliente en la fase de análisis asegurando de esta manera que las siguientes fases del proyecto cumplan a en menor tiempo asegurando la calidad en los mismo, evitando así un reproceso (reingeniería) en las distintas fases de la Ingeniería de Software.

La utilización de QFD propone la consecución de cuatro etapas *(Capitulo II, sección 2.2.1.2, Pág. 44)*, por lo que es necesario realizar un plan de proyecto se software *(Capitulo III, sección 3.2.1, Pág. 179)* en la etapa de organización y plantación, el estudio de requerimientos de software *(Capitulo III, sección 3.4.1.4, Pág. 193)* en la etapa descriptiva. La etapa de progreso consiste en las fases de diseño, desarrollo y pruebas con su respectiva Configuración del Software descrita en el (Capitulo III, sección 3.3.2, Pág. 189).

La metodología del QFD propone cuatro matrices básicas en las que se pretende evaluar los factores relevantes del proyecto de software

(referencia) la matriz A1 refleja la evaluación de los requerimientos de usuario, la misma que se realiza a través de métricas en la fase de análisis de la ingeniería de software (referencia) y representado en la matriz de Reporte de Resultados de Métricas *(Capitulo 3, Sección 3.3.2.2, Pág. 189),* la cual es una relación de la matriz de relaciones, la matriz de a voz del ingeniero se ve reflejada en el Area de ingeniería ya que cada fase evalúa factores específicos de calidad.

3.2. Área de Gestión

El área de gestión se enfoca en dos áreas básicas: Planificación del proyecto de software y la Gestión de Configuración del Software. La primera área se encarga del estudio preliminar necesario para la realización del proyecto, mientras que la Gestión de Configuración de Software se encarga de recoger todos los cambios y versiones producidas en el proyecto de software.

3.2.1. Planificación del proyecto de software

De acuerdo a lo expuesto en Gestión de Proyectos de Software (Capítulo 2, Sección 2.3, Pág. 46), una correcta planificación parte de un establecimiento claro de los objetivos y de los límites para un proyecto de software. Esto permitirá al administrador del proyecto realizar una estimación objetiva de los recursos necesarios para el desarrollo del mismo[55] y una adecuada planificación de calidad dentro de las distintas etapas.

El reto del gestor de proyectos es entonces realizar una predicción sobre los recursos temporales, tecnológicos, humanos y económicos.

[55] Anexo C. Practica Clave Nivel 2 *"Planeación del proyecto de software"*

3.2.1.1. Estudio Preliminar de Proyecto

Esta actividad consiste en determinar los objetivos y limitaciones del módulo o sistema a construir así como el ámbito de operación del mismo. Para alcanzar la determinación del ámbito del sistema, se debe organizar reuniones con los clientes, de esta manera se puede determinar las existencias de software y las plataformas tecnológicas con las que cuenta el cliente.

I. Software Existente

Esta sección pretende recopilar información referente al software que posee el cliente o empresa. Esta información será útil al momento de estimar recursos tanto tecnológicos como humanos que serán necesarios para la ejecución del proyecto.

a. Plataformas

Establecer los sistemas operativos y lenguajes de programación que utiliza los sistemas actuales.

Por tanto se llevara un registro que será expuesto a continuación:

Tabla No.3.1. Registro de Sistemas Operativos existentes

EPP01

Sistema Operativo	No. máquinas

Tabla No.3.2. Registro de herramientas de software existentes

EPP02

Herramientas de programación	No. máquinas

b. Base de Datos

Determinar las base de datos con las que cuenta el cliente y/o que interactúan con los sistemas existentes.

Para su control se deberá llenar la siguiente información:

Tabla 3.3. Registro de Bases de datos

EPP03

Base de Datos	Versión	Tipo De Licencias

El tipo de licencia se refiere al modo de utilización de la base de datos con la que cuenta el cliente (Servidor, Cliente, entre otras).

c. Software propios

Identificar que módulos, programas o sistemas con los que cuenta la organización.

Se debe registrar estos sistemas de la manera indicada en la Tabla 3.3.

Tabla 3.4 Registro de sistemas propios

EPP04

Sistema	Módulos	Base de datos	Módulos relacionados

II. Hardware existente

Permite determinar los recursos de hardware con los que cuenta la empresa, ya sea que interactúen o no con el sistema actual.

El objetivo de detallar el hardware con el que cuenta el cliente es el de considerar estas existencias en la estimación de recursos (Capitulo 3, Sección 3.2.1.3, Pág. 183)

a. Características de PC's

Todos los computadores con los que cuenta el cliente serán detallados a fin de establecer un inventario de características de las mismas.

Los computadores deberán registrarse según el formato de la Tabla 3.5.

Tabla 3.5. Características de computadores.

EPP05

No. Serie	Disco Duro		Procesador		Memoria RAM		Puertos	Velocidad de tarjeta Madre
	Cantidad	Capacidad	Velocidad	Fabricante	Capacidad	Ranuras		

b. Hardware de comunicaciones

Consiste en listar todo implemento utilizado en las comunicaciones del cliente.

Debe listarse el hardware de comunicaciones de acuerdo al formato expuesto en la Tabla 3.6.

Tabla 3.6. Elementos de comunicación existentes

EPP06

No. Serie	Elemento	Función	Número puertos	
			Disponibles	Utilizados

c. Implementos especiales

Esta sección recopila todos aquellos aparatos especiales utilizados por el cliente tales como lectores de código de barras, detectores de metal, elementos de seguridad, cámaras, entre otros.

Estos elementos pueden ser utilizados por el sistema requerido para complementar su funcionamiento se fuese el caso y deben ser detallados de acuerdo al formato de la tabla 3.7.

Tabla 3.7. Elementos especiales de hardware

EPP07

No. Serie	Elemento	Puertos

III. Comunicaciones existentes

Esta sección permitirá establecer una guía sobre la utilización de los canales de comunicación existentes y la posibilidad de utilización de las mismas.

a. Tipo de comunicación

Se listan los tipos de comunicaciones existentes – Radio, satélite, Frame Relay, entre otras – en la infraestructura del cliente, las mismas que deben ser detalladas de acuerdo al formato de la tabla 3.8.

Tabla 3.8. Tipo de comunicación

EPP08

Tipo de comunicación	Medio de comunicación	Medio de transmisión	Velocidad	Tamaño del canal

b. Monitoreo de comunicaciones

Consiste en describir los resultados del monitoreo de comunicación, frecuencia de utilización, ancho de banda consumido, entre otros con la finalidad de determinar disponibilidad de recursos de comunicación al momento de estimar recursos.

Esta información deberá ser detallada de acuerdo al formato expuesto en la Tabla 3.9.

Tabla 3.9 Utilización de recursos de comunicación.

EPP09			
Tipo de comunicación	**Tamaño del canal**		**Frecuencia de utilización**
	Disponible (%)	**Utilizado (%)**	

3.2.1.2. Especificación de necesidades

Esta sección consiste en determinar claramente los objetivos que persiguen el sistema o módulo a construir así como sus limitaciones. Es necesaria la inclusión de una descripción detallada de los procesos que se pretenden cubrir con el mismo.

Estas necesidades se deben detallar en la sección II y III del *Plan de Proyecto de Software* (Capitulo 3, Cuadro 3.1. Plan de Proyecto de Software. Pág. 186).

3.2.1.3. Estimación de recursos

Una vez identificado el ámbito del sistema, se debe determinar la cantidad de recursos económicos, humanos, tecnológicos, físicos y de tiempo. El objetivo de esta actividad es el de presentar una propuesta al cliente para su posterior revisión.

3.2.1.3.1. Estimación de tiempo y recursos humanos Orientada a Datos

Las técnicas clásicas de estimación para los proyectos de software resultan poco objetivas debido a que requieren información histórica de proyectos previos, pese a esto, se considera que se debe utilizar la métrica de Puntos de Función para dicha estimación, por ser la menos empírica.

A partir del cálculo de puntos de función *(Capitulo II, Sección 2.8.1.1.1., Ecuación 2.1, Pág. 105)*, se debe aplicar la Tabla 2.7. *(Capitulo II, Sección 2.8.1.1.1., Pág. 107.)*, con lo que encontrara el número de líneas de código equivalentes para que una persona realiza un Punto de Función y su respectiva equivalencia en Líneas de Código necesarias.

Utilizando la Tabla 2.8. (*Capitulo II, Sección 2.8.1.1.1., Sección Pág. 107)*, se determina el tiempo requerido para el desarrollo del sistema, así como el número de personas requeridos para el mismo.

3.2.1.3.2. Estimación de tiempo y recursos humanos Orientada a Objetos

La estimación de recursos debe ser realizada basada en el número de clases que se encuentran presentes en el sistema, para lo que se utiliza las métricas propuestas por Lorenz y Kidd Capitulo 2 *(Sección 2.8.2.2.3. Pág. 132)* en la que se indica que las clase base corresponden del 20 al 40% del total de clases. Las clases base deberán ser presentadas en el siguiente formato.

Tabla.3.10 Descripción de clases base

ER01	
Número de clase:	
Nombre de la clase:	
Descripción de la clase:	

La Estimación de tiempo requerido para el proyecto se realiza por medio de las métricas propuestas por Lorenz y Kidd *(Capitulo II, Sección 2.8.2.1.2, Ecuación 2.23, Pág. 127)* tomando como punto de partida la especificación de clases del punto anterior.

La cantidad de personal necesario para la realización del proyecto se estima a partir del tiempo total requerido para el sistema, Es importante recordar que el esfuerzo requerido por el grupo de desarrollo para realizar el proyecto es inversamente proporcional al número de personas que participan en el mismo, pero que a la vez, un tamaño exagerado del grupo de desarrollo puede reducir la productividad del mismo.

La productividad del grupo de trabajo es directamente proporcional al número de personas requeridas para el sistema, es decir que al aumentar el número de personas aumenta la productividad, y es inversamente proporcional al esfuerzo requerido para la realización del sistema.

La ecuación del tiempo total del proyecto *(Capitulo II, Sección 2.8.2.1.2, Ecuación 2.24, Pág. 127)* debe ser utilizada de manera que el número de personas utilizadas para el proyecto optimice el tiempo de desarrollo del proyecto (Fig. 3.1).

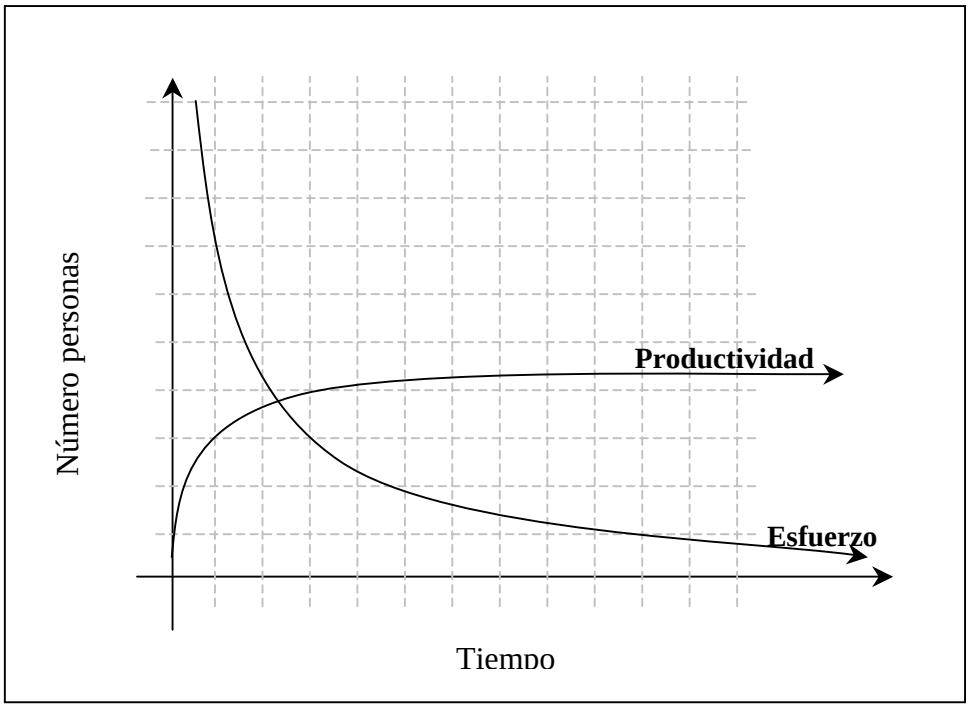


Fig. 3.1 Esfuerzo vs. Productividad para el Proyecto de Software

3.2.1.3.3. Estimación de recursos tecnológicos

Se requiere detallar los requerimientos adicionales del hardware, software y comunicaciones existentes basándose en las existencias del cliente y utilizando los formatos especificados para los mismos (Capitulo 3, Sección 3.2.1.1, Pág. 179).

3.2.1.3.4. Estimación de costos

Comprende el cálculo de gastos en el que se incurrirá al momento de desarrollar el sistema, incluyen gastos directos e indirectos, que se derivan de las estimaciones anteriores.

3.2.1.4. Planificación

Consiste en realizar un cronograma de las tareas a ser realizadas en función del tiempo, para lo cual se debe utilizar las siguientes técnicas:

- Diagramas de Gantt
- Diagramas de Pert

Además se debe realizar una distribución de las tareas al grupo de trabajo que esta encargado de desarrollar el proyecto.

1. **Plan de distribución de tareas**

   Consiste en desarrollar un perfil para cada persona que integrará el grupo de trabajo para asignar las tareas respectivas a cada perfil.

2. **Cronograma de actividades.**

   El cronograma de actividades debe ser presentada en forma de un diagrama de Pert o Gantt, en el que se debe especificar, tiempo, actividades y responsables.

3.2.1.5. Documentación a entregar

A lo largo de las actividades mencionadas anteriormente, se genera información relevante para el proyecto de desarrollo, y que será la base para el inicio de las etapas de la ingeniería del software. El documento que engloba toda la información requerida es el Plan de Proyecto de Software. El documento debe presentar el siguiente formato.

Cuadro 3.1. Plan de proyecto de software.

PLAN DEL PROYECTO DE SOFTWARE

I. Análisis previo de proyecto de software(Sección 3.2.1.1)
II. Objetivo del proyecto de software
III. Dominio proyecto de software.
    a. Alcance del Proyecto de software.
    b. Limitaciones del proyecto de software
IV. Estimaciones.
    1. Estimación de Tiempo y Recurso Humano.
    2. Estimación de Recursos Tecnológicos.
    3. Estimación de Costos.
V. Plan de distribución de tareas
    1. Perfil profesional del grupo de trabajo
    2. Tareas por perfil.
VI. Cronograma de actividades

3.2.2. Plan de aceptación del proyecto de software.

A lo largo del proyecto de software, es necesario determinar con el cliente la concordancia de los requerimientos del sistema.

Las aceptaciones del cliente son documentos que deben ser entregados al cliente cada vez que se termina algún producto en las fases de la ingeniería de software.

3.2.2.1. Formato del documento de aceptación

Este documento deberá ser preparado por el jefe de proyecto y presentado al cliente con el respectivo producto.

Tabla 3.11. Documento de aceptación de producto

PAPS01
Fase de ingeniería de software
Producto
Responsable
Fecha
Descripción
Versión
Responsable
Firma de aceptación

3.2.2.2. Documento de salida

El documento de salida de esta actividad consiste en la recopilación de los documentos de aceptación de cada producto presentado al cliente, por lo que solamente contara con los documentos especificados anteriormente organizados de la siguiente manera:

Cuadro 3.2. Formato de documentos de aceptación

Documentos de aceptación
1. Nombre del proyecto 2. Nombre del responsable 3. Documentos de aceptación

3.3. Área de Calidad

3.3.1. Plan de calidad del proyecto de software

3.3.1.1. Estándares utilizados

En esta sección se debe listar los estándares que se utilizarán en el proyecto de software. Para determinar los mismos, se debe realizar reuniones con el cliente y el desarrollador para llegar a un acuerdo sobre la utilización de estándares.

1. **Estándares Internacionales:** El plan de calidad del proyecto debe contar con una descripción de los estándares internacionales (ISO,

IEEE, entre otros), con los que debe estar desarrollado dicho proyecto.

Esto permite definir las condiciones bajo las que el proyecto de software será revisado o evaluado. No es un requisito obligatorio.

2. **Estándares propios:** Consiste en el detalle de los estándares que requiere el cliente, pueden ser referentes a formato de interfaces de usuario, nombres para programas o rutinas, componentes, formatos de entrada o salidas del sistema, entre otros.

   Deben ser detallados de acuerdo a la siguiente tabla:

Tabla 3.11 Descripción de estándares propios

PCPS01	
Ámbito	Detalle del ámbito de aplicación del estándar (formato de entrada / salida, formato de interfaces, entre otros)
Sección	Determinación de la sección específica del ámbito en el que se aplica (Cabecera, nomenclatura, sintaxis, entre otros)
Formato	Descripción del formato que se debe aplicar en cada estándar.

Si la organización no cuenta con estándares propios, se sugiere algunos en el *Anexo K*.

3.3.1.2. Control de calidad de software

Dentro de la gestión de calidad del software se debe desarrollar un plan de contingencia que prevenga cualquier anomalía en el desarrollo del software.

En el desarrollo de las actividades mencionadas anteriormente se produce información relevante la misma que se basa en la prevención que cualquier dificultas que se pueda presentar en el la etapa de gestión.

La técnica de control de calidad debe ser escogida de acuerdo a la descripción que se hace referencia en el (capitulo 2 apartado 2.9.2).

Tabla 3.12. Tabla del método utilizado.

PCPS02

Método	Etapas	Justificación

En la técnica se documenta en la tabla 3.11, cual es la técnica a utilizar y su respectiva justificación del porque se ha escogido dicha técnica y además que etapas intervienen dentro de la misma.

3.3.1.3. Documentación a entregar

La planificación de la calidad recogerá varios temas relevantes. Se debe presentar un documento que contenga el plan de calidad del proyecto de software, el mismo que debe ser desarrollado con la estructura presentada a continuación.

Cuadro 3.3. Plan de calidad del proyecto de software

Plan de Calidad del Proyecto de Software

1. Objetivos del plan de calidad del proyecto de software
2. Alcance del plan de calidad del proyecto de software
3. Estándares.
   a.Estándares internacionales.
   b.Estándares propios.
4. Control de la calidad de software (Capitulo 3, Sección 3.3.2)

3.3.2. Gestión de configuración de software

Consiste en determinar los recursos necesarios para la configuración del software y las actividades que se realizan en la misma. El propósito de la configuración de software es establecer y mantener la integridad de los

productos de software a lo largo del ciclo de vida del proyecto de software[56].

3.3.2.1. Determinación de elementos de configuración

Consiste en registrar los elementos susceptibles a modificaciones (Objetos simples y objetos compuestos)[57]. De acuerdo a los formatos especificados en las Tablas 3.13 y 3.14.

Tabla 3.13. Objetos simples; Elementos de Configuración de Software

GCS01	
Nombre del objeto:	
Número del Objeto:	Número único de identificación del objeto.
Atributos del Objeto:	Listado de atributos.
Versión:	OS.«No.Objeto».«No. Secuencial»

Tabla 3.14 Objetos compuestos

GCS02		
Nombre del objeto:		
Número del Objeto:	Número único de identificación del objeto.	
Versión:	OC.«No.Objeto».«No. Secuencial»	
Objetos Simples:	**Nombre Objetos simple**	**Versión**

3.3.2.2. Control de configuración

Consiste en la realización de registros de cambios realizados a una versión de los objetos definidos anteriormente (Tabla 3.13). Además se registra los resultados de las métricas aplicadas en cada fase de la ingeniería de software (Tabla 3.15), debido a que los cambio pueden ocurrir al no cumplirse con las métricas de calidad especificadas en cada fase.

[56] Anexo C. Practica clave "Administración de la configuración del software".
[57] Capitulo 2, Sección 2.7.22, Pág. 100.

Tabla 3.15 Reporte de resultado de métricas

GCS03					
FASE					
Fecha	**Evaluador**	**Nombre métrica**	**Resultado**		**Observación**
			Valor	**Ponderación**	
				Observación General	

En la tabla 3.16. Se debe especificar:

- Fase: Fase en la que se aplica la métrica
- Fecha: Fecha en la que se realiza la medición.
- Evaluador: Nombre de la persona que realiza la medición.
- Nombre métrica: Nombre de la métrica utilizada.
- Resultado: Indicador obtenido de la métrica.
    1. Valor: Se refiere al valor numérico obtenido en la métrica
    2. Ponderación: Consiste en asignar una calificación al valor obtenido por la métrica, se utilizara la siguiente escala[58]:
        a. No alcanzado(porcentajes)
        b. Parcialmente alcanzado
        c. Ampliamente alcanzado
        d. Totalmente alcanzado
- Interpretación: Interpretación técnica del valor obtenido.
- Observación: Conclusión de la interpretación, se determina la aceptación del indicador.
- Observación General: Indica la aprobación de las métricas evaluadas en la fase de la ingeniería. Esta observación determina

[58] Escala propuesta por SPICE para la evaluación de atributos. Capitulo II, Sección 2.5.3.2, Pág. 79.

si algún proceso dentro de cada fase debe ser repetido y las justificaciones de dicha decisión.

Tabla 3.16. Control de configuración

GCS04	
Nombre del objeto:	
Versión:	OS.«No.Objeto».«No. Secuencial»
Versión anterior:	OS.«No.Objeto».«No. Secuencial»
Descripción de cambio:	
Razón del cambio:	

La razón del cambio debe indicar si el cambio se realizo debido a fallas durante el desarrollo de la fase o si se debió a la decisión del evaluador de las métricas de cada fase.

3.3.2.3. Control de versiones

Consiste en registrar todos los cambios efectuados a las versiones a lo largo del proyecto. Se utilizara el siguiente formato:

Tabla 3.17. Control de versiones

GCS05			
Nombre del objeto	**Fecha de creación**	**Fecha de cambio**	**Responsable**

3.3.2.4. Documentos a entregar

Se debe entregar un reporte de configuración de software, el mismo que debe contener la información de todos los cambios realizados a lo largo del proceso de la Ingeniería de software.

El formato del documento de reporte de configuración de Software se presenta a continuación:

Cuadro 3.4. **Reporte de Configuración de Software**

Reporte de Configuración de Software
1. Registro de Objetos a. Objetos simples b. Objetos compuestos 2. Registro de control de configuración a. Reportes de resultados de métricas b. Reportes de control de configuración 3. Registro de control de versión

3.4. Área de Ingeniería

El propósito de esta área es el de describir los procesos involucrados en el proceso de vida del proceso de software, así como la manera de realizar mediciones cuantitativas[59] de las mismas

3.4.1. Fase de Análisis de requisitos de la Ingeniería de Software.

En la fase análisis se realizan las actividades de recolección de información con la finalidad de que el ingeniero de software defina con certeza la naturaleza del software, así como la función requerida, comportamiento, rendimiento e interconexión.

3.4.1.1. Documentación de entrada

La documentación requerida antes de iniciar la fase de análisis de la ingeniería de software es el Plan del Proyecto de Software (Capitulo 3, Tabla 3.10) en el que se identifican los objetivos del proyecto, su alcance, entre otros.

3.4.1.2. Levantamiento de información

Se debe realizar el levantamiento de información a través de encuestas cuando el universo de personas es demasiado grande. Se pude optar

[59] Anexo C . *Practica clave nivel 4* Administración cuantitativa del proceso .

por entrevistas cuando las opiniones o criterios de las personas fuente de información sea valiosa para el levantamiento de información[60] *(ver Anexo D)* las mismas que deben ser realizadas con el personal encargado del manejo de los procesos que se requiere automatizar y aquellos relacionados con los mismos.

3.4.1.3. Análisis preliminar

Este punto se enfoca hacia el entendimiento del manejo del negocio en la organización de manera que se aclaren los puntos que el sistema a construir abarcara y aquellos que quedaran al margen.

1. **Estudio de situación actual**

   Este estudio comprende una descripción de los procesos envueltos en el ámbito del sistema a construir definiendo pasa a paso las actividades realizadas.

2. **Mapa de procesos**

   El mapa de procesos debe especificar los procesos con los que cuenta una organización, determinar el ámbito operacional en el que se desenvuelven y las actividades intermedias que se realizan así como las actividades que pueden ser automatizadas.

   El mapa de procesos debe ser realizado con el formato de la Tabla. 3.3 En la que «SISTEMA», es el nombre del sistema general que se pretende desarrollar, «SUB SISTEMA» son las áreas de negocios con los que cuenta la organización, «PROCESO» son los procesos involucrados en cada subsistema, «PROCEDIMIENTOS»

[60] Prof. Leyva Zea, Francisco. NOCIONES DE METODOLOGIA DE INVESTIGACION CIENTIFICA, cuarta edición 1996, capitulo V, Pág. 39

relacionados con los procesos y al final las «ACTIVIDADES» que se realizan en cada procedimiento.

Tabla 3.18. Mapa de procesos.

ARIS01

Sistema	Sub Sistema	Proceso	Procedimiento	Actividad	Automatizable
Consulta externa	**1. Pediátrico**	**1.1.** Registro	**1.1.1.** Ingreso	**1.1.1.1.** Historia clínica	✓
				1.1.1.2. Asignación de medico	✓
			1.1.2 Cita	**1.1.2.1.** Asignación de día	✓
				1.1.2.2. Asignación de consultorio	✓
		1.2. Atención	**1.2.1.** Revisiones	**1.2.1.1.** Revisión de síntomas.	×
				1.2.1.2. Identificación problema	×
			1.2.2. Diagnóstico	**1.2.2.1.** Definir medicación	×
				1.2.2.2. Registro en historia clínica.	✓

3.4.1.4. Especificación de requerimientos de software

El documento de especificación de requerimientos debe basarse en el Estándar IEEE 830, en la sección 3.1 (Requisitos Funcionales) de este documento *(ver Anexo A)*.

3.4.1.5. Realización de Diagramas

a. Modelo Orientado a Datos

Se utiliza los Diagramas de Flujo de Datos (DFD) dentro de los cuales se representa la manera en la que la información se maneja a través de los procesos de la organización.

La representación de los DFD se realizara utilizando la simbología presentada en el *Anexo* F.

b. Modelo Orientado a Objetos

Para este modelo se realizara los diagrama de casos de uso, los que debe ser realizados en formato UML, utilizando los casos de uso definidos en las especificaciones funcionales del ERS *(ver Anexo A)*.

3.4.1.6. Documentación de salida

El documento que se debe presentar al final de esta etapa es el del Estándar IEEE 830 *(Cuadro 3.4)*, adicionalmente se debe adjuntar el Mapa de Procesos y el Diagrama de Casos de Uso o el Diagrama de Flujo de Datos en la sección 7 *(Apéndice)* del documento de Especificación de Requisitos de Software, utilizando el formato que especificado en el apartado anterior.

Cuadro 3.5. Documentación de salida Fase de Análisis

ESPECIFICACIÓN DE REQUISITOS DEL PROYECTO DE SOFTWARE

Tabla de Contenidos

1. Introducción
   - 1.1 Propósito
   - 1.2 Alcance
   - 1.3 Definiciones, siglas, y abreviaciones
   - 1.4 Referencias
   - 1.5 Apreciación global
2. Descripción global
   - 2.1 Perspectiva del producto
   - 2.2 Funciones del producto
   - 2.3 Características del usuario
   - 2.4 Restricciones
   - 2.5 Atención y dependencias
3. Los requisitos específicos (*Anexo A*)
   - 3.1. Requerimientos funcionales
   - 3.2. Requerimientos de internas externa
   - 3.3. Requerimientos de desempeño
   - 3.4. Restricciones de diseño
   - 3.5. Atributos
   - 3.6. Otros
4. Apéndices
5. Índice

3.4.1.7. Métricas a aplicar

Dentro del desarrollo de la guía es importante medir la eficiencia y la eficacia con que se ha llevado a cabo las actividades, por lo que se aplicaran ciertas métricas[61] que permiten obtener una apreciación de la calidad con que se realiza cada una de las actividades propuestas para la fase de análisis del desarrollo de software.

Se debe conformar un grupo de evaluación de los documentos de salidas de esta fase, el mismo que debe estar formado por representantes de los usuarios, del grupo de análisis y de individuos no relacionados con el sistema.

[61] Anexo A. Práctica Clave Nivel 2. *"Administración de requerimientos"*.

Los principales factores que se deben controlar en la especificación de requisitos de proyecto de software *(Cuadro. 3.5)*, son la ausencia de ambigüedad[62], compleción (qué tan completo son los requerimientos) y corrección (qué tan correcto es la interpretación de los requerimientos), esto se logra mediante la revisión del documento por parte del grupo definido.

Se parte por tanto con el número de requisitos definidos para el sistema determinada en la sección 3 del documento de especificación de requerimientos de software y aplicando la *Ecuación 2.2 (Capitulo II, Sección 2.8.1.1.3., Pág. 111)*.

La ausencia de ambigüedad se calcula por medio de la *Ecuación 2.3. (Capitulo II, Sección 2.8.1.1.3., Pág. 111*). Cuanto mas cerca esté Q de 1, menor será la ambigüedad de la especificación, por lo que se entiende que el documento esta redactado de forma clara.

La compleción se mide por medio de la *Ecuación 2.4. (Capitulo II, Sección 2.8.1.1.3., Pág. 111*), es una medida porcentual, por lo que un valor cercano al 100% es mas optimo, sin embargo, debido a que esta métrica se enfoca principalmente a los requisitos funcionales, es necesario tomar la medida de corrección (*Capitulo II, Sección 2.8.1.1.3., Ecuación 2.5, Pág.112*) con la que se constata la validez de dichos requisitos, verificándose un valor porcentual por lo que un valor mas próximo al 100% es el deseado.

[62] Ambiguo: Que puede entenderse de varios modos o admitir distintas interpretaciones y dar por consiguiente motivo a duda, incertidumbre o confusión., Diccionario Enciclopédico Siglo XXI.

3.4.2. Fase de Diseño de la Ingeniería de Software.

La fase de diseño es la parte central del proceso técnico para el desarrollo del software. El proceso de diseño consiste en realizar un bosquejo del software que se desea desarrollar, se constituye en una guía para el grupo de desarrollo para cumplir con los requisitos propuestos por la organización.

3.4.2.1. Documentación de entrada.

La documentación que será utilizada para iniciar esta parte del proceso de la ingeniería del software es el de Especificación de Requisitos de Software (Cuadro 3.2).

Es también necesario el Plan de del Proyecto de Software para la obtención de las clases definidas para el sistema (Capitulo 3, Sección 3.2.1.5, Pág. 186).

Esta documentación será la base para crear bosquejos que ayuden al grupo de desarrollo a completar el software requerido.

3.4.2.2. Selección de Ciclo de Vida del Software

Se debe seleccionar un ciclo de vida sobre el que se desarrollara el proyecto de software[63], basado en el tipo de sistema a construir.

3.4.2.3. Realización de diagramas

Los diagramas necesarios para realizar el diseño del proyecto de software dependerán del modelo utilizado.

Los diagramas pueden ser realizados con la ayuda herramientas case vigentes de acuerdo a la metodología utilizada y al tipo de sistema que se realizará[64].

[63] Para la selección de un ciclo de vida de software se recomienda la utilización del estándar de la calidad ISO 12207 *(Ver Anexo B)*

a. Modelo Orientado a Datos

Se Inicia con el Diseño de los Modelos Conceptual y Físico, que representan la perspectiva del sistema. El Modelo Conceptual representa la idea del negocio, mientras que el Modelo Físico representa la concepción informática de modelo conceptual.

Además se debe diseñar el Diagrama de Entidad Relación que presenta la relación entre cada entidad que conforman los módulos del sistema.

Este diagrama se debe encontrar normalizado y de acuerdo a los estándares de la organización. Si la organización no cuneta con un estándar para este diagrama, se debe utilizar el estándar propuesto en el *Anexo G* de esta guía.

b. Modelo Orientado a Objetos

Los bosquejos del sistema que se realiza se representan por medio de los diagramas propuestos por el UML[65].

1. Diagrama de clases

El punto de partida de esta actividad es la determinación de clases en el Plan de Proyecto de Software (Capitulo 3, Sección 3.2.1.5, Pág.186).

2. Diagrama de interacción

Los diagramas definidos para la interacción por UML son los de colaboración y secuencia.

---

[64] Para de selección de una herramienta CASE, se recomienda la utilización de el estándar de calidad ISO/IEC 14102 *(Ver Anexo B)*

[65] UML: Lenguaje Unificado de Modelamiento. Lenguaje impulsado por la corporación Racional con la finalidad de normar el modelado de información en Orientación a Objetos. Se genera a en conjunto por Jacobson, Roombo y

i. **Diagrama de colaboración**

Consiste en mostrar la relación entre objetos entrelazados para la realización de un caso e uso (Capítulo 3, Sección 3.4.1.5, Pág. 193).

ii. **Diagrama de secuencia**

Este diagrama pretende mostrar las colaboraciones entra objetos y actores a lo largo de la vida del objeto.

3. **Diagrama de estados**

Este diagrama muestra la manera en la que una clase reacción ante una clase.

4. **Diagrama de Actividad**

Este diagrama es utilizado básicamente para casos más complejos puesto que muestra la relación entre clases y casos de uso.

5. **Diagrama de implementación**

La función principal de estos diagramas es la de mostrar la arquitectura de la aplicación, es decir la interacción de los componentes de hardware y software.

i. **Diagrama de componentes**

Parte de implementación de un sistema que se puede distribuir tanto a nivel de código de software como de hardware. Este diagrama puede ser omitido si no se utiliza ningún componente.

ii. **Diagrama de despliegue**

Muestra los enlaces de comunicación física entre elementos hardware y las relacione entre maquinas físicas y procesos. Si el

sistema se ejecutará en un solo computador o en una sola capa, este diagrama puede ser omitido.

3.4.2.4. Especificación de Interfaz Gráfica de Usuario.

Es la manera de unificar cada una de las pantallas y cuadros de texto mensajes que se vayan desplegando a lo largo del proyecto. Estas deben ajustarse a los estándares del cliente en caso de contar con ellos. Si la organización no cuenta con un estándar para el diseño de interfaces, se debe generar uno apoyándose en la Guía de GUI *(ver Anexo H)*

3.4.2.5. Documentación de salida

Se deben entregar los diagramas explicados en al sección 3.4.2.2., considerando los formatos propuestos en a continuación:

Cuadro 3.6. Guía de Diseño del proyecto de software[66]

GUIA DE DISEÑO DEL PROYECTO DE SOFTWARE
I. Ciclo de vida. **II. Descripción de Entidades** **III. Descripción de capas** a. Descripción de componentes b. Descripción de interfaces **IV. Diagramas** **V. Estándar de GUI**

3.4.2.6. Métricas a aplicar

1. Metodología Orientada a datos

i. La complejidad estructural

Mientras el valor de la complejidad estructural *(Capitulo II, Sección. 2.8.1.2.1., Ecuación 2.6, Pág. 111)* aumenta, aumenta la complejidad del diseño del sistema, esto implica que el esfuerzo requerido para el

[66] *ver Anexo E*

desarrollo de la aplicación aumenta al igual que para su mantenimiento.

Un valor elevado refleja un diseño pobre que se presenta como dificultades en el desarrollo y mantenimiento, debido a que indica que los módulos tienden a centralizar las funciones del sistema, es decir mide la dispersión de los módulos.

ii. La complejidad de datos

Representa la cantidad de información procesada por el módulo *(Capitulo II, Sección. 2.8.1.2.1., Ecuación 2.7, Pág. 112)* y es proporcional a la complejidad del sistema. Mientras mayor sea el valor de la complejidad de datos, el módulo se vuelve mas complicado de manejar y de mantener. Es recomendable mantener niveles bajos en esta métrica, ya que al ser mayor el número de variables de entrada y salida, implica que el módulo no esta concebido de manera correcta.

iii. La complejidad del sistema

Se define como la suma de las complejidades estructural y de datos *(Capitulo II, Sección 2.8.1.2.1., Ecuación 2.8, Pág. 112)*, a medida que aumentan los valores de complejidad, lo que indica una mayor probabilidad de que aumente el esfuerzo para la integración y las pruebas.

iv. Acoplamiento del sistema

Se define como un indicador de conectividad entre módulos *(Capitulo II, Sección 2.8.1.2.2., Ecuación 2.10, Pág. 115)*, se recomienda evitar valores próximos a uno, ya que estos causarían un bajo

acoplamiento entre módulos, aumentando la complejidad del sistema.

2. Metodología Orientada a Objetos

i. Factor de acoplamiento

Acoplamiento implica la existencia de alguna relación ente clases, por lo que un gran acoplamiento indica que un cambio podría propagarse por toda la arquitectura del sistema. Al incrementar el acoplamiento entre clases *(Capitulo II, Sección 2.8.2.2.1, Ecuación 2.25, Pág. 130)*, se incrementa la densidad de defectos y disminuye la facilidad de mantenimiento.

ii. Factor de herencia de métodos (FHM)

Este factor se constituye en un indicador del grado en que la arquitectura de clases hace uso de la herencia para métodos y atributos y esta dado por la *ecuación 2.26 (*Capitulo II, *Sección 2.8.1.2., Pág. 130)*. No es recomendable contar con un Factor de Herencia alto por que reflejaría una gran dependencia (acoplamiento), mientras que una ausencia (valor bajo) de herencia puede reflejarse como falta de abstracción en la definición de las clases.

3. Métricas para Interfaces Gráficas de Usuario (IGU)

Una IGU (Interfaz Gráfica de Usuario) típica usa entidades de representación, iconos gráficos, texto, menús, ventanas y otras para ayudar al usuario a completar tareas.

Se evalúa las posiciones absolutas y relativas de cada entidad de representación, la frecuencia con que se utilizan y el costo de la

transición (Capitulo II, *sección 2.8.1.2.4.*, *Ecuación 2.11, Pág. 112*) de una entidad de representación a la siguiente y el grado de facilidad de utilización del software *(Capitulo II, sección 2.8.1.2.4., Ecuación 2.12, Pág. 113).*

3.4.3. Fase de desarrollo de la Ingeniería de software

En la fase de desarrollo se realiza la codificación del sistema y sus componentes, en otras palabras de efectúa el sistema como tal

3.4.3.1. Documentación de entrada

La documentación necesaria para iniciar esta fase es la guía de diseño del proyecto de software (Cuadro 3.2).

3.4.3.2. Codificación

1. **Interfases**

   Consiste en diseñar cada una de las interfases para el sistema (interfases lógicas e interfases gráficas de usuario), basado en los estándares propios del usuario en caso de existirlos.

2. **Componentes**

   Consiste en realizar la programación y documentación de cada componente en los módulos del sistema.

3. **Bases de Datos**

   Consiste en la construcción de la base de datos basado en el diagrama entidad relación, además de debe documentar el diccionario de datos de la base de datos.

3.4.4.1. Documentación de salida

Los documentos a entregar entre esta fase son

- Manual técnico *(ver Anexo I)*
- Diccionario de Datos *(ver Anexo I)*

3.4.3.3. Métricas a aplicar

Las métricas a aplicar variaran de acuerdo a la metodología utilizada para el desarrollo (Orientada a Datos u Orientada a Objetos).

a. **Modelo Orientado a Datos**

Se deberá utilizar la métrica de Volumen Mínimo de Información propuesta por Halsted *(Capitulo II, Sección 2.8.1.3., Ecuación 2.15. Pág. 113)* considerando que un volumen alto de información dificultara el entendimiento, retardara el funcionamiento del sistema y afectara a la fase de mantenimiento del software. Se debe recordar que los autores recomiendan que el valor deba ser inferior a 1.

b. **Modelo Orientado a Objetos**

Para el modelo orientado a objetos se utilizara las medidas de Tamaño Medio de Operación *(Capitulo 2, Sección 2.8.2.3.1, Pág. 140)* y la de Número de Parámetros entre Operaciones *(Capitulo 2, Sección 2.8.2.3.2, Pág. 140)* las que son un indicador de comunicación entre módulos del sistema. Un valor alto en estas métricas representa una mala definición de funciones y por tanto deficiencia en el comportamiento esperado del sistema.

3.4.4. Fase de Validación y Pruebas

3.4.4.2. Documentación de entrada

Las pruebas se realizan en base a la Especificación de Requerimientos de Software debido a que es ahí donde se indica las funciones específicas del sistema y son ellas las que deben probarse.

3.4.4.3. Plan de pruebas

El objetivo de este plan la determinación del alcance recursos requeridos, calendario y responsables de la fase de pruebas.

Este plan debe contar con las siguientes secciones:

1. **identificador del Plan**

   Consiste nombrar a plan de pruebas de acuerdo al requerimiento que se desea evaluar.

2. **Alcance**

   Indica las propiedades y elementos del software a ser evaluados.

3. **Técnicas de pruebas**

   Describe las técnicas patrones a utilizarse en el diseño de casos de prueba, que se detalla en la Sección. 2.9.1.5.2.

4. **Configuración de plan de pruebas**

   Determina las condiciones bajo las cuales el plan puede ser suspendido, reiniciado, concluido. En caso de suscitarse algún defecto en la prueba se debe detallarse cuando debe suspenderse la prueba y en que punto debe reiniciar el proceso de pruebas.

5. Recursos

Especifica las propiedades necesarias para un ambiente de prueba, abarcando características de hardware, software y recursos humanos necesarios para el proceso

6. Calendario

Determina el cronograma de las actividades de la fase de pruebas que puede ser representado en un diagrama de Gantt o Pert.

7. Responsables

Determina los responsables de las tareas previstas en el plan.

3.4.4.4. Criterios de Verificación

Consiste en listar los requerimientos determinando un factor de evaluación de cada uno de ellos.

3.4.4.3.1. Determinación de casos de prueba

Se debe listar cada uno de los casos requerimientos de software como los requerimientos de casos de prueba.

3.4.4.3.2. Reporte de resultados

Se debe indicar la prueba que se realiza, junto con el responsable y el resultado de la misma de acuerdo al siguiente formato:

Tabla 3.19. Matriz de resultado de pruebas

PIS01

Fecha	Prueba	Responsable	Observación
Fecha	Nombre de la prueba que se realiza	Nombre del responsable o responsables de la prueba	Resultado de las pruebas, Número de reintentos por prueba, descripción de fallos encontrados en cada repetición, razones de suspensión, razones de aceptación.

3.4.4.5. Auditorias y verificaciones

Se debe realizar una agenda de auditorias del sistema, así como condiciones para los ambientes de verificación.

3.4.4.6. Documentación de salida

Se debe entregar el reporte de resultados de las pruebas así como un informe de auditoria de las mismas (*ver Anexo J).*

Cuadro 3.7. Reporte de resultados de pruebas

Reporte de resultados de las pruebas
1. Plan de pruebas a. identificador del Plan b. Alcance c. Técnicas de pruebas d. Configuración de plan de pruebas e. Recursos f. Calendario g. Responsables h. Casos de prueba 2. Resultado de pruebas 3. Informe de auditoria

3.4.4.7. Métricas a aplicar

Las métricas de diseño proporcionan información sobre facilidad o dificultad asociada con la prueba de integración y la necesidad de software especializado de prueba. La complejidad ciclomática se constituye en una métrica que se ajusta a la evaluación de los casos de pruebas determinados *(Capitulo III, sección 3.4.4.3.1, Pág. 206)*.

Además, la complejidad ciclomática puede usarse para elegir módulos como candidatos para pruebas más profundas. Los módulos con gran complejidad ciclomática tienen más probabilidad de tendencia a errores que los módulos con menor complejidad ciclomática, por esta razón, el analista debería invertir un esfuerzo extra para descubrir

errores en el módulo antes de integrarlo en un sistema. El esfuerzo de las pruebas también se puede estimar usando métricas obtenidas de medidas de Halstead.

Se pude utilizar tanto en objetos como en datos la misma métrica puesto que la complejidad ciclomática es una medida de la complejidad de los métodos de las clases, por tanto una clase cuyos métodos presenten una complejidad ciclomática alta se constituye en un candidato propicio para la realización de pruebas.

3.4.5. Fase de implementación

Se tratan los aspectos generales que requiere el proyecto para ser puestos en funcionamiento.

3.4.5.1. Manual de Usuario

Debe abarcar los aspectos clave para que el usuario final pueda utilizar el sistema.

3.4.5.2. Documentación de salida

El documento de salida de esta fase es el manual de usuario. (Cuadro 3.6).

Cuadro 3.8. Manual de usuario

Manual de usuario
1. Introducción del sistema 2. Descripción de los procesos del sistema 3. Manejo del sistema 4. Preguntas frecuentes 5. Recomendaciones de uso

3.4.6. Fase de mantenimiento

El mantenimiento surge por nuevos requerimientos del cliente, debido a cambio de plataformas, cambio del enfoque de negocio de la organización, entre otros.

Es básico comprender que la fase de mantenimiento esta sumamente ligada a la fase de desarrollo, ya que el mantenimiento que se le puede dar a un sistema depende de as características que se le entregue en su desarrollo.

3.4.6.1. Documentos de entrada

Especificación de nuevos Requisitos.

3.4.6.2. Documentación de salida

Documento técnico de mantenimiento, el cual contendrá (Cuadro 3.3):

1. Responsable del cambio
2. Fecha del cambio
3. Descripción del cambio

El responsable es el nombre del la persona designada para realizar el mantenimiento o el nombre del director del grupo designado para esta labor.

La descripción contendrá un detalle de todos los cambios realizados al sistema describiendo el estado inicial del elemento a modificar y el cambio que se realizará.

La fecha será la fecha en la que se entrega el cambio.

Cabe mencionar que si los cambios que se realizaron incluyen la base de datos se debe documentar la tabla la cual fue afectada o en su defecto la nueva tabla.

Cuadro 3.9. Documento de salida fase de mantenimiento

Documento técnico de mantenimiento
1. Responsable del cambio 2. Fecha del cambio 3. Descripción del cambio

3.4.6.3. Métricas a aplicar

El mantenimiento se evalúa de acuerdo al esfuerzo necesario para realizar esta actividad, resultando inversamente proporcional al esfuerzo invertido en el diseño y desarrollo; un gran esfuerzo en el diseño y el desarrollo resulta en un bajo esfuerzo al momento de realizar un mantenimiento.

Las métricas, por tanto, a utilizar en esta etapa serán las de acoplamiento utilizadas en la fase de diseño *(Capitulo III, Sección 3.4.2.5, Pág. 201).*

CAPITULO IV: APLICACIÓN DE LA GUIA A UN CASO DE ESTUDIO

4.1. Descripción del Sistema Integrado de Gestión Hospitalaria para el Hospital Pediátrico Baca Ortiz.

El sistema cuenta con los siguientes módulos:

- Seguridad
- Estadística
- Consulta externa
- Emergencias

Los mismos que permiten el ingreso y manejo de información a través de los procesos del hospital, los cuales están acorde a las normativas del Sistema Nacional de Salud.

4.2. Aplicación de la Guía para la Gestión de Calidad en las fases de Análisis y Diseño para el Sistema Integrado de Gestión Hospitalaria para el Hospital Pediátrico Baca Ortiz.

PLAN DE PROYECTO DE SOFTWARE

I. Análisis previo de proyecto de software

1. Software Existente

a. Plataformas

EPP01

Sistema Operativo	No. máquinas
Windows NT SERVER 4.0	1
Windows XP	8

EPP02

Herramientas de programación	No. máquinas
Developer	

b. Base de Datos

EPP03

Base de Datos	Versión	Tipo De Licencias
ORACLE	8	SERVIDOR
ORACLE	8	CLIENTE

c. Software propio

EPP04

Sistema	Módulos	Base de datos	Módulos relacionados
ORION	Estadistica	Oracle	0

2. Hardware existente

a. Características de PC's

EPP05

No. Serie	Disco Duro		Procesador		Memoria RAM		Puertos	Velocidad de tarjeta Madre
	Cantidad	Capacidad	Velocidad	Fabricante	Capacidad	Ranuras		
PC01	1	80 GB	2.4 GHz	Intel	256 MB	2	2	533 MHz
PC02	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz
PC03	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz
PC04	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz
PC05	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz
PC06	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz
PC07	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz
PC08	1	80 GB	2.4 GB	Intel	256 MB	2	2	533 MHz

b. Hardware de comunicaciones

EPP06

No. Serie	Elemento	Función	Número puertos	
			Disponibles	Utilizados
----------	RACK DE PISO	----------	----------	----------
----------	SWITCH	CAPA 2	0	24

II. Objetivo del proyecto de software

Automatización de todos los procesos del Hospital Pediátrico Baca Ortiz referente a los procesos de hospitalización, emergencia y consulta externa.

III. Dominio proyecto de software.

a. Alcance del Proyecto de software.

El sistema a desarrollado cumplirá con todos los procedimientos y normativas establecidas en el sistema nacional de salud del Ecuador, el cual cubre los procesos indicados anteriormente.

b. Limitaciones del proyecto de software

EL proyecto no incluye el área de "Trabajo Social", proceso dirigido a la categorización de personas con escasos recursos económicos.

IV. Estimaciones.

1. Estimación de Tiempo y Recurso Humano.

a. Listado de clases base todas las clases descritas en lugar de matriz

Para el proyecto se utilizaron las siguientes clases base, cuyo ámbito no se describe debido a que todas tienen un carácter reservado:

1. Médico
2. Turno
3. Ficha medica
4. Nota de evolución
5. Antecedentes Patológicos
6. Período natal
7. Antecedente psicológico
8. Intervención quirúrgica
9. Vacuna
10. Laboratorio de Microbiología
11. Laboratorio Clínico
12. Laboratorio General
13. Tratamiento
14. Pedidos
15. Facturación
16. Paciente
17. Odontología
18. Oftalmología
19. Proveedor
20. Cliente
21. Producto
22. Stock
23. Ingresos
24. Egresos
25. Devoluciones
26. Control previo
27. Emergencia
28. Cirugías
29. Admisión
30. Alta
31. Ventas

El sistema cuenta además con un módulo de seguridades conformado por 7 clases base, un módulo de estadística compuesta por 8 clases base y un módulo de hospitalización formada por 2 clases base, las mismas que por políticas de la institución se mantienen reservadas.

b. Número de clases soporte

$$\text{Clases soporte} = \frac{(100 - porcenaje\_clases\_clave)*\#clases\_clave}{porcentaje\_clases\_clave}$$

$$\text{Clases soporte} = \frac{(100-40)*48}{40}$$

$$\text{Clases soporte} = \frac{60*48}{40}$$

$$\text{Clases soporte} = 72\ clases$$

c. Tiempo requerido

Tiempo requerido para Clases base

$$\text{TRCB} = 10*\sum(n*f)$$

$$TRCB = 10*(20*2.5+10*2+18*3)$$

$$TRCB = 1240\ DIAS$$

Tiempo requerido para Clases soporte

$$\text{TRCS} = 3*\sum(\text{n}*\text{f})$$

$$TRCS = 3*(72*2.5)$$

$$TRCS = 540\ DIAS$$

Total de días

$$\text{TD} = \text{TRCB} + \text{TRCS}$$

$$\text{TD} = 1240 + 540$$

$$\text{TD} = 1780\ \text{DIAS}$$

$$TDP = TD / TOTAL\_PERSONAS$$

$$TDP = 1780/4$$

$$TDP = 445\ \text{DIAS}$$

$$\text{TDP} = 20\ \text{MESES}$$

1. Estimación de Recursos Tecnológicos.

a. Software requerido

Plataformas

EPP01

Sistema Operativo	No. máquinas
Windows 2000 Server	1

EPP02

Herramientas de Programación	No. máquinas
Power Builder	4
Power Designer	4

Base de Datos

EPP03

Base de Datos	Versión	Tipo De Licencias
ORACLE	9 i	ENTRERPRISE

b. Hardware requerido

Características de PC's

EPP04

No. Serie	Disco Duro		Procesador		Memoria RAM		Puertos	Velocidad de tarjeta Madre
	Cantidad	Capacidad	Velocidad	Fabricante	Capacidad	Ranuras		
NO APLICA	5/36 USCI		3.2/ Xeon Intel		2 G /XX		XXX	533

Hardware de comunicaciones

EPP05

No. Serie	Elemento	Función	Número puertos	
			Disponibles	Utilizados
3	SWITCH	Capa2	4	20

c. Comunicaciones requeridas

- **Tipo de comunicación**

EPP07

Tipo de comunicación	Medio de comunicación	Medio de transmisión	Velocidad	Tamaño del canal
IP		Utp cat 6	10/100/100	

2. Estimación de Costos.

El costo del proyecto es de $ 37.754 desglosados de la siguiente manera:

	Monto
Sueldos	$ 36.400
Costos papelería	$ 364
Depreciación de equipos	$ 810
Impresora	$ 162
TOTAL	$ 37.736

V. Plan de distribución de tareas

1. Perfil profesional del grupo de trabajo

El grupo de trabajo está conformado por 5 ingenieros de software, distribuidos de la siguiente manera:

- 1 Gestor superior.
- 1 Gestor técnico del proyecto.
- 3 Programadores.

2. Tareas por perfil.

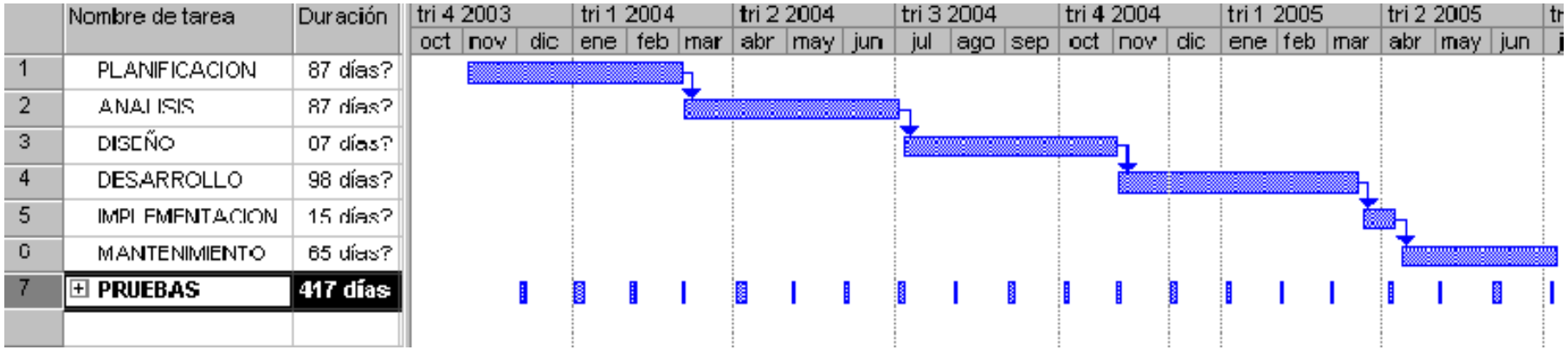
Las tareas para cada uno de los profesionales requeridos son las siguientes:

- Gestor superior: Define los aspectos del negocio que a menudo tienen una significativa influencia en el proyecto.

- Gestor técnico del proyecto: Encargado de planificar, motivar, organizar y controlar a los profesionales que desarrollan el producto de software.
- Programadores: Profesionales que proporcionan las capacidades técnicas necesarias para la ingeniería de un producto o aplicación.

V. Cronograma de actividades

De acuerdo a lo propuesto por QFD, el tiempo requerido para las fases de análisis y diseño del proyecto de software corresponde al 60% del tiempo total estimado para el proyecto.



Plan de Calidad del Proyecto de Software

1. **Objetivos del plan de calidad del proyecto de software**

   Garantizar la utilización de hitos mensurables de calidad y procedimientos escritos, que se encuentren de acuerdo a normas nacionales e internacionales, de obligatorio cumplimiento que involucrará a todo el personal, quienes serán responsables por la excelencia de su trabajo y el mejoramiento continuo del mismo.

2. **Alcance del plan de calidad del proyecto de software**

   El Plan de Calidad del Proyecto de Software constituye parte de la política de calidad de la empresa, por lo que es utilizado por para todos sus proyectos de software.

3. **Estándares.**

   a. **Estándares internacionales.**

   - Se utilizara el estándar de especificaciones de requerimientos de software IEEE 830.
   - Se utilizara el estándar para código de programación propuesto por Power Builder.

   b. **Estándares propios.**

PCPS01	
Ámbito	Interfaz Grafica de Usuario
Sección	Sintaxis y presentación
Formato	Las ventanas deben ser separadas por colores de acuerdo al área a la que pertenece la información presentada o requerida.

PCPS01	
Ámbito	Interfaz Grafica de Usuario
Sección	Sintaxis y presentación
Formato	Los elementos presentados en las ventanas deberá ser diseñados de forma que al presentarse un cambio de estado de los mismos, este sea advertido por el usuario.

PCPS01	
Ámbito	Interfaz Grafica de Usuario
Sección	Formato Ventana de mantenimiento
Formato	

1. Título de la ventana: Tipo oración
3. Posición: X = 0 , Y = 0, Maximizado
4. Estilo de presentación: Grid
5. Etiquetas:
   - Letra: 9, Arial, Bold, Navy
   - Alineación: Centrado
   - Fondo: Transparente
   - Borde: 3D Raised
   - Posición: Y = 8, Alto = 60
6. Campos:
   - Letra: 9, Arial, Regular, Black
   - Fondo: Blanco
   - Borde: Ninguno
   - Posición: Y= 0, Alto = 60
7. Datawindow:
   - Color: ButtonFace
   - Borde: Ninguno
   - Posición: X = 0, Y = 0 en la ventana.
   - Cabecera:
   - Alto: 72
   - Color: Transparente
8. Detalle:
   - Alto: 64
   - Color: Transparente

PCPS01	
Ámbito	Base de datos
Sección	Tablas de la base de datos
Formato	

Escriba en singular, todo en minúscula, utilice hasta 30 caracteres
Nombre de entidades: Nombre representativo de la tabla
Utilice el siguiente estándar.

mnnn_mmm... donde:

m	Es la letra del Código del subsistema , ejm.
p	
nnn	Identificador del objeto (pasa al atributo)
mmm	Nombre del objeto

Tablas temporales:

tmp_ nnnnnnn.... donde:

- tmp Prefijo que identifica tabla temporal
- nnnnn.. Nombre de la tabla.

PCPS01	
Ámbito	Interfaz Grafica de Usuario
Sección	Formato Ventana de Maestro Detalle
Formato	

1. *Título de la ventana:* Tipo oración
2. *Posición:* X = 0 , Y = 0, Maximizado
3. *Maestro:*
   - Estilo de Presentación: FreeForm
4. *Etiquetas:*
   - Letra: 9, Arial, Bold, Navy
   - Alineación: Derecha
   - Fondo: Transparente
   - Borde: Ninguno
5. *Campos:*
   - Letra: 9, Arial, Regular, Black
   - Fondo: Blanco
   - Borde 3D Lowered
6. *Detalle:*
   - Estilo de Presentación: Tabular
   - Etiquetas: Letra: 9, Arial, Bold, Navy
   - Alineación: Centrado
   - Fondo: Transparente
   - Borde: Ninguno
7. *Campos:*
   - Letra 9, Arial, Regular, Black
   - Fondo Blanco
   - Borde 3D Lowered
8. *Datawindow:*
   - Color: ButtonFace
   - Borde: 3D Raised
   - Cabecera: Alto: 76
   - Color: Transparente
   - Detalle: Alto: 84
   - Color: Transparente

PCPS01	
Ámbito	Interfaz Grafica de Usuario
Sección	Formato Ventana de DropDown
Formato	

1. Sin Etiquetas
2. Posición del primer campo: X = 5 , Y = 0
3. Alto de los campos 60
4. Sin separación entre campos
5. Estilo de presentación: Tabular
6. Borde: Ninguno
7. Campos: 9, Arial, Regular, Black
8. Fondo: Blanco
9. Borde: Ninguno
10. Datawindow: Fondo Blanco
11. Ordenar el datawindow
12. Alto de la sección de detalle: 64

PCPS01	
Ámbito	Base de datos
Sección	Convenciones Generales
Formato	

Se ha considerado conveniente adoptar las siguientes convenciones para identificar los estándares de base de datos.

- **( )** = Identifica la constante de la categoría del atributo o del objeto a definir en la base de datos.
- **{ }** = Indica que se debe escoger al menos una de las opciones mostradas dentro de las llaves.
- **[ ]** = Es opcional y se puede escoger una o más de las opciones mostradas dentro de los corchetes.
- _ = Underscore, usado para separar los elementos identificados como categorías, tipos y opciones de objetos, en este manual. No se usará como separador en cadenas de caracteres correspondientes a nombres de datos definidos por el usuario.
- **.....** = Indica que el nombre del elemento puede variar en longitud.
- **nnn** = Sustituciones, representado por caracteres itálicos indicando que el elemento está compuesto por caracteres provistos por el usuario.
- **kkk** = Constantes representadas por caracteres tipo romano.
- Se usará el término de objeto para referenciar a los distintos componentes de una base de datos.

PCPS01	
Ámbito	Base de datos
Sección	Objetos de la base de datos
Formato	

Las identificaciones de los objetos en la base de datos se deberá sujetar al siguiente esquema:

(nnn)_{ x, y / z}_{ mmmmmmm.....}

⇓ ⇓ ⇓

Nombre asignado *Opciones* *Categoría*

Los parámetros opcionales pueden estar separados entre comas (,) o barras (|), de acuerdo a la siguiente convención:

- Separados con comas, cuando se puede seleccionar uno o más parámetros
- Separados con barras, cuando se debe seleccionar uno de ellos únicamente.

PCPS01

Ámbito	Base de datos
Sección	Servidores de base de datos

Formato

nnn.._uu_yyy donde:

- *Nnn:* Identificador de las siglas del reparto
- *uu:* Número del servidor
- *yyy:* Identifica el tipo de servidor

dat = Servidor de datos
bac = Servidor de backup
com = Servidor de comunicación, etc.

En el caso de existir más servidores, abrevie con los 3 primeros dígitos de su tipo de servidor.

Ejemplo : disift_01_dat

PCPS01

Ámbito	Base de datos
Sección	Clientes de base de datos

Formato

nnn.._mmm_uu donde:

- nnn Identificador del reparto
- mmm Nombre del área de trabajo
- uu Número de estación

Ejemplo : disift_des__01

PCPS01

Ámbito	Base de datos
Sección	Usuarios de base de datos

Formato

n{n}mmmm..... donde:

- n Es la letra del primer nombre del usuario.
- {n} Es la primera o más letras del segundo nombre del usuario, para evitar duplicados.
- mmm Es el apellido del usuario

Ejemplo : pamolina
Smarevalo

Para grupos de usuarios: El nombre del grupo será de acuerdo a la función a definirse
mmmmm..... donde:

mmm.. Palabras descriptivas y significativas

Ejemplo : administrativo
logístico

PCPS01	
Ámbito	Base de datos
Sección	base de datos
Formato	
mmmm donde: mmmm Nombre descriptivo de la base de datos	

PCPS01	
Ámbito	Base de datos
Sección	Esquemas
Formato	
nn_mmmm....._ddd donde: nn Mnemónico del sistema mmmm.... Nombre descriptivo del subsistema ddd Indica el uso de la base de datos. des desarrollo pru pruebas pro producción ent entrenamiento	

PCPS01	
Ámbito	Base de datos
Sección	Tablas de la base de datos
Formato	
n(r)_nnnn_mmmm.. donde: n Es la letra del Código del Subsistema (r) Letra que identifica que es una relación nnn Identifica entidad padre mmm Identifica entidad dependiente Se puede utilizar agrupaciones de caracteres para lo cual se usará el UNDERSCORE (_) como separador. Además dentro de la relación documente lo siguiente: Label: describa la relación existente entre las entidades. Role: describa utilizando verbos de la siguiente manera: Emite Tiene Corresponde Pertenece, etc.	

PCPS01	
Ámbito	Base de datos
Sección	Atributos de la base de datos
Formato	
nnn_mmmm..... donde: - nnn Identifica los 3 primeros dígitos del nombre del identificador del objeto a la cual pertenece el atributo. - mmmm... Es el nombre asignado por el usuario para agrupaciones de caracteres se usará el UNDERSCORE como separador. El nombre deberá tener significado para el usuario.	

Ejemplos : pro_codigo

PCPS01	
Ámbito	Base de datos
Sección	Constrains de la base de datos
Formato	

nnnn...._ { pk | fk }. . donde:

- nnnn..... Nombre del dato sobre el cual se aplica el constraint.
- { pk | fk} Representa el tipo de constraint:
- El formato para un constraints es el siguiente:
- pk = de clave primaria
- fk = de clave foránea.

Ejemplo : pro_codigo_pk

PCPS01	
Ámbito	Base de datos
Sección	Indices de la base de datos
Formato	

Para índices que no han sido generados automáticamente por POWER DESIGNER

idx_ { u | c | nc} _nnnnn..... donde:

- idx Identificación el índice
- u Identificación de índice único (unique)
- c Indica que es un índice clusterizado (cluster)
- d Indica que es un índice duplicado (duplique)
- nnnn..... Nombre del atributo

Ejemplo : idx_c_cod_cta_cte.

PCPS01	
Ámbito	Base de datos
Sección	Procedimientos almacenados de la base de datos
Formato	

sp_mmm.... donde:

- sp Constante que indica Store Procedure
- mmmn..... Es el nombre del stored procedure

Se puede utilizar agrupaciones de caracteres para lo cual se usará el UNDERSCORE (_) como separador.

Ejemplo : rsp_dig_verf

Todo stored procedure deberá tener un encabezado constituido por un conjunto de comentarios que deben ir al inicio del programa y que debe incluir la siguiente información: archivo, stored procedure, base de datos, texto indicativo del producto, nombre del programador, fecha de codificación, descripción del programa, fechas de modificaciones y objetivo de las mismas.

PCPS01	
Ámbito	Base de datos
Sección	Funciones almacenados de la base de datos
Formato	

f_mmm.... donde:

- f Constante que indica Función
- mmmn..... Es el nombre de la función

Se puede utilizar agrupaciones de caracteres para lo cual se usará el UNDERSCORE (_) como separador.

Ejemplo : f_dig_verf

Toda Función deberá tener un encabezado constituido por un conjunto de comentarios que deben ir al inicio del programa y que debe incluir la siguiente información: archivo, función, base de datos, texto indicativo del producto, nombre del programador, fecha de codificación, descripción del programa, fechas de modificaciones y objetivo de las mismas.

PCPS01	
Ámbito	Base de datos
Sección	Triggers almacenados de la base de datos
Formato	

tr_nnnnn....._mmmm... donde:

- tr Identificación de trigger,
- nnn... Nombre-del-trigger

Se pueden utilizar agrupaciones de caracteres para lo cual se usará el UNDERSCORE (_) como separador.

- mmm... Nombre de la tabla.

Ejemplo: tr_sueldo_logistico

PCPS01	
Ámbito	Base de datos
Sección	Procedimientos almacenados de la base de datos
Formato	

f_mmm.... donde:

- f Constante que indica Función
- mmmn..... Es el nombre de la función

Se puede utilizar agrupaciones de caracteres para lo cual se usará el UNDERSCORE (_) como separador.

Ejemplo : f_dig_verf

Toda Función deberá tener un encabezado constituido por un conjunto de comentarios que deben ir al inicio del programa y que debe incluir la siguiente información: archivo, función, base de datos, texto indicativo del producto, nombre del programador, fecha de codificación, descripción del programa, fechas de modificaciones y objetivo de las mismas.

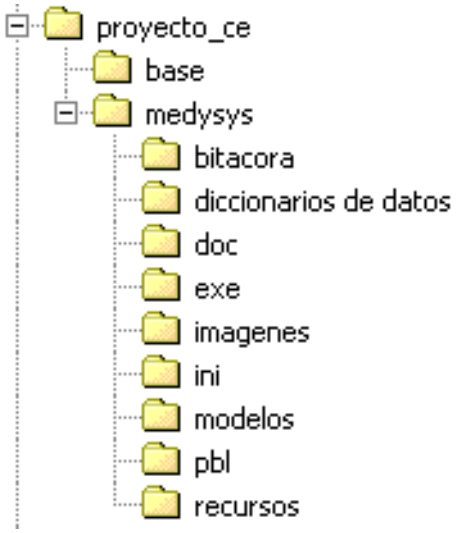
PCPS01	
Ámbito	Base de datos
Sección	Comentarios de la base de datos
Formato	
Comentarios **#nnnnnnnnnnn..... ó** **--nnnnnnnnnnn.....** Se debe crear una línea de comentario para cada sentencia o grupo de sentencias PL/SQL, que realicen un proceso o trabajo específico. También se usarán para documentar una sentencia o conjunto de sentencias SQL, o para hacer notar que se ha ejecutado una determinada acción. El formato para autodocumentar sentencias SQL, es: **Comandos SQL # objetivo de las sentencias** Ejemplo: if SQLCODE = 0 then # Verifica ejecución de la transacción	

PCPS01	
Ámbito	Base de datos
Sección	Scripts de la base de datos
Formato	
nnnnnnn..... [**.extensión**] donde: nnn..... Nombre del script, hasta 10 caracteres extensión: Identifica la extensión de la base de datos que se este utilizando Ejemplo : logistico.sql	

PCPS01	
Ámbito	Base de datos
Sección	Viatas almacenados de la base de datos
Formato	
v_nnnnnn..... donde: v identificación de vista nnn nombre de la vista Ejemplo : v_ctasctes	

PCPS01	
Ámbito	Instalación
Sección	Estructura del directorio
Formato	

La estructura de directorios para la ubicación de los productos del desarrollo es la siguiente:



C:\proyeco_ce\ medysys\pbl\proceso
C:\proyeco_ce\ medysys\exe
C:\proyeco_ce\ medysys\ini
C:\proyeco_ce\ medysys\doc {manuales, análisis, diseño, actas}
C:\proyeco_ce\ medysys\imagenes
C:\proyeco_ce\ medysys\recursos

PCPS01	
Ámbito	Código fuente
Sección	Comentarios
Formato	

Inicialmente se puede colocar comentarios en el script pero para la generación de los ejecutables se retiran todos los comentarios. Documentación interna mínima:
En la cabecera del scrip se deberá agregar la siguiente información :

- Descripción de la funcionalidad.
- Funciones que llama, variables importantes, tablas y campos que usa.
- Descripción del proceso.
- Fecha Ultima modificación
- Responsable original.
- Responsable cambios.

PCPS01	
Ámbito	Formato de librerías
Sección	Nombres de objetos
Formato	

Los nombres de objetos almacenados dentro de las librerías se escribirán en minúsculas con el siguiente formato:
VENTANA: W_NOMBRE, pero también se sugiere la siguiente codificación que puede ser utilizada por el programador a su conveniencia.

wm_nombre	Mantenimiento
wr_nombre	Reporte
wd_nombre	Maestro – detalle
wp_nombre	Principal

DATAWINDOW: D_NOMBRE, pero también se sugiere la siguiente codificación que puede ser utilizada por el programador a su conveniencia.

dm_nombre	Mantenimiento
dr_nombre	Reporte
dt_nombre	Dropdown
de_nombre	Externo
ds_nombre	Maestro

MENÚ: M_NOMBRE (se utilizará un solo menú para implementar las seguridades)

APLICACIÓN: AP_NOMBRE

FUNCIÓN: f_nombre

FUNCIÓN DE VENTANA: WF_NOMBRE
El programador puede incluir dos letras y un "_" antes del nombre del objeto para identificar de manera única el objeto.

PCPS01	
Ámbito	Levantamiento de información
Sección	Metodología.
Formato	Se realiza por medio de entrevistas encuestas y foros con el personal que realiza los procesos que se automatizan.

PCPS01

Ámbito	Código fuente
Sección	Nombres de variables

Formato

Los nombres de las variables se definen de la siguiente manera at_nombre, donde:

- *a* es el alcance de la variable como se indica en la tabla.

Alcance	**Letra**
Local	L
Instancia	I
Shared	S
Global	G
Argumento	A

- *t* es el tipo de dato de la variable como se indica en la tabla.

Tipo	Letras
Integer	I
Long	L
Double	Db
Decimal	Dc
String	S
Date	D
Datetime	Dt
Datawindow	Dw
Datawindowchild	Dwc
Time	T

PCPS01

Ámbito	Formato de librerías
Sección	Nombres de librerías

Formato

Los nombres de librerías seguirán el siguiente formato *p_pi_objeto*; donde:

pi :es una abreviación del nombre de los distintos procedimientos
objeto: define la siguiente lista de objetos:

- aplicación (aplicación, objetos de usuario, estructuras)
- datawindow (otros datawindow)
- dropdown (datawindow tipo dropdown)
- función (funciones)
- menú (menús)
- otro (otros objetos)
- reporte (datawindow tipo reporte)
- seguro (objetos del módulo de seguridad y auditoria)
- ventana (ventanas de la aplicación)

Reporte de Configuración de Software

1. Registro de control de configuración

a. Reportes de resultados de métricas

GCS03

FASE PLANEACION					
Fecha	Evaluador	Nombre métrica	Resultado		Observación
			Valor	Ponderación	
18/11/2004		Número de clases soporte	72	F	
18/11/2004		Tiempo requerido para clases base	1240 días	F	
18/11/2004		Tiempo requerido para clases soporte	540 días	F	
18/11/2004		Tiempo total (persona – mes)	1780 días	F	
18/11/2004		Tiempo total del proyecto	445	F	
		Observación General			Se ha cumplido con la planificación del proyecto

GCS03

FASE ANALISIS					
Fecha	Evaluador	Nombre métrica	Resultado		Observación
			Valor	Ponderación	
17/11/2004		Especificidad de requisitos	0.075	F	Los requisitos fueron recopilados de manera clara.
17/11/2004		Compleción	0.00057	P	La asignación de funciones para el sistema es completo.
17/11/2004		Corrección	1	F	Los requisitos funcionales has sido definidos adecuadamente.
		Observación General			Los procesos comprendidos en esta fase se han realizado de manera satisfactoria

GCS03

FASE DISEÑO					
Fecha	**Evaluador**	**Nombre métrica**	**Resultado**		**Observación**
			Valor	**Ponderación**	
18/11/2004		Factor de acoplamiento	698E-5	F	El sistema tiene un bajo acoplamiento, lo que reduce el esfuerzo necesario para el desarrollo, pruebas y mantenimiento
18/11/2004		Factor de herencia de métodos (FHM)	0.372	F	La utilización de herencia en el diseño es moderada por lo que se asegura una baja complejidad y una abstracción aceptable.
18/11/2004		Métricas para Interfaces Gráficas de Usuario (GUI)	No aplica		Las interfaces fuero desarrolladas de acuerdo a las especificaciones del cliente y basadas en ele estándar de Windows, por lo que se considera a las GUI aceptadas.
		Observación General			Los procesos comprendidos en esta fase se han realizado de manera satisfactoria

Las ponderaciones se han realizado utilizando el método propuesto por SPICE, para la evaluación de atributos**.**

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Por ser la ingeniería de software un proceso compuesto por etapas, claramente definidas, la aplicación de los métodos de gestión de calidad total son aplicables. El proceso de mejora continua es un concepto propuesto por varios autores para el proceso de software, y es justamente la manera de asegurar que cada esfuerzo por entregar calidad a un producto de software no sea aislado.
- A lo largo de las etapas de la ingeniería de software se ha propuesto una serie de controles e indicadores para medir la eficacia con la que se ha concluido una etapa, varias de las métricas recomendadas proponen distintas formas de llegar a un mismo fin y otras presentan ambigüedad en su sintaxis por lo que se dificulta el proceso de selección de el método preciso para un proyecto de software y provoca el incremento de la incertidumbre en un proceso de Gestión de Calidad para el software.
- A pesar de existir gran cantidad de métodos de gestión de calidad y estándares para calidad, la aplicación de un método o estándar no garantiza en si mismo calidad en los proyectos, sino que se debe realizar los procesos especificados en cada método o estándar concientemente para alcanzar la calidad
- La guía propuesta para la Gestión de la Calida de software puede ser aplicada a proyectos en desarrollo o finalizados ya que por medio de los indicadores propuestos se puede realizar actividades de auditoria de cada etapa del proceso de la ingeniería de software.

5.2. Recomendaciones

- La aplicación de metodologías para gestionar calidad en un proyecto debe ser introducido como un paradigma, tal como se propone en la Administración de Calidad Total, es decir hacer de la gestión de la calidad de software una actividad de responsabilidad de todos los implicados en el proyecto
- A lo largo del desarrollo de la guía existen ciertos indicadores, los mismos que permiten establecer la eficiencia y la eficacia con la que se realizo el software, es primordial considerar los limites inferiores o superiores que se proponen en dichos indicadores al momento de interpretar los resultados de alguna métrica.
- El grupo de desarrollo debe encargarse de un correcto archivo de los documentos resultantes de la presente guía de manera que estos puedan ser utilizados como retroalimentación en proyectos futuros

BIBLIOGRAFIA

- Pressman, Roger S. (2002). Ingeniería del software. Un enfoque práctico. Quinta Edición. Madrid. Mc. Graw Hill.
- Booch, Grady. Jacobson, I. & Rumbaugh, J.(2002). Utilización del UML en la ingeniería del software con Objetos y componentes. Primera Edición. Madrid. Addison Wesley.
- Donaldson Scout. (2001) Successful Software development. Segunda Edición.
- Booch G. (1999). El Lenguaje Unificado de Modelado. Primera Edición

FUENTES

- http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/aucalid.htm
- http://www.iti.upv.es/actualidadtic/2003/07/2003-07-calidad.pdf
- http://www.issco.unige.ch/ewg95/node14.html
- www.itber.edu.mx/comunidad /material /ing-software / jesus masrtinez sangerman

Diana Elizabeth Arias López, nacida un 19 de Junio de 1980 en la ciudad de Quito curso sus estudios en la Escuela Nuestra Madre de la Merced, posteriormente realizo sus estudios secundarios en el Colegio Nuestra Madre de la Merced obteniendo el titulo de Bachiller en Físico – Matemático en el año de 1998, sus estudios universitarios lo realizo en la Escuela Politécnica del Ejercito en la carrera de Ingeniería de Sistemas e Informática. A Continuación haré mención a otros cursos realizados:

- Escuela Politécnica del Ejercito

  Seminario Taller "INFORMATICA EDUCATIVA"

  Junio de 2003

- Microsoft

  Curso de TechNet 2000

  Enero de 2001

- Escuela Politécnica del Ejercito

  Curso de Motivación Expositor Miguel Ángel Cornejo

  Mayo de 2001

- Cámara de Comercio de Quito

  Escuela de Informática de 7 niveles

  Enero de 1995

- Gío Modelos

  Curso de Modelaje y Protocolo

  Agosto de 1996

Montesdeoca Stacey Diego Esteban, nacido un 2 de noviembre de 1980 en la ciudad de Quito. Curo sus estudios en las escuela Unidad Educativa Particular Mixta Bilingüe “Ángel Polivio Chávez” y sus estudios secundarios los realizo en el Colegio San Gabriel obteniendo el titulo Bachiller de Físico Matemático (1998), posteriormente realizo sus estudios universitarios en la Escuela Politécnica del Ejercito en la carrera de Ingeniería de Sistemas e Informática. A continuación haré mención a otros cursos realizados:

- Administración de Linux y servicios de Internet

  Escuela Superior Politécnica del Ejército

  Fecha: 19 al 30 de agosto 2002
- Escuela Politécnica del Ejercito Facultad de Idiomas.

  Suficiencia en el idioma Inglés

  Fecha: 3 de marzo 2000
- Curso de Mantenimiento en computadores e Internet. S.E.C.A.P.

  Titulo: Auxiliar técnico en mantenimiento de computadores

  Fecha: 25 de Febrero 2000
- CENTRO DE ESTUDIOS DE LENGUAS Y ESPECIALIZACIONES ADMINISTRATIVAS. C.E.N.D.I.A.

  Suficiencia en el idioma Inglés

  Fecha: 19 de diciembre 1998
- UNIPLEX.

  Brio Query Design

  Fecha: diciembre 2002