



Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ingeniería en Ciencias y Sistemas

**ANÁLISIS DE LOS MÉTODOS DE CONFIABILIDAD QUE SON APLICABLES  
EN LA INGENIERÍA DEL SOFTWARE**

**Walter Estuardo Vásquez Sarazúa**

Asesorado por el Ingeniero Neftalí de Jesús Calderón Méndez

Guatemala, mayo de 2009

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

**ANÁLISIS DE LOS MÉTODOS DE CONFIABILIDAD QUE SON APLICABLES  
EN LA INGENIERÍA DEL SOFTWARE**

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA  
FACULTAD DE INGENIERÍA

POR:

**WALTER ESTUARDO VÁSQUEZ SARAZÚA**

ASESORADO POR EL ING. NEFTALÍ DE JESÚS CALDERÓN MÉNDEZ

AL CONFERÍRSELE EL TÍTULO DE  
**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, MAYO DE 2009

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA  
FACULTAD DE INGENIERÍA



**NÓMINA DE JUNTA DIRECTIVA**

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Inga. Glenda Patricia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Milton de León Bran
VOCAL V	Br. Isaac Sultan Mejía
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

**TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO**

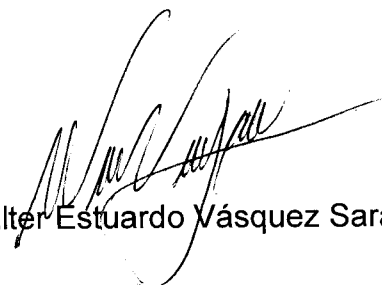
DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADORA	Inga. Virginia Victoria Tala Ayerdi de León
EXAMINADORA	Inga. Sonia Yolanda Castañeda de De Paz
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

## HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

### **ANÁLISIS DE LOS MÉTODOS DE CONFIABILIDAD QUE SON APLICABLES EN LA INGENIERÍA DEL SOFTWARE**

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha Febrero de 2008.



Walter Estuardo Vásquez Sarazúa

Guatemala, 23 de Marzo 2009



Ingeniero  
Carlos Alfredo Azurdia Morales  
Revisor de Trabajos de Graduación

Respetable Ing. Azurdia:

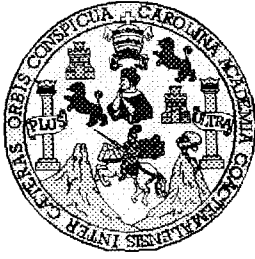
Por este medio hago de su conocimiento que he asesorado el trabajo de graduación del estudiante WALTER ESTUARDO VÁSQUEZ SARAZÚA, carné: 1992-13423, titulado: "ANALISIS DE LOS METODOS DE CONFIABILIDAD QUE SON APLICABLES EN LA INGENIERÍA DEL SOFTWARE", y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Sin otro particular, me suscribo de usted.

Atentamente,



Neptalí de Jesús Calderón Méndez  
Ingeniero en Ciencias y Sistemas  
Colegiado No. 8051  
Asesor de Trabajo de Graduación



Universidad San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 05 de Mayo de 2009

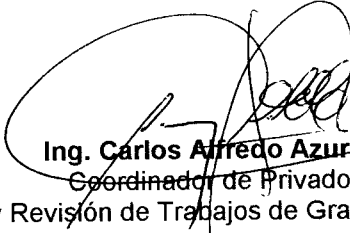
Ingeniero  
**Marlon Antonio Pérez Turk**  
Director de la Escuela de Ingeniería  
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **WALTER ESTUARDO VASQUEZ SARAZUA**, titulado: "**ANÁLISIS DE LOS MÉTODOS DE CONFIABILIDAD QUE SON APLICABLES EN LA INGENIERÍA DEL SOFTWARE**", y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

  
**Ing. Carlos Alfredo Azurdia**  
Coordinador de Privados  
y Revisión de Trabajos de Graduación



E  
S  
C  
U  
E  
L  
A  
  
D  
E  
  
C  
I  
E  
N  
C  
I  
A  
S  
  
Y  
  
S  
I  
S  
T  
E  
M  
A  
S

UNIVERSIDAD DE SAN CARLOS  
DE GUATEMALA



FACULTAD DE INGENIERÍA  
ESCUELA DE CIENCIAS Y SISTEMAS  
TEL: 24767644

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado "ANÁLISIS DE **LOS MÉTODOS DE CONFIABILIDAD QUE SON APLICABLES EN LA INGENIERÍA DEL SOFTWARE**", presentado por el estudiante WALTER ESTUARDO VÁSQUEZ SARAZÚA, aprueba el presente trabajo y solicita la autorización del mismo.*

**"ID Y ENSEÑAD A TODOS"**

Ing. Marlon Antonio Pérez Turk  
Director, Escuela de Ingeniería Ciencias y Sistemas

Guatemala, 20 de mayo 2009



El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **ANÁLISIS DE LOS MÉTODOS DE CONFIABILIDAD QUE SON APLICABLES EN LA INGENIERÍA DEL SOFTWARE**, presentado por el estudiante universitario **Walter Estuardo Vásquez Zarazúa**, autoriza la impresión del mismo.

IMPRÍMASE.

A large, handwritten signature in black ink, appearing to read 'Murphy Olympo Paiz Recinos', written over a large, empty oval shape.

Ing. Murphy Olympo Paiz Recinos  
DECANO



Guatemala, mayo de 2009

/cc  
c.c. archivo.



## **ACTO QUE DEDICO A:**

Este trabajo de graduación es dedicado a cada una de las personas que pusieron su grano de arena en el transcurso de mi carrera, principalmente a :

Ser Supremo	Por darme la oportunidad de estar presente hoy aquí, darme la sabiduría y la fuerza necesaria para obtenerlo.
Mis padres	Jesús Sarazúa y Arturo Vásquez, por esa fortaleza, sabiduría y persistencia en la vida, que uno a uno fueron forjando con su ejemplo el diario existir.
Mi abuela	Dolores Flores, por ser como una madre y siempre apoyarme con sus sagrados alimentos.
Mi hermano	El Dr. Rony Vásquez, por ser un ejemplo, un padre y un amigo de toda la vida.
Mi novia	Mónica Estrada, por darme ánimos cuando hubo que caminar cuesta arriba.

## **AGRADECIMIENTOS A :**

- Principalmente a Dios y a los que espiritualmente me apoyaron y me guiaron en el transcurso de mi carrera.
- La universidad de San Carlos de Guatemala.
- La Biblioteca Central, por brindarme la oportunidad de poner en práctica mis conocimientos y darme el tiempo necesario para cumplir con aprendizaje.
- Mis compañeros de trabajo que me brindaron el material necesario en los momentos críticos y muy especialmente a Angélica de Lara, Nicolas Betancourt, Zulma Calderón, Lic. Mercedes de Beeck, Angelica Menchu, a todos ellos gracias por brindarme su experiencia ayuda y amistad.
- Mi amigo Salomón Herrera por brindarnos un lugar en donde llevar a cabo proyectos, y amistad sincera.
- Mi asesor Neftali de Jesus Calderon Mendez, por tomar un poco de su tiempo en la revisión de este trabajo de graduación.
- Mis compañeros de estudio, por compartir decepciones, triunfos y especialmente la amistad.
- Mi amigo Héctor Cardona López, compañero, amigo y casi hermano.

## ÍNDICE GENERAL

<b>ÍNDICE DE ILUSTRACIONES</b>	VII
<b>GLOSARIO</b>	IX
<b>RESUMEN</b>	XIII
<b>OBJETIVOS</b>	XV
<b>INTRODUCCIÓN</b>	XVII
<b>1 LA CALIDAD</b>	<b>1</b>
1.1 Etapas del control de calidad	2
1.2 Calidad total	5
1.3 La gerencia de un proyecto en el control de calidad	7
1.3.1 La acción gerencial	7
1.3.2 Estrategia para agregar calidad a la gestión	8
1.4 Control de Calidad Total	9
1.5 La garantía de la calidad	10
1.5.1 El círculo de calidad	11
1.6 Catorce principios expuestos por Dr. Deming	12
1.6.1 Las siete enfermedades mortales del control de calidad	15
1.7 Obstáculos del control de calidad	17
1.8 EL control de la calidad a través de herramientas estadísticas	17
1.8.1 Hoja de recolección de datos	19
1.8.2 Histograma	20
1.8.3 Diagrama de Pareto	21
1.8.4 Diagrama de Causa-Efecto	22
1.8.5 Diagrama de dispersión	23
1.8.6 Estratificación	24
1.8.7 Corridas y gráficas de control	25

<b>2</b>	<b>INGENIERÍA DEL SOFTWARE</b>	<b>29</b>
2.1	Antecedentes	29
2.1.1	La evolución del software	30
2.1.2	Características del software	33
2.1.3	La ingeniería del software	34
2.2	Metodologías, técnicas y herramientas de la ingeniería del software	37
2.3	Metodologías	39
2.3.1	Modelo en cascada o ciclo de vida clásico	39
2.3.2	El modelo en espiral	40
2.3.3	El modelo desarrollo rápido de aplicaciones	42
2.3.4	El modelo de ensamblaje de componentes	42
2.4	Técnicas	43
2.4.1	Técnicas de construcción de prototipos	43
2.4.2	Técnicas de cuarta generación	44
2.5	Herramientas	45
2.5.1	Herramientas CASE	45
2.6	Gestión de ingeniería del software	47
2.6.1	Personal	48
2.6.1.1	Los participantes	48
2.6.1.2	Los jefes de equipo o gestor de proyecto	49
2.6.1.3	El equipo de trabajo	50
2.6.2	El problema	51
2.6.2.1	Ámbito del software	51
2.6.2.2	Descomposición del problema	52
2.6.2.3	Problemas por los mitos del software	53
2.7	El proceso	54

<b>3</b>	<b>ISO PARA DESARROLLADORES DE SOFTWARE</b>	<b>55</b>
3.1	Qué es ISO?	55
3.2	Qué es una norma	56
3.3	Cómo trabajan las normas ISO 9000	56
3.4	La certificación ISO	57
3.5	ISO para desarrolladores de software	59
3.6	Norma ISO-9000-3: Guías para la aplicación de ISO 9001 para el desarrollo, entrega y mantenimiento de software	60
3.7	Secciones de la norma ISO 9000-3	60
	<b>Sección 3</b> Definiciones	61
	<b>Sección 4</b> Sistemas de calidad marco de trabajo	62
	<b>Sección 4.1</b> Responsabilidad de la gestión	62
	<b>Sección 4.1.1</b> Responsabilidad administrativa del proveedor	62
	<b>Sección 4.1.2</b> Responsabilidad administrativa de los clientes	63
	<b>Sección 4.1.3</b> Reuniones de revisión	64
	<b>Sección 4.2</b> Sistemas de calidad	64
	<b>Sección 4.2.1</b> Generales	64
	<b>Sección 4.2.2</b> Documentación de los sistemas de calidad	65
	<b>Sección 4.2.3</b> Plan de calidad	65
	<b>Sección 4.3</b> Auditorías internas del sistema de calidad	65
	<b>Sección 4.4</b> Acción correctiva	66
	<b>Sección 5</b> Sistemas de calidad -ciclo de vida de las actividades	67
	<b>Sección 5.1</b> Generalidades	67
	<b>Sección 5.2</b> Revisiones del contrato	67
	<b>Sección 5.3</b> Especificaciones de los requisitos del comprador	68
	<b>Sección 5.4</b> Planificación del desarrollo	69
	<b>Sección 5.5</b> Planificación de la calidad	69

<b>Sección 5.6</b>	Diseño e implementación	70
<b>Sección 5.6.1</b>	Diseño	71
<b>Sección 5.6.2</b>	Implementación	72
<b>Sección 5.7</b>	Ensayos y validación	72
<b>Sección 5.7.1</b>	Planificación de las pruebas	73
<b>Sección 5.7.2</b>	Validación	74
<b>Sección 5.8</b>	Aceptación	74
<b>Sección 5.9</b>	Reproducción, entrega e instalación	75
<b>Sección 5.9.1</b>	Entrega	75
<b>Sección 5.9.2</b>	Instalación	76
<b>Sección 5.10</b>	Mantenimiento	76
<b>Sección 5.10.1</b>	Plan de mantenimiento	77
<b>Sección 5.10.2</b>	Tipos de actividades del mantenimiento	77
<b>Sección 5.10.3</b>	Registro y reportes de mantenimiento	78
<b>Sección 6</b>	Sistemas de Calidad- Actividades y Soporte	79
<b>Sección 6.1</b>	Planificación de la configuración administrativa	80
<b>Sección 6.1.1</b>	Actividades de la gestión de la configuración administrativa	80
<b>Sección 6.1.1.1</b>	Identificación de la configuración y su seguimiento	80
<b>Sección 6.1.1.2</b>	Control de cambios	81
<b>Sección 6.1.1.3</b>	Reporte de los estados de las configuraciones	82
<b>Sección 6.2</b>	Control de documentos	82
<b>Sección 6.2.1</b>	Tipos de documentos	82
<b>Sección 6.2.2</b>	Cambio de documentos	84
<b>Sección 6.3</b>	Registros de calidad	84
<b>Sección 6.4</b>	Medición	85
<b>Sección 6.4.1</b>	Medición del producto	85
<b>Sección 6.4.2</b>	Proceso de medición	86

<b>Sección 6.5</b>	Reglas, prácticas y convenciones	87
<b>Sección 6.6</b>	Herramientas y técnicas	87
<b>Sección 6.7</b>	Entrega	87
<b>Sección 6.7.1</b>	Evaluación de subcontratos	87
<b>Sección 6.7.2</b>	Validación del producto entregado	88
<b>Sección 6.8</b>	Producto de software incluido	89
<b>Sección 6.9</b>	Capacitación	89
<b>4</b>	<b>MÉTRICAS DEL SOFTWARE</b>	<b>91</b>
4.1	Actividades del proceso de medición	92
4.2	¿Qué es un métrico?	93
4.3	Clasificación de las métricas	94
4.4	Diferentes enfoques de las métricas	95
4.5	Diseño de la métrica	96
4.6	Panorama de las métricas del producto	101
4.6.1	Métricas para el modelo de análisis	103
4.6.1.1	Métrica basadas en la función	103
4.6.1.1.1	Puntos de función (PF)	103
4.6.1.1.2	Métricas del tamaño del sistema (La métrica Bang)	108
4.6.1.1.3	Métrica para la calidad de la especificación	111
4.6.2	Métricas para el modelo de diseño	113
4.6.2.1	Métricas del diseño arquitectónico	113
4.6.2.2	Métricas al nivel de componente	115
4.6.2.2.1	Métricas de cohesión	115
4.6.2.2.2	Métricas de acoplamiento	117
4.6.2.2.3	Métricas de complejidad	119
4.6.2.3	Métricas de la interfaz	120
4.6.2.4	Métricas de diseño orientado a objetos	122
4.6.2.4.1	Métricas orientadas a clases	124

4.6.2.4.2	Métricas para el diseño orientado a objetos	128
4.6.3	Métricas para el código fuente	129
4.6.4	Métricas para las pruebas	131
4.6.5	Métricas para el mantenimiento	134
<b>5</b>	<b>TÉCNICAS DE ESTIMACIÓN DE COSTOS DE SOFTWARE</b>	<b>135</b>
5.1	Juicio de un experto	135
5.2	La estructura de los modelos de estimación	136
5.3	El modelo COCOMO	137
5.3.1	Modelo básico	141
5.3.1.1	Modelo orgánico	141
5.3.1.2	Modelo empotrado	142
5.3.2	Modelo intermedio	143
5.3.2.1	Ecuaciones nominales de coste	143
5.3.2.2	Atributos de coste	143
5.3.3	Modelo avanzado	144
5.4	La ecuación del software	144
5.5	El modelo CMM	146
5.6	Modelo de productividad	149
5.7	Estimación para proyectos orientados a objetos	149
<b>6</b>	<b>RESULTADOS DEL TRABAJO DE CAMPO</b>	<b>153</b>
	<b>CONCLUSIONES</b>	<b>161</b>
	<b>RECOMENDACIONES</b>	<b>163</b>
	<b>BIBLIOGRAFÍA</b>	<b>165</b>



## ÍNDICE DE ILUSTRACIONES

### FIGURAS

1	El círculo de calidad	11
2	Hoja de recolección de datos	19
3	Histograma de frecuencias	20
4	Diagrama de Pareto	24
5	Diagrama de Causa-Efecto	23
6	Diagrama de dispersión	24
7	Estratificación	25
8	Corridas y gráfica de control	26
9	Evolución del software	31
10	Ciclo de vida del software con comunicación entre etapas	40
11	Modelo en espiral	41
12	Construcción de prototipos	44
13	Técnicas de cuarta generación	45
14	Elementos de una herramienta CASE	47
15	Tamaño de diseño y desarrollo de un sistema de cómputo	104
16	Jerarquía de clases	126
17	Modelo de productividad	149
18	Gráfico de conocimiento sobre las normas ISO	153
19	Gráfico de control de calidad en el desarrollo de software	154
20	Gráfico para medir la importancia de la métrica	155
21	Gráfico del uso de métricas conocidas	156
22	Gráfico de métricas aplicables al modelo de análisis	157

23	Gráfico de conocimiento y uso de métricas en los modelos orientados a objetos	158
24	Gráfico de uso de métricas para diseño orientados a objetos	159

## **TABLAS**

I	Cálculo de puntos de función	106
II	Factores de ajuste para puntos de función	107
III	Preguntas para el cálculo del factor de ajuste	107
IV	Modelos de estimación orientados a líneas de código	137
V	Modelos de estimación orientados a líneas de código	137
VI	Factores multiplicadores de esfuerzo en COCOMO	140
VII	Constantes para calcular el esfuerzo del modelo COCOMO	141
VIII	Tipo de interfaz y multiplicador	150

## GLOSARIO

<b>Administración</b>	Es el proceso de planificar, ejecutar y controlar un conjunto de actividades de manera eficiente con el objeto de coordinar un grupo de personas y recursos tecnológicos esenciales para alcanzar un objetivo.
<b>Actividad</b>	Cualquier paso o función que se realiza de forma mental o física para alcanzar algún objetivo, incluyendo todo trabajo realizado para realizar las tareas del proyecto o la organización.
<b>Calidad</b>	Palabra que designa el conjunto de atributos o propiedades de un objeto que nos permite emitir un juicio acerca de él.
<b>Calidad total</b>	Es el sistema administrativo en el que quedan coordinados los esfuerzos de todos, administradores y trabajadores, a favor de la calidad de los productos o servicios que presta la empresa, en el que se definen procedimientos con el fin de producir en forma económica los bienes a entera satisfacción del cliente o usuario.
<b>Circulo de calidad</b>	Es un grupo pequeño de personas, que desarrolla actividades que buscan mejorar los procesos, el ambiente y la utilización de los recursos.
<b>Cultura organizacional</b>	Es el conjunto de valores y principios que

determinan la forma de ser de una organización y los factores que constituyen su fortaleza y los elementos decisivos de su productividad.

**Control**

Es un proceso mediante el cual se verifica si los resultados concuerdan con los valores aceptables.

**COCOMO**

Constructive Cost Model o Modelo Constructivo de costos, es un modelo de estimación de costes de software que incluye tres submodelos, donde cada uno ofrece un nivel de detalle y aproximación cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado.

**CMM**

Capability Maturity Model o Modelo de Capacidad y Madurez, modelo de evaluación de procesos para una organización.

**Evolución**

Es simplemente cambio, desarrollo gradual, crecimiento o avance de las cosas o de los organismos.

**Equipo de trabajo**

Un equipo de trabajo es un grupo pequeño de personas cuyas capacidades individuales se complementan y que se comprometen conjuntamente para una causa común, logran metas, operan con una metodología común, comparten responsabilidades.

<b>Gerencia<sup>i</sup></b>	conjunto de empleados de alta calificación que se encarga de dirigir y gestionar los asuntos de una <b>empresa</b> . El término también permite referirse al cargo que ocupa el director general (o gerente) de la empresa, quien cumple con distintas funciones: coordinar y representar a la compañía frente a terceros y controlar las metas y objetivos.
<b>Hardware</b>	El hardware abarca todas las piezas físicas de un ordenador.
<b>ISO<sup>ii</sup></b>	International Organization for Standardization - Organización Internacional para la Estandarización. Se encarga de crear estándares o normas internacionales.
<b>Ingeniería del software</b>	El enfoque sistemático para el desarrollo, operación modificación y eliminación de software.
<b>Inspección</b>	Acción o efecto de examinar una cosa en comparación de otra, de un estándar o una referencia establecida.
<b>PF</b>	Siglas utilizadas para nombrar Puntos de Función, técnica para estimar tiempo de duración de un proyecto de software.
<b>Sistema<sup>iii</sup></b>	Conjunto de elementos interrelacionados que persiguen un objetivo.

---

<sup>i</sup> <http://definicion.de/gerencia/>

<sup>ii</sup> <http://www.iso.org>, <http://www.alegsa.com.ar/Dic/iso.php>

<sup>iii</sup> [http://html.rincondelvago.com/empresas\\_1.html](http://html.rincondelvago.com/empresas_1.html)

**Software**

Programas, procedimientos, reglas y documentación posible asociada con la computación, así como los datos pertenecientes a la operación de un sistema de cómputo, relacionado con el trabajo intelectual que se realiza en una empresa.

## RESUMEN

El uso de las Tecnologías de Información y Comunicación (TIC) por las empresas, ha sido un factor importante para aumentar notablemente la productividad, permitiéndoles el acceso y manejo de la misma en tiempo real, y obtener de esa manera una ventaja competitiva.

Por el uso de la información que actualmente se hace a través de sistemas de computación, es indispensable que el software, como cualquier otro producto, cumpla con ciertas normas y estándares en su desarrollo para que sea confiable, rentable, productivo y que sea considerado de alta calidad.

En este trabajo se habla de calidad y cómo lograr la calidad total, que no involucra solamente al producto final, sino a todos los involucrados en el proceso, y cómo la acción gerencial debe hacer un cambio de pensamiento y hacer estrategias orientadas hacia el control de calidad, y entender, qué es un proceso continuo y que en cada etapa hay que identificar los obstáculos para lograr el objetivo, valiéndose de las herramientas estadísticas que orienten al gerente en que rumbo tomar.

Además, se describen las metodologías, técnicas y herramientas que se utilizan en la Ingeniería del Software, cuyas características se centran en garantizar la calidad de los procesos de desarrollo de software, enfocándose, en cómo pueden ser utilizadas en cada una de las fases del desarrollo y su monitoreo, para, así garantizar un producto que satisfaga las expectativas del cliente, sin dejar por un lado la importancia del factor humano, donde la comunicación y líneas de mando son vitales para la motivación y un equipo de trabajo en armonía del lado del cliente y del desarrollador.

Adicionalmente, se explica lo que es una métrica y sus aplicaciones en la Ingeniería del Software, y la importancia de su uso para obtener un

parámetro de medida que ayude a indicar que tanto se aproxima a los valores aceptables de calidad. Además, se hace énfasis en las secciones de la aplicación de la norma ISO 9000-3, que es aplicable al desarrollo de productos de software, y la importancia de su certificación como valor agregado en un mercado global competitivo, y al final, se muestran los resultados de la toma de una encuesta desarrolladoras de software referente a la aplicación de algún método que les ayude a garantizar la calidad en el producto que fabrican.

En adición, se explica las técnicas utilizadas para la estimación de costos en un producto de software, basadas algunas en el juicio de un experto y otras orientadas específicamente al modelo de desarrollo elegido, sin dejar por un lado las enfocadas directamente a la programación orientada a objetos.



## **OBJETIVOS**

### **GENERAL :**

Dar a conocer el concepto de calidad y las diferentes metodologías de desarrollo de productos de software, y como una norma ISO puede ser aplicada para garantizar que el producto final sea de calidad.

### **ESPCÍFICOS :**

1. Conocer estándares y modelos de software que al seguir su aplicación en las diferentes fases del desarrollo de software ayudan a garantizar la calidad de los procesos.
2. Hacer una descripción de la norma ISO 9000-3, que a nivel internacional es aplicable y garantizar la calidad del producto de software a entregar.
3. Describir las métricas que pueden aplicarse a cada una de las fases de desarrollo de software.
4. Dar resultados de las encuestas realizadas para verificar el conocimiento de las diferentes técnicas que son aplicables en la ingeniería del software y que ayudan a garantizar la calidad del producto terminado.



## INTRODUCCIÓN

La calidad es un concepto que existe desde hace muchos años, a lo largo de la historia el concepto calidad ha sufrido un sin número de cambios, desde la etapa artesanal, pasado por la revolución industrial, períodos de guerra y postguerra y a lo largo de estos períodos el fin fue siempre garantizar la satisfacción del cliente, y la computación de igual manera ha evolucionado en generaciones desde la década de los 50, hasta nuestros días, dando saltos altos a nivel de tecnología no solo en hardware sino en software, y de ahí la importancia que sea posible el garantizar que el software fabricado sea de calidad.

El tema del trabajo de graduación se desarrolla en cinco capítulos; el primer capítulo habla específicamente sobre la calidad, su control y los obstáculos para llegar a la calidad total, seguido de las herramientas que se utilizan para poder medirla.

En el segundo capítulo encontrará, sobre la ingeniería del software, algunos antecedentes de la misma, características y algunas metodologías y técnicas utilizadas en el desarrollo de software, de igual manera se habla sobre su gestión y trata temas como el personal, el equipo de trabajo y el problema del desarrollo de software.

El capítulo tres aborda específicamente sobre lo que es una norma, el cual trata sobre las normas ISO y específicamente sobre la ISO-9000-3 para desarrolladores de software, hay una descripción de las secciones de dicha norma y cómo es aplicable a cada una de las fases del desarrollo.

El cuarto capítulo involucra las métricas del software, las cuales están orientadas al modelo de análisis, diseño, pruebas, mantenimiento y orientadas a objetos, con el fin de garantizar que para cada una de las fases se conozca las técnicas que se utilizan en el desarrollo, todo con el objetivo de que se entregue un producto con una calidad aceptable y competitiva.

El capítulo cinco habla sobre las técnicas de estimación de los costos al momento de desarrollar un producto de software, tocando modelos como el Juicio de un Experto, el Modelo COCOMO, el modelo CMM y trata brevemente como estimar costos en modelos orientados a objetos.

Y por último, el capítulo seis es un trabajo de campo, el cual consistió en el llenado de una encuesta por un número de 30 empresas evaluando las técnicas que ellos utilizaban y así verificar cuáles utilizaban para garantizar que su producto será entregado con un nivel aceptable de calidad.

## 1. LA CALIDAD

Es un concepto que existe desde hace muchos años, cuando el hombre fabricó su primera herramienta y se dio cuenta que al modificarla y perfeccionarla para su uso, obtenía mejores resultados, buscó desde entonces la perfección en cada una de las actividades en las que se encuentra inmerso.

A lo largo de la historia el concepto calidad ha sufrido un sin número de cambios, en la etapa artesanal, la calidad se entendía en hacer las cosas bien, independientemente del coste o esfuerzo, cuyo fin era únicamente satisfacer al consumidor. En la revolución industrial, la calidad consistía en hacer muchas cosas, no importando que no fueran de calidad, en la Segunda Guerra Mundial fue asegurar la eficiencia del armamento sin importar el costo, con la mayor y más rápida producción, el fin fue garantizar la disponibilidad de un armamento en la cantidad y el momento preciso, durante el periodo de posguerra fue crear las cosas bien y la finalidad fue minimizar los costos y satisfacer al consumidor surgiendo así el concepto de competitividad, al finalizar la Segunda Guerra Mundial la finalidad fue, cuanto mayor producción mejor, y el fin fue satisfacer la gran demanda de bienes causada por la guerra.

De esto deducimos que ***“calidad es un conjunto de propiedades o atributos de un objeto que permite que se emita un juicio a cerca de él”<sup>1</sup>***. Según esta definición, podemos pensar que la calidad puede ser mala, buena o excelente.

Cuando se dice que algo tiene calidad, esta expresión provee un juicio positivo a las características del objeto. Con lo anterior hace un equivalente al significado de excelencia y perfección.

---

<sup>1</sup> <http://www.monografias.com/trabajos7/catol/catol.shtml?relacionados>

Si se habla de calidad se puede pensar en perfeccionar una función o proceso y salta a la mente el concepto de sistema para la calidad. Desde que inicia la educación formal se comprende la definición de sistema como “**Un conjunto de elementos interrelacionados con un objetivo común**”<sup>2</sup>.

De lo anterior se observa que el concepto de calidad se puede aplicar a cualquier tipo de producto que se encuentre en proceso de fabricación, de igual forma al momento de desarrollar software se está fabricando un producto, el cual tendrá un grado de calidad, el que se basará en su facilidad de uso, disponibilidad y eficiencia. Cabe mencionar que en dicho proceso se puede utilizar el seguimiento de estándares para buscar la calidad, y así garantizar que el producto al finalizar posea un nivel de aceptación óptimo, con un parámetro para indicar el grado de calidad con el cual cuenta.

### **1.1 Etapas del control de la calidad:**

- **Control de calidad mediante la Inspección:** Cuando se inicia con la producción de productos en línea es importante su inspección y verificar si cumple con la finalidad para lo cual ha sido creado, todo esto debido a la falta de uniformidad del producto. Las empresas se vieron obligadas a montar departamentos de control de calidad para dicho trabajo, y hubo que realizar la inspección de cerca y de una manera crítica y de esa forma medir su calidad y detectar los errores. Si se observaba un desperfecto, personas especializadas en la materia debían ponerle remedio.
  
- **El control estadístico de la calidad:** Esto fue posible por los estudios que realizaron Bell Telephone Laboratories, W.A. Shewhart, Harold Dodge, Harry Roming y más tarde, G: D: Edwards y Joseph Juran.

---

<sup>2</sup> <http://paginespersonals.upcnet.es/~jmg2/libro/ds0k9.htm>

W.A. Shewhart, fue el primero en reconocer que una producción posee una variación en dicho proceso, esto según su libro publicado en el año 1931 titulado "Economic Control of Quality of Manufactured Product", de aquí que la variación debe de ser estudiada con los principios de la probabilidad y de la estadística.

Según el concepto de administración no interesa la eliminación de dicha variación lo cual sería hasta cierto punto imposible, sino ver en que rango es aceptable sin originar problemas.

El muestreo parte que en una producción masiva es imposible llevar la inspección de todos los productos y diferenciar los buenos de los malos. De esto existe la necesidad de tomar una muestra de un lote, analizarlo y decidir si es aceptable o no.

- **Aseguramiento de la calidad:** esta etapa se caracteriza por la toma de conciencia por parte de los administradores y del papel que cada uno de ellos tiene en aseguramiento de la calidad, el establecimiento de planes de acción preventivo en lo que a calidad se refiere. Es necesario que el mejoramiento logrado a través de el control estadístico, y esto lleva a desarrollar profesionales dedicados para el aseguramiento de la calidad e involucrar a todos para ese logro, esto implica un mayor compromiso por parte de la administración.

Dentro del control de calidad existen cuatro exponentes de renombre, quines son Edward Deming, Joseph Juran, Armand Feigenbaum y Philip B.Crosby. El Dr. Deming, pone a relieve la responsabilidad que la alta gerencia tiene en la producción de artículos defectuosos. Juran, investiga los costos de calidad. Feigenbaum, por su parte concibe el sistema administrativo como coordinador, en la

compañía, del compromiso de todos en orden a lo largo de calidad. Crosby es el promotor del movimiento denominado cero defectos

- **La calidad como estrategia competitiva:** tiene como fin el mercado y las necesidades del consumidor, reconociendo el efecto que puede tener la calidad como estrategia fundamental para alcanzar competitividad.

El hecho de aplicar métodos estadísticos para el control de los procesos no indica que la calidad pase a ser una estrategia competitiva solo por sí sola, como de igual manera que todos se comprometan a la creación de productos sin defectos, pues de nada sirve si no hay mercado.

La calidad pasa ser una estrategia competitiva en el momento en que la alta gerencia toma como punto de partida para su planeación estrategia los requerimientos del consumidor y la calidad de los productos de los competidores. Se trata de planear cada una de las actividades de la empresa y en tal forma que se ofrezcan productos que llenen las expectativas de los requerimientos del cliente y que tengan una calidad mayor a la que ofrecen los competidores.

- **La reingeniería de procesos:** Con los cambios tecnológicos y con el mejoramiento de la comunicación, se llegó a la globalización de los mercados, y el término de Reingeniería de Procesos se hizo popular, las empresas lo han utilizado para mejorar de una manera rápida, los procesos administrativos, producción y comercialización, pues al no hacerlo les resta competitividad.



Existen muchas definiciones acerca de Reingeniería<sup>3</sup> como revisión fundamental y el rediseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y competentes, tales como calidad, costos, servicio y rapidez de entrega, sin embargo en lenguaje común se puede definir como "empezar de nuevo".

## 1.2 Calidad Total

Para alcanzar el éxito en el proceso de mejoramiento continuo es importante establecer una buena política de calidad, en la que se defina de forma precisa lo que se espera de empleados y productos o servicios que se brindan al consumidor.

Una política de calidad en su redacción debe indicar de forma precisa que se espera de un empleado y poderse aplicar, de igual forma, la calidad de los productos o servicios que ofrece la compañía. Así también, establecer claramente los estándares de calidad y lograr cubrir todos los aspectos relacionados con el sistema de calidad.

Para dar efecto a la implantación y puesta en marcha de dicha política de calidad, es necesario que los empleados tengan los conocimientos para las exigencias de los clientes y así ofrecer los productos y servicios que puedan satisfacer o exceder las expectativas de los consumidores.

La calidad total es un concepto, una estrategia, un modelo de hacer negocios, la cual está enfocada directamente hacia el cliente, y el consumidor final quien utilizara nuestro producto o servicio.

Según el maestro de Calidad Total el Dr. Edward Deming define la Calidad Total como "**satisfacción de los requerimientos y las expectativas**

---

<sup>3</sup> <http://www.tuobra.unam.mx/publicadas/041229173633-Reingeni.html>

***de nuestros clientes, tanto internos como externos, en lo que se refiere a productos y servicios, la primera vez y a tiempo todas las veces" <sup>4</sup>***

La calidad total no se refiere al producto o servicio como tal, sino al mejoramiento continuo enfocado al aspecto organizacional, tomando un empresa como una maquina gigantesca en la que cada trabajador desde el gerente hasta el funcionario de más bajo nivel jerárquico este comprometido con los objetivos de la empresa.

Existen tres factores importantes para entender en su plenitud el concepto de calidad total:

- 1. Un cambio de actitud:** si en el proceso de producción se realiza el muestreo al fin de la línea para determinar, si un bien o servicio es bueno o malo, un cambio de actitud **es atender en su totalidad el proceso de producción para lograr la mejora.**
- 2. Un nuevo punto de referencia:** para establecer un nuevo punto de referencia hay que pensar que un producto puede ser bueno, pero si no satisface las expectativas de los consumidores, no sirve de nada. De esto, el punto de referencia para definir la calidad; es el hecho de que dichos productos satisfagan las expectativas de los consumidores.
- 3. Una nueva filosofía:** La clave es el proceso de mejora constante para que una empresa se consolide en el mercado, y se de mucha atención al ser humano y a su exigencias como cliente interno de la empresa y como el principal colaborador que al final es quien le agrega valor a los bienes y servicios que la empresa produce o genera.

---

<sup>4</sup> <http://www.aulafacil.com/administracionempresas/Lecc-9.htm>

### **1.3 La gerencia de un proyecto en el control de calidad**

Las características de los tiempos actuales inciertos, turbulentos, de cambios imprevistos imprimen particular relevancia a los modos de gestión y a las formas de pensar, decidir y actuar de los responsables de la conducción de las empresas. El funcionamiento efectivo y eficiente de las organizaciones y el logro de la misión para la cual fueron creadas, depende, en gran parte, de la habilidad que tenga el gerente para alcanzar los objetivos, mediante la cooperación voluntaria y el esfuerzo conjunto de todos. Cuando se habla del gerente, se refiere particularmente a su capacidad para orientar, dirigir, tomar decisiones y lograr resultados; de él depende su éxito personal, la empresa y del grupo que está dirigiendo. Obviamente que para pensar, tomar decisiones y emprender acciones de calidad se requiere, además de una formación gerencial, un patrón de criterios y una filosofía clara de la administración, de la concepción del hombre y una ideología del trabajo, que le permita ganar apoyo efectivo y partidarios comprometidos con una misión cuyo significado y trascendencia merece entrega.

#### **1.3.1 La acción gerencial**

La evolución en el campo de la gestión muestra un cambio hacia una fragmentación del trabajo gerencial, en el cual se entremezclan una serie de papeles interpersonales, informativos y decisivos con las clásicas funciones de la administración. A pesar de la importancia que hasta ayer se les ha asignado a los procesos administrativos de planificar, organizar, dirigir, coordinar y controlar. La esencia de la gerencia no está en ninguna de estas funciones, ni siquiera en la suma de todas ellas. La esencia de la acción gerencial hoy es imaginar, visionar, crear, innovar, integrar, hacer seguimiento, saber ser para integrar al hacer.

Lo que constituye al rasgo fundamental de la gerencia es la acción. Una acción gerencial de calidad la determina la alta gerencia, la calidad está en la mente, está en el corazón del gerente, es él quien tiene la responsabilidad de incidir para que se produzcan los cambios en los sistemas. El modo de pensar del gerente sobre las cosas, las personas y las organizaciones, es un factor crítico para el mejoramiento de la calidad, la productividad y la integración del personal.

El centro de la gerencia es el ser humano, a quien se sirve y el que sirve. Quien se sirve, paga un servicio o acude al servicio y espera calidad, y pagará o acudirá con mayor devoción en la medida en que este satisfecho. El que sirve, le agrega valor a su trabajo de manera que su esfuerzo produzca un bien que satisfaga. ¿De qué depende la dinámica entre "al que se sirve y el que sirve"? Simplemente, de cómo nos vemos nosotros mismos: el gerente no da lo que no tiene, ni expresa lo que no es. La visión que tiene de sí mismo afecta no sólo sus actitudes y comportamientos, sino también, la visión que tiene de otras personas. Por ello debe valorarse a sí mismo, valorar a los demás y tener claridad de los fines y principios constituye la esencia y el fundamento de una acción eficaz.

### **1.3.2 Estrategia para agregar calidad a la gestión**

Los planteamientos antes esbozados pueden ser logrados si se cultivan nuevos patrones de pensamiento. A tal efecto, se sugiere una estrategia dirigida a ampliar la capacidad para crear los resultados que se aspiran obtener. De lo anterior, las empresas exitosas serán aquellas que puedan sistematizar maneras de reunir a la gente para desarrollar estrategias creativas y así poder enfrentar las situaciones venideras.

La forma de agregar calidad a la gestión es una visión compartida, la visión compartida ha sido el centro del éxito y el elemento alrededor del cual

giran todos los demás componentes organizacionales, ello significa, que no es posible concebir una organización que haya alcanzado cierta grandeza, éxitos, egresados de calidad, sin una misión, visión, metas, valores y políticas que sean profundamente compartidas dentro de la organización. Una visión compartida es vital para las organizaciones, porque brinda concentración y energía para el aprendizaje. Y al mismo tiempo despierta el compromiso de mucha gente, porque ésta refleja la visión personal de esa gente. Las visiones compartidas crecen como subproductos de integración de visiones individuales.

Ello significa que la tarea fundamental de un gerente es redefinir su filosofía. Abandonar el viejo dogma de que la función de un gerente consiste en planificar, organizar, dirigir, coordinar y controlar. Trátese de investigación, docencia, extensión u otra actividad, en todas ellas la fuerza activa es la gente. Esta gente tiene su propia voluntad, su propio parecer y su modo de pensar. Esta gente tiene motivaciones e intereses que los gerentes gradualmente van destruyendo cuando desatienden sus necesidades. Si éstos no están motivados para alcanzar metas de crecimiento y desarrollo tecnológico, no habrá crecimiento, no habrá productividad, no habrá competitividad.

#### **1.4 Control total de calidad**

Si se toma en cuenta que la alta gerencia es la responsable que el sistema funcione adecuadamente, mientras que los demás trabajan dentro del mismo, aportando ideas para resolver los problemas y administrar la empresa de tal forma que la institución, como sistema, queda orientada a la calidad.

De lo anterior, se designa que control total de calidad es un sistema administrativo en que quedan coordinados los esfuerzos de los administradores y trabajadores, a favor de la calidad del producto o servicio que presta la empresa.

La introducción y puesta en marcha del control total de calidad implica efectuar un cambio de mentalidad en el personal y asimilar los nuevos procedimientos implicados en este cambio; esto requiere un período largo de tiempo. Adicionalmente, implica que la alta gerencia elija el tipo de organización que más convenga a la compañía, articulando metas, estrategias y asegure su fiel cumplimiento, de tal forma que los trabajadores, proveedores y clientes sepan que esperar de la empresa.

### **1.5 La garantía de la calidad**

Toda empresa debe garantizar que el cliente pueda comprar un bien o servicio y hacer uso de él con toda la confianza que se dan las características ofrecidas y al mismo tiempo se pueda utilizar un tiempo razonable en forma satisfactoria. A esto es lo que se conoce con el nombre de garantía de la calidad.

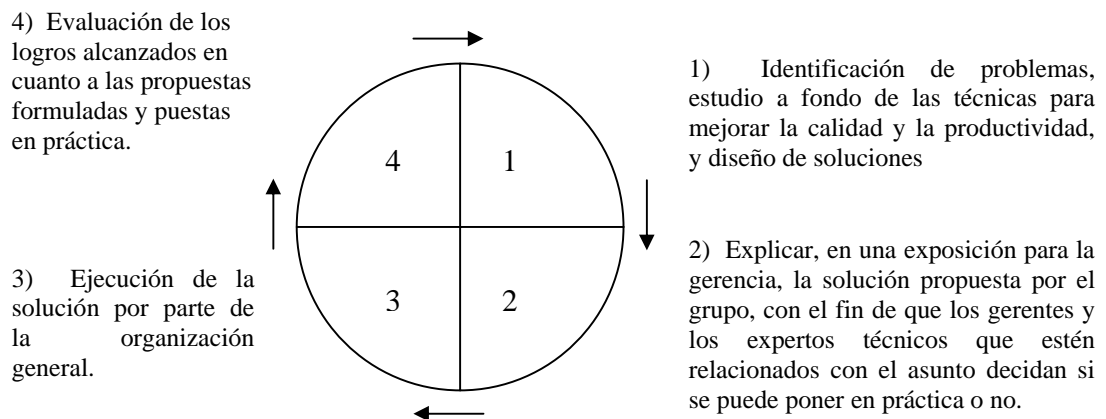
Según el Dr. Deming, para obtener la calidad que satisfaga al cliente se debe hacer lo siguiente:

- Dar una interacción de las actividades de investigación de mercado, de diseño del producto, de fabricación y de ventas con el propósito de mejorar los niveles de calidad.
- Repetir esta inspección en forma cíclica. Esta forma cíclica suele expresarse mediante el denominado círculo de Deming o ciclo de calidad.

### 1.5.1 El círculo de calidad<sup>5</sup>

Es un grupo pequeño de empleados que realizan un trabajo igual o similar en un área, y que trabajan para el mismo supervisor, aunque no siempre, generalmente el supervisor es el líder o jefe del círculo. En su función de líder del círculo el supervisor no da órdenes ni toma decisiones, siendo adoptadas las decisiones de manera grupal. Las reuniones son voluntarias y periódicas, y son entrenados para identificar, seleccionar y analizar problemas y posibilidades de mejora relacionados con su trabajo, recomendar soluciones y llevar a cabo su implantación.

**Figura 1. El círculo de calidad**



Se observa que la implantación de un círculo de calidad requiere de la intervención de personas inmersas en diferentes áreas de la producción y que tengan contacto directo con el problema a tratar, de igual manera se requiere que sean problemas medibles y a corto plazo, y algo importante es la intervención de personas de alta gerencia para obtener apoyo constante de la

<sup>5</sup> <http://www.monografias.com/trabajos30/circulos-control-calidad/circulos-control-calidad.shtml>

dirección, los círculos de calidad deben ser considerados como un punto de partida para enfoques más participativos por utilizar en el futuro.

### **1.6 Catorce principios expuestos por Dr. Deming<sup>6</sup>**

Los aspectos más importantes que son necesarios introducir en las empresas que han adoptado el control total de la calidad son expuestos por Dr. Deming en sus catorce puntos o acciones. Estas acciones representan una filosofía básica de administración que es compatible con los métodos estadísticos. A continuación se listan dichos puntos:

- 1) **Crear constancia en el propósito de mejorar:** El propósito es mejorar constantemente los productos y servicios de la empresa, teniendo como objetivo la consecución de la competitividad permaneciendo en el mercado para proporcionar empleo por medio de la innovación, la investigación, el mejoramiento continuo y el mantenimiento adecuado.
  
- 2) **Adoptar la nueva filosofía:** Se trata de adoptar una nueva filosofía de empresa, ya que estamos viviendo una nueva era económica (más ahora), en la que los gerentes deben tomar conciencia de sus responsabilidades y afrontar la cuota de liderazgo que les concierne para lograr el cambio
  
- 3) **No depender de la inspección masiva:** Se debe dejar de depender de la inspección masiva para alcanzar la calidad, hay que eliminar la inspección en masa a través de la integración del concepto de calidad en todo el proceso de producción, lo cual aminora costos y permite aumentar calidad.

---

<sup>6</sup>Bravo, Roberto, Calidad Total. UNED, San José Costa Rica. 1998. Pag. 37-48.,  
<http://www.monografias.com/trabajos12/desorgan/desorgan.shtml>



- 4) **Acabar con la práctica de adjudicar contratos de compra basándose exclusivamente en el precio:** Hay que eliminar la práctica de comprar basándose exclusivamente en el precio, ya que los departamentos de compras tienden a elegir al proveedor con los precios más bajos. En su lugar, se deben concentrar esfuerzos en minimizar los costos totales, creando relaciones sólidas y duraderas con un solo proveedor para cada materia prima, basándose en la fidelidad y la confianza.
- 5) **Mejorar continuamente:** La búsqueda por mejorar debe ser continua, no momentánea ni estática, se deben mejorar los procesos productivos, el servicio y la planeación, además la administración debe inclinarse por la minimización de costos a través de la reducción de pérdidas y disminución de productos defectuosos.
- 6) **Instituir la capacitación en el trabajo:** Se debe instituir el entrenamiento y la capacitación de los trabajadores como una de las tareas del diario acontecer, con esto no sólo se consiguen mejores empleados sino mayores resultados en cuanto a calidad y costos.
- 7) **Instituir el liderazgo:** Las organizaciones deben adoptar e instituir el liderazgo, de manera que la labor de los supervisores o jefes no se limite a dar órdenes o impartir castigos, sino que más bien se convierta en un orientador que le ayude a la gente a hacer mejor su trabajo y que identifique quiénes son las personas que necesitan mayor ayuda para hacerlo.

- 8) **Desterrar el temor:** Las firmas deben desterrar el temor y el miedo de todos sus niveles, hay que generar confianza entre la gente de manera que no sientan temor de opinar o preguntar, esto permite mayor efectividad en el trabajo y permite que las personas se esfuercen porque quieren que la empresa alcance el éxito.
  
- 9) **Eliminar las barreras que existen entre las áreas de Staff y las de línea:** Romper las barreras que existan entre los diferentes departamentos y su gente, no crear competencias que las hagan chocar sino más bien generar la visión de largo plazo que les permita a todos trabajar por conseguir los mismo objetivos, permitiendo así la colaboración y la detección temprana de fallos.
  
- 10) **Eliminar los slogans, las exhortaciones y las metas numéricas para el personal:** Hay que borrar los slogans o las frases preestablecidas, estos no sirven y lo que causan es relaciones adversas que redundan en pérdidas de competitividad y calidad. Los errores, en su mayoría no provienen de los trabajadores, sino del sistema mismo; por eso, es muy frecuente que las amonestaciones generen frustraciones y resentimiento. Las metas numéricas sin un método para alcanzarlas son inútiles.
  
- 11) **Eliminar las cuotas numéricas:** Deben eliminarse las cuotas numéricas, tanto para trabajadores como para gerentes. Las cuotas sólo toman en cuenta los números, no los procesos, los métodos o la calidad y por lo general se constituyen en garantía de baja calidad y altos costos. Las cuotas se deben sustituir con liderazgo, eliminando el concepto de gerencia por objetivos.
  
- 12) **Derribar las barreras que miden el orgullo de hacer bien un trabajo:** Hay que derribar las barreras que le quitan a las

personas el orgullo que les produce su trabajo, eliminando los sistemas de comparación o de méritos, estos sistemas sólo acarrearán nerviosismo y disputas internas.

- 13) **Instituir un programa vigoroso de educación y reentrenamiento:** Se debe establecer un programa interno de educación y automejoramiento para cada quien. El hecho de que la empresa tenga gente capacitada en su organización no es suficiente. Ella debe estar adquiriendo continuamente los nuevos conocimientos y las nuevas habilidades que se necesitan para manejar nuevos materiales y nuevos métodos.
  
- 14) **Tomar medidas para lograr la transformación:** Todos, absolutamente todos los miembros de la organización deben esforzarse por alcanzar la transformación en cuanto a calidad, procesos, productos y servicios, la transformación es el trabajo de todos, pero eso sí, hay que basarse en un equipo que reúna condiciones suficientes de capacidad y liderazgo.

### **1.6.1 Las siete enfermedades mortales del control de calidad<sup>7</sup>**

Además de los 14 principios de Deming aplicables a la mejora continua, existen siete enfermedades en la gerencia que limitan el crecimiento de las empresas, que a continuación se listan:

- 1) **La falta de constancia en el propósito:** La falta de constancia significa la ruina para una empresa, una compañía que carece de constancia en la búsqueda de su propósito no cuenta con planes a largo plazo para permanecer en el negocio.

---

<sup>7</sup> <http://www.gestiopolis.com/canales5/get/gksa/127.htm>

- 2) **El énfasis en las utilidades a corto plazo:** Las empresas hacen malabares financieros que den impresión de solidez económica en vez de mejorar los productos y servicios. El énfasis en las utilidades a corto plazo está alimentado por el temor a una adquisición hostil por otras empresas o por el devastador sistema de apalancamiento para eliminar a un socio.
  
- 3) **Las evaluaciones de méritos o de desempeño individual:** la evaluación de desempeño estimula el desempeño a corto plazo. Desestimula la decisión de correr riesgos, fomenta el miedo, hace que los empleados se enfrente por las mismas recompensas. En resumen, se destruye el trabajo en equipo y se fomenta la rivalidad.
  
- 4) **La movilidad de la alta dirección:** Los gerentes que cambian de un puesto a otro nunca entienden a las compañías para las cuales trabajan y nunca están ahí el tiempo suficiente para llevar a cabo los cambios a largo plazo que son necesarios para garantizar la calidad y la productividad. La gente necesita tiempo para aprender a trabajar en grupo, la causa principal es la falta de satisfacción con el trabajo.
  
- 5) **El administrar la compañía basándose solo en las cifras visibles:** Las cifras visibles son importantes porque dan idea a los administradores de la ruta por la cual se encuentra la empresa, pero las cifras que no se conocen y no se pueden conocer son incluso más importantes.
  
- 6) **Los costos médicos excesivos:** los costos médicos, medicinas y hospitales normalmente son altos e implican directamente tiempo de recuperación de algún trabajador, lo cual deriva en baja producción en la empresa y gastos adicionales como horas extras de alguien más. El verdadero esfuerzo de costo y tiempo se debe dar en la medida

preventiva de cada persona. Los chequeos periódicos realizados en forma ordenada colaboran como prevención mas que como corrección.

- 7) **Costos excesivos de responsabilidad legal:** Costos excesivos de garantía fomentados por abogados que cobran con base en honorarios altos para los casos imprevistos de demandas.

### **1.7 Obstáculos del control de calidad**

Se puede tomar como obstáculo cualquier acción que impida la realización de los 14 principios expuestos anteriormente o que ayude a las siete enfermedades; esto nos trae a calificar como obstáculos incluso algunos de los catorce principios si estos se aplican mal o si tratan de ponerse a operar sin el apoyo adecuado.

Otro problema citado por el Dr. Deming es la “copia de modelos”, que algún administrador pudo tomar de alguna lectura u otra fuente y tratar de imponerlo de esta forma en el país, a personas de otra cultura o en una realidad totalmente diferente.

### **1.8 El control de la calidad a través de herramientas estadísticas<sup>8</sup>**

La calidad del producto fabricado está determinada por sus características de calidad, es decir, por sus propiedades físicas, químicas, mecánicas, estéticas, durabilidad, funcionamiento, etc. que en conjunto determinan el aspecto y el comportamiento del mismo. El cliente quedará satisfecho con el producto si esas características se ajustan a lo que esperaba, es decir, a sus expectativas previas.

---

<sup>8</sup> <http://www.monografias.com/trabajos7/herba/herba.shtml>

El uso de herramientas para medir la calidad nos permite responder a la pregunta de ¿Para qué se miden las características de calidad?. El análisis de los datos medidos permite obtener información sobre la calidad del producto, estudiar y corregir el funcionamiento del proceso y aceptar o rechazar lotes de producto. En todos estos casos es necesario tomar decisiones y estas decisiones dependen del análisis de los datos. Los valores numéricos presentan fluctuación aleatoria y por lo tanto para analizarlos es necesario recurrir a técnicas estadísticas que permitan visualizar y tener en cuenta la variabilidad a la hora de tomar las decisiones.

Algunas de estas técnicas fueron agrupadas y se conocen como “Las 7 Herramientas Estadísticas de Calidad” y estas son:

- Hoja de Recolección de Datos
- Histograma
- Diagrama de Pareto
- Diagrama de Causa – Efecto
- Diagrama de Dispersión
- Estratificación
- Gráfico de Control

### 1.8.1 Hoja de recolección de datos

En el control estadístico de la calidad se hace uso con mucha frecuencia de las hojas de verificación, ya que es necesario comprobar constantemente si se han recabado los datos solicitados o si se han efectuando determinados trabajos.

El esquema general de estas hojas es la siguiente: en la parte superior se anotan los datos generales a los que se refiere las observaciones o verificaciones a hacer en la parte inferior se transcribe el resultado de dichas observaciones y verificaciones.

**Figura 2. Hoja de recolección de datos.**

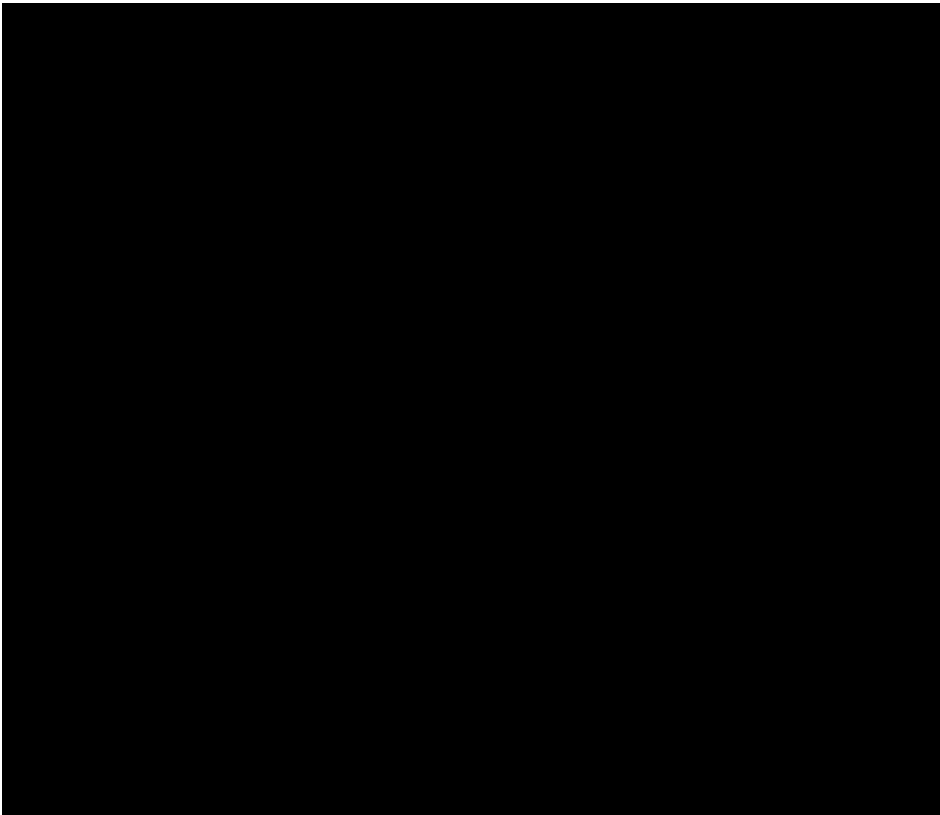
<b>Título de la institución</b>				
<b>Servicio o área que se evalúa:</b>				
Orden	Descripción de Evaluación	Fecha Inicio	Fecha Final	Observación
<u>VoBo.</u> _____				

### 1.8.2 Histograma

El histograma ordena las muestras, tomadas de un conjunto, de tal forma que se vea de inmediato con qué frecuencia ocurren determinadas características que son objeto de observación. El histograma en el control estadístico de calidad se utiliza para visualizar el comportamiento del proceso con respecto a determinados límites.

El histograma se construye tomando como base un sistema de coordenadas. El eje horizontal se divide de acuerdo con las fronteras de clase. El eje vertical se gradúa para medir la frecuencia de las diferentes clases. Estas se presentan en forma de barra que se levantan sobre el eje horizontal.

**Figura 3. Histograma de frecuencias**





### 1.8.3 Diagrama de Pareto

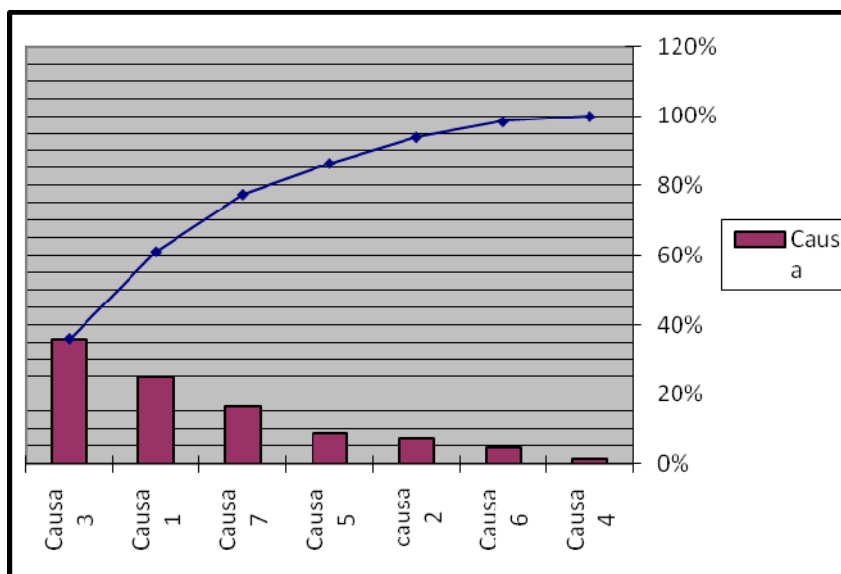
Se utiliza con el propósito de visualizar rápidamente qué factores de un problema y qué causas o qué valores en una situación determinada, son los más importantes y por consiguiente, cuáles de ellos hay que atender en forma prioritaria, a fin de solucionar el problema o mejorar la situación.

El diagrama de Pareto consiste en un gráfico de barras similar al histograma que se conjuga con una ojiva o curva de tipo creciente y que representa en forma decreciente el grado de importancia o peso que tienen los diferentes factores que afectan a un proceso, operación o resultado.

La estructura del Diagrama de Pareto es:

- a) Sobre el eje horizontal se muestran barras de la misma dimensión, en cuya base debe llevar el nombre del efecto o problema. Estas barras son de ordenadas de izquierda a derecha y de mayor a menor frecuencia en cuanto a su aparición.
- b) Sobre el eje vertical izquierdo se muestra la frecuencia de aparición de efecto o problema
- c) Sobre el eje vertical derecho se gráfica el porcentaje relativo acumulado (eje para trazar la ojiva o curva).

**Figura 4. Diagrama de Pareto**



#### **1.8.4 Diagrama de Causa – Efecto (conocido como espina de pescado)**

También conocido como Esqueleto de pescado o Diagrama de Ishikawa: es una herramienta sistémica para la resolución de problemas que permiten apreciar la relación existente entre una característica de calidad (efecto) y los factores (causas) que la afectan, para así poder definir las causas principales de un problema existente en un proceso.

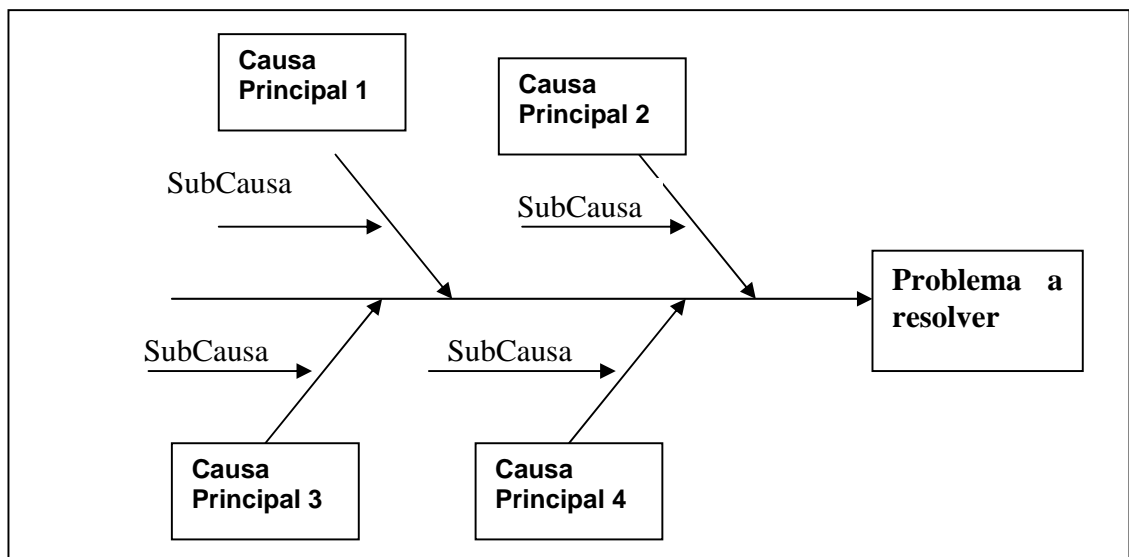
Las causas son determinadas pensando en el efecto que tiene sobre el resultado, indicando por medio de flechas la relación lógica entre la causa y el efecto.

La primera sección está constituida por una flecha principal hacia la que convergen otras flechas, consideradas como ramas del tronco principal, y sobre las que indiquen nuevamente flechas más pequeñas, las sub-ramas. En esta primera sección quedan, pues, organizados los factores casuales.

La segunda sección está constituida por el nombre de la característica de calidad. La flecha principal de la primera sección apunta precisamente hacia este nombre, indicando con ello la relación casual que se da entre el conjunto de factores con respecto a la característica de calidad.

El diagrama de causa-efecto es aplicable en cualquier proceso (administrativo, productivo, etc.) en donde se requiera solucionar un problema o en donde se desee implementar una mejora.

**Figura 5. Diagrama de Causa - Efecto**



### 1.8.5 Diagrama de dispersión

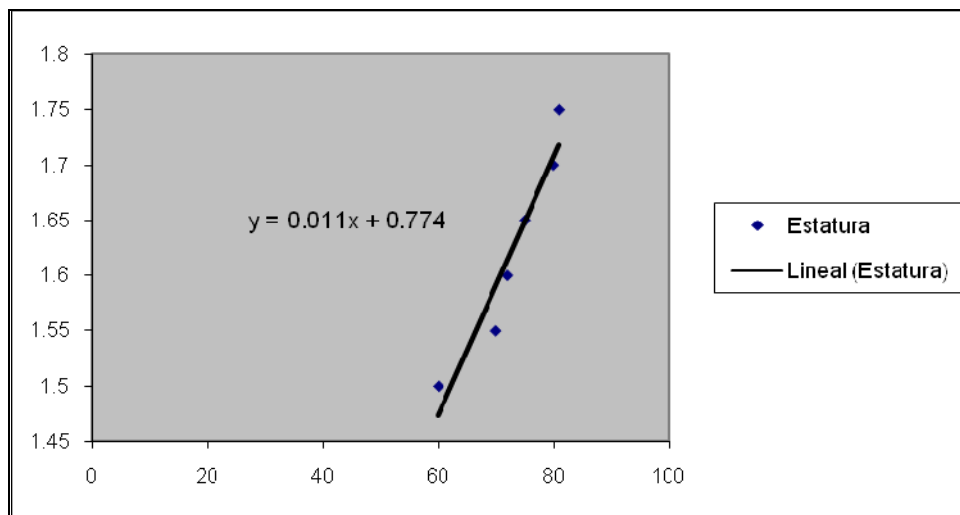
Un diagrama de dispersión se usa para estudiar la posible relación entre una variable y otra; también sirve para probar posibles relaciones de causa-efecto; en este sentido no puede probar que una variable causa a la otra, pero deja más claro cuándo una relación existe y la fuerza de esta relación.

Dadas 2 variables "X" & "Y", se dice que existe una correlación entre ambas si cada vez que aumenta el valor de "X" aumenta proporcionalmente el valor de "Y" (Correlación positiva) o si cada vez que aumenta el valor de "X" disminuye en igual proporción el valor de "Y" (Correlación negativa).

La relación entre dos tipos de datos pueden ser:

- Una característica de calidad y un factor que inciden sobre ella.
- Dos características de calidad relacionadas, o bien dos factores relacionados con una sola característica.

**Figura 6. Diagrama de dispersión**

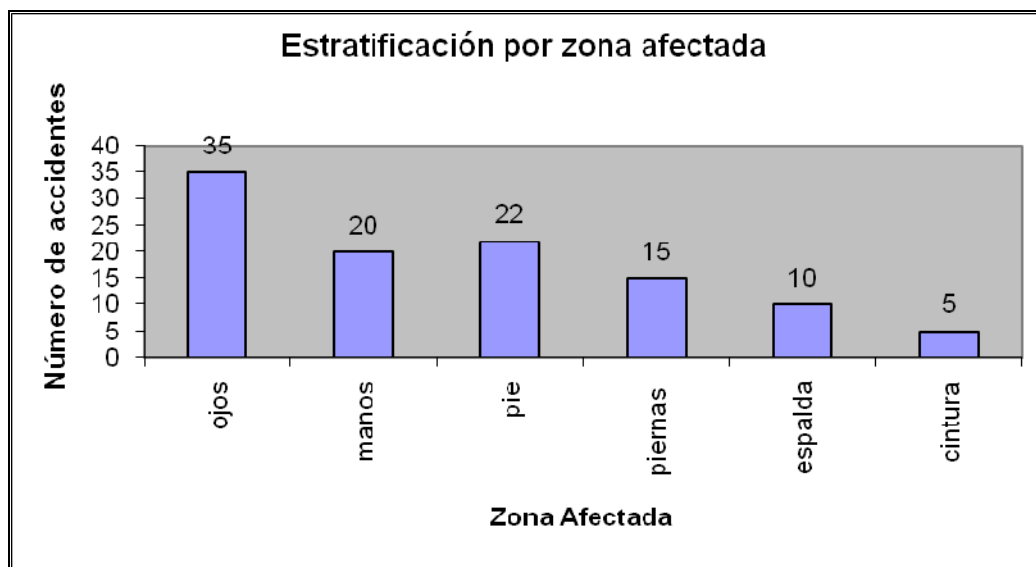


### 1.8.6 Estratificación

Es la herramienta estadística que clasifica los datos en grupos con características semejantes. A cada grupo se le denomina estrato. La clasificación se hace con el fin de identificar el grado de influencia de determinados factores o variables en el resultado de un proceso.

La situación que en concreto va a ser analizada determina los estratos que se van a utilizar. Por ejemplo, si se desea analizar el comportamiento de los operarios, éstos pueden estratificarse por edad, sexo, experiencia en el trabajo, capacitación recibida, etc. La forma más común de presentar la estratificación es el histograma.

**Figura 7. Estratificación**



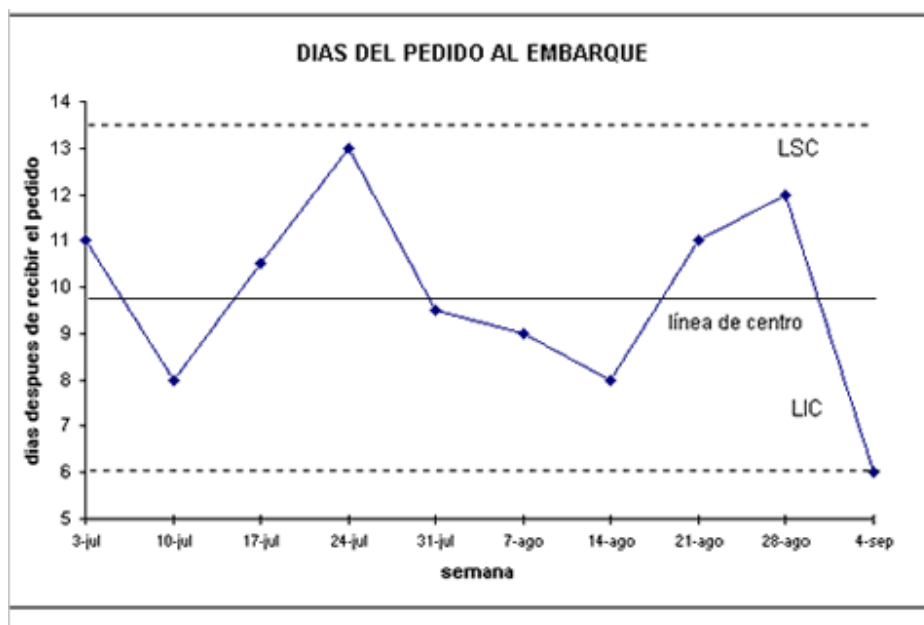
### 1.8.7 Corridas y gráficas de control

Las corridas permiten evaluar el comportamiento del proceso a través del tiempo, medir la amplitud de su dispersión y observar su dirección y los cambios que experimenta. Estas se elaboran utilizando un sistema de coordenadas, cuyo eje horizontal indica el tiempo en el que quedan enmarcados los datos, mientras que el eje vertical sirve como escala para transcribir la medición efectuada. Los puntos de medición se unen mediante líneas rectas.

Las gráficas de control son herramientas estadísticas más complejas, que permiten obtener un conocimiento mejor del comportamiento del proceso a

través del tiempo, ya que en ella se transcriben tanto la tendencia central del proceso como la amplitud de su variación. Estas gráficas están formadas por dos corridas en paralelo: una de ellas, la que se coloca en la parte superior, se destina a graficar una medida de tendencia central, que puede ser la media aritmética o la mediana; la otra, colocada en la parte inferior, se destina a graficar estadísticos que miden el rango de dispersión con respecto a dicha medida central. Estos estadísticos pueden ser “el rango muestral” o “la desviación estándar muestral”.

**Figura 8. Corridas y gráficas de control**



Como se puede observar, la calidad total está inmersa en todas las actividades del hombre y entiende que la calidad total tiene como fin la satisfacción del cliente e involucrar a cada uno de los miembros de la empresa en la garantía de la calidad, del bien o servicio que se pretende brindar.

El análisis de todos los factores, ayuda a llenar los requisitos crecientes de productos y consumidores y así vencer cada uno de los obstáculos, con el fin de alcanzar el premio deseado; es decir; un producto de confiabilidad que satisfaga las expectativas del cliente.

Como se puede observar la idea de la calidad es la satisfacción del cliente, la persona que adquiere el bien o servicio, en los últimos años el software hecho a la medida ha adquirido mayor importancia en las empresas y mas aún con los adelantos tecnológicos, que día a día permiten a mas empresas se interconecten y ofrezcan información a los usuarios, con todo esto, el concepto de calidad dentro de los sistemas informáticos es un tema latente que permite la entrega de software en las fechas exactas y en la forma que el cliente lo desea.

El seguimiento de metodologías de control de calidad para el desarrollo de software brinda un gran beneficio, permitiendo llevar un control de las etapas del desarrollo y de los resultados que se han ido obteniendo, por lo cual es de gran beneficio tener conceptos y técnicas de control de calidad para la verificación de el desarrollo de software y así alcanzar el objetivo primordial en un proceso producción, el cual es satisfacer las expectativas y necesidades del cliente.





## 2 INGENIERÍA DEL SOFTWARE

### 2.1 Antecedentes

Durante el transcurso de las primeras décadas de la informática, el principal desafío fue desarrollar hardware para computadora, y así reducir el costo de procesamiento y almacenamiento de información. Los avances que ha tenido la tecnología hoy día, como avances en la microelectrónica han dado un mayor potencial de cálculo y una reducción de costos.

El potencial de las computadoras fue creciendo, agregando capacidad de procesamiento y almacenamiento, esto, permitió tener computadoras personales cada vez mas poderosas. Y así, el software se convirtió en el mecanismo que facilitó la utilización de este potencial.

El desarrollo de software creó la necesidad de enfoques sistemáticos para el desarrollo de estos productos, esto llegó a patentizar dichos enfoques en el año 1960, durante este período aparecieron las computadoras de tercera generación y esto llevó al desarrollo de técnicas de programación, como la multiprogramación y de tiempo compartido.

Para considerar los problemas crecientes de la tecnología del software, se realizaron dos reuniones, una en Alemania y otra en Roma (1968 y 1969), para estimular el interés hacia los aspectos técnicos y administrativos utilizados en el desarrollo y mantenimiento de productos de software.

Así el término en inglés "Software Engineering" (Ingeniería de Software o Ingeniería de Productos de Programación), fue utilizado por primera vez como término de estimulación en dichas reuniones, y continuó usándose en las siguientes reuniones de trabajo que discutieron la importancia de crear una disciplina formal para el desarrollo de productos de software.

Así, la Ingeniería del Software se convirtió en un área de la informática, que ofrece métodos y técnicas para desarrollar y mantener software de calidad y resolver problemas de todo tipo. La ingeniería del software trata con áreas muy diversas de la informática y de las ciencias de computación, tales como el desarrollo de compiladores, sistemas operativos, etc., de igual forma abarca áreas como Negocios, Medicina, Producción, etc.

Actualmente, el software ha superado al hardware y ha sido la clave de éxito de muchas empresas que utilizan los sistemas informáticos para llevar su negocio, el software es el factor que marca la diferencia. Hoy día el principal desafío es mejorar la calidad de las soluciones basadas en computadoras.

Esta nueva disciplina tiene como función la mejora de calidad y el aumento en productividad y satisfacción profesional de los ingenieros, así como entregar un producto que sea bien desarrollado para satisfacción de los usuarios.

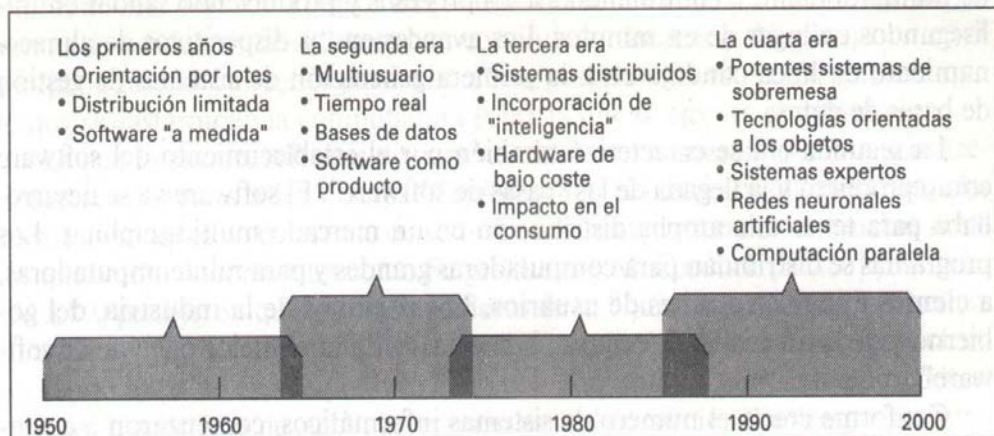
### **2.1.1 La evolución del software**

Como se puede observar, el software ha venido evolucionado conforme la tecnología ha ido cambiando, se ha mejorado su desempeño en el hardware, reduciéndose en tamaño y en costos. Esto ha llevado que el software se vuelva más complejo y las computadoras puedan realizar más operaciones en paralelo.

Como se puede observar, la figura 8 brinda una descripción de la evolución que el software ha sufrido desde el surgimiento de las grandes computadoras hasta la llegada de los componentes microelectrónicos.

La programación de computadoras se convirtió en un arte, en el que no se contaba con métodos sistemáticos de desarrollo.

**Figura 9. Evolución del software**



- En el primer período, lo normal fue que el hardware fuera de propósito general y ejecutara programas realizados a la medida. La programación se hacía de una manera por lotes. Y el software como producto para venta, carecía de interés, la mayoría de software se desarrollaba y era utilizado por la misma persona u organización, la misma persona escribía, ejecutaba, y si fallaba, lo depuraba.
- En la segunda era, surge la multiprogramación y los sistemas multiusuario, que proveyeron de nuevos conceptos, lo que tenía que ver en la interacción hombre-máquina. Se poseían sistemas que recogían información, analizaban y se brindaban resultados en tiempo real, de igual forma se analizan procesos en segundos y se producían salidas en milisegundos, se dió principio a los

primeros sistemas de bases de datos, y el advenimiento de las casas de software y se vio este como un producto.

- La tercera era se caracterizó por el surgimiento del procesamiento distribuido, microprocesadores, computadoras personales y computadoras realizando proceso simultáneos comunicándose entre si. Surgimiento de redes locales y de área global para accesos instantáneos a los datos. Se empieza la fabricación de hardware estándar y a suministrar software con cada uno de los dispositivos de hardware que se compra, lo que marco la diferencia. Se empezó a gastar más dinero en software que en hardware.
- La cuarta era es en la que nos encontramos, programación orientada a objetos, las que están reemplazando el desarrollo convencional, las técnicas de cuarta generación para el desarrollo de software han cambiando la forma en que construyen los programas de computadoras. El surgimiento de sistemas expertos y de inteligencia artificial ya están en sistemas informáticos en organizaciones para un rango de problemas el mundo real. Las redes neuronales artificiales han abierto posibilidades para reconocimientos de formas y habilidades de procesamiento de información que semejan la forma en que los humanos lo hacen.

Como una respuestas inmediata a la crisis que ha venido surgiendo, con el creciente cambio en el desarrollo de software, muchas organizaciones han adoptado prácticas de ingeniería del software que faciliten el desarrollo de

software que sea cada vez mas eficiente y satisfagan sus necesidades de la forma mas eficiente, en tiempo y a bajos costos.

### 2.1.2 Características del software

Se habla acerca del software, pero para comprender qué es el software y posteriormente la ingeniería del software, es de suma importancia examinar algunas características acerca del software que lo hace diferente de cualquier otra cosa que el ser humano sea capaz de construir.

El software es un elemento lógico, en lugar de físico, por lo que tiene características distintas a las del hardware, y se listan a continuación:

- ***El software se desarrolla, no se fabrica:*** la calidad en el software se desarrolla a través de un buen diseño, y los costos de fabricación de software se encuentran en la ingeniería. Esto significa que los proyectos de software no se pueden gestionar como si fueran proyectos de fabricación.
- ***El software no se estropea:*** el software no se ve afectado por elementos externos (polvo, cambios de corriente etc.) que afecten su funcionamiento, sin embargo, si se detectara alguna falla, y se hacen cambios, es bastante probable que se introduzcan nuevos defectos, haciendo que el software se deteriore debido a dichos cambios.
- ***La mayoría del software se construye a la medida, en vez de ensamblar componentes existente:*** Al momento de desarrollarse hardware, los diseñadores buscan únicamente en catálogos los suministros o componentes electrónicos basados en algún numero de referencia, solicitarlos y posteriormente el ensamblaje del mismo. Existe software a la venta pero solo como unidad completa, no como componente que puede reensamblarse en nuevos programas. Todo esto porque las necesidades

individuales de las empresas hacen que el software que se desarrolla este hecho en base a especificaciones propias de cada organización.

### 2.1.3 La ingeniería del Software

Los adelantos tecnológicos que hasta el día de hoy se han visto, han facilitado la vida del ser humano, el software se ha convertido poco a poco en una pieza clave, proporcionando información, procesándola y en algunos casos ayudando a los administradores en la toma de decisiones.

La ingeniería del software es una disciplina que ofrece métodos y técnicas para desarrollar y mantener software de calidad y que ayude a resolver problemas de todo tipo, en el libro del IEEE titulado **Standard Glossary of Software Engineering** se define la ingeniería del software como “el enfoque sistemático para el desarrollo, operación, mantenimiento y eliminación de software<sup>9</sup>”, se define software como, “aquellos programas, procedimientos, reglas y documentación posible asociada con la computación, así como los datos pertenecientes a la operación de un sistema de cómputo”.

Para este trabajo de graduación, se utiliza la siguiente definición:

**La ingeniería de software es la disciplina tecnológica y administrativa dedicada a la producción sistemática de productos de programación, que son desarrollados y modificados a tiempo y dentro de un presupuesto definido.<sup>10</sup>**

Como puede observarse la ingeniería del software es la aplicación práctica de conocimiento científico, esto indica que debe de seguir lineamientos

---

<sup>9</sup> [http://standards.ieee.org/reading/ieee/std\\_public/description/se/610.12-1990\\_desc.html](http://standards.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html)

<sup>10</sup> Definición textual tomada de Ingeniería de software, Richard Fairley, McGrawHill, pagina 2.

del método científico en el diseño y construcción de programas de computadora, de igual forma debe de llevarse un adecuado control de la documentación asociada y requerida con el mismo, que faciliten el proceso de operación y mantenimiento del mismo.

Para aplicar la Ingeniería del Software adecuadamente, se debe definir un proceso de desarrollo de software. El trabajo que se asocia a la Ingeniería del Software se puede dividir en tres fases genéricas, definición, desarrollo y mantenimiento, que son aplicables al desarrollo, independientemente del área de aplicación, del tamaño del producto o de la complejidad del mismo.

A continuación se listan estas tres fases:

- La fase de definición se centra en el "qué", es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento, interfaz, restricciones, y criterios de validación se necesitan para definir un sistema correcto. Por tanto, ha de identificarse los requisitos clave del sistema a desarrollar.
- La fase de desarrollo se centra en el "cómo". Es decir, durante esta fase el ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, poner en práctica detalles de procedimientos, personalización de interfaces, la traducción del diseño en un lenguaje de programación y cómo han de realizarse las pruebas.

Los métodos aplicados durante la fase de desarrollo variarán, pero producirán tres pasos concretos.

- Diseño del software
  - Generación de código
  - Prueba del software
- 
- La fase de mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas, a medida que evoluciona el entorno del software, y a cambios debidos a las mejoras producidas por los requisitos siempre cambiantes del cliente. Esta fase vuelve a aplicar los pasos de definición y desarrollo, pero en el contexto del software ya existente. Contiene cuatro tipos de cambios:
    - **Corrección:** El mantenimiento correctivo modifica el software para corregir los defectos.
    - **Adaptación:** El mantenimiento adaptador produce modificaciones en el software para acomodarlo a los cambios de su entorno externo.
    - **Mejora:** El mantenimiento de perfección lleva al software más allá de sus requisitos funcionales originales.
    - **Perfectivo:** Amplía el software más allá de sus requisitos funcionales originales, o sea hace modificaciones al mismo con el objetivo de perfeccionar cada vez mas el conjunto de operaciones para la cual fue realizado.



Las fases y los pasos relacionados descritos en la visión genérica de la Ingeniería del Software, se complementan con un número de actividades protectoras. Entre las actividades típicas de esta categoría se incluyen:

- Seguimiento y control del proyectos de software
- Revisiones técnicas formales
- Garantía de calidad del software
- Gestión de configuración del software
- Preparación y producción de documentos
- Gestión de reutilización
- Mediciones
- Gestión de riesgos

Todas las actividades de protección se aplican en todo lo largo del proceso de software, en el cual se establece un marco de trabajo que será aplicable a cualquier proyecto, sin importar su tamaño o complejidad.

## **2.2 Metodologías, técnicas y herramientas de la ingeniería del software**

La mayoría de definiciones de Ingeniería del Software orientan a la persona encargada, de la importancia de aplicar principios de ingeniería para el desarrollo de software. La ingeniería del software abarca tres elementos importantes:

- **Métodos:** indica “cómo” construir técnicamente el software. Los métodos abarcan un amplio numero de tareas tales como: planificación y estimación de proyectos, análisis de los requisitos del sistema y de software, diseño de estructuras de datos, arquitectura de programas y procedimientos, codificación, prueba y mantenimiento.

- **Herramientas:** son las que ayudan a que los métodos se puedan llevar a cabo, y el control de los mismos. Algunas veces las herramientas se pueden integrar y cuando esto se realiza, existe la posibilidad que los datos generados se utilicen por otra, a esto se le conoce como “*ingeniería del software asistida por computadora*<sup>11</sup>” (del inglés CASE, *Computer Aided Systems Engineering*).
- **Procedimientos:** son los que definen la secuencia en la que se aplican los métodos, brindando a la persona encargada del proyecto (gestor de proyecto) información para la evaluación del desarrollo. De esta manera los procedimientos son los que unen las herramientas y los métodos para la elaboración de un proyecto de software.

Como se puede observar, lo anterior facilita a la persona encargada de desarrollo de software el control de dicho proceso, y suministra las bases para la construcción de software de alta calidad de una forma productiva.

Para la resolución de problemas reales, un ingeniero de software o equipo de ingenieros; debe incorporar una estrategia de desarrollo que acople al proceso, los métodos y las herramientas. Esta estrategia a menudo se le conoce con el nombre de “**modelo de proceso de software o paradigma de ingeniería del software**”, la cual permite la selección de un modelo, según la naturaleza del proyecto y del tipo de aplicación, ayudando al equipo de desarrollo a encontrar las insistencias, redundancias y las partes que se construyen, de todo esto, la elección de un modelo ayuda a las personas encargadas del desarrollo a comprender la brecha que existe entre lo que deber ser y lo que es.

---

<sup>11</sup> [http://es.wikipedia.org/wiki/Ingenier%C3%ADa\\_asistida\\_por\\_computadora](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_asistida_por_computadora)

A continuación, se presenta un conjunto de modelos de la ingeniería de software, útiles en el proceso de desarrollo de productos de software, que tienen como fin el garantizar el desarrollo de un software eficiente y de calidad.

## **2.3 Metodologías**

### **2.3.1 Modelo en cascada o ciclo de vida clásico**

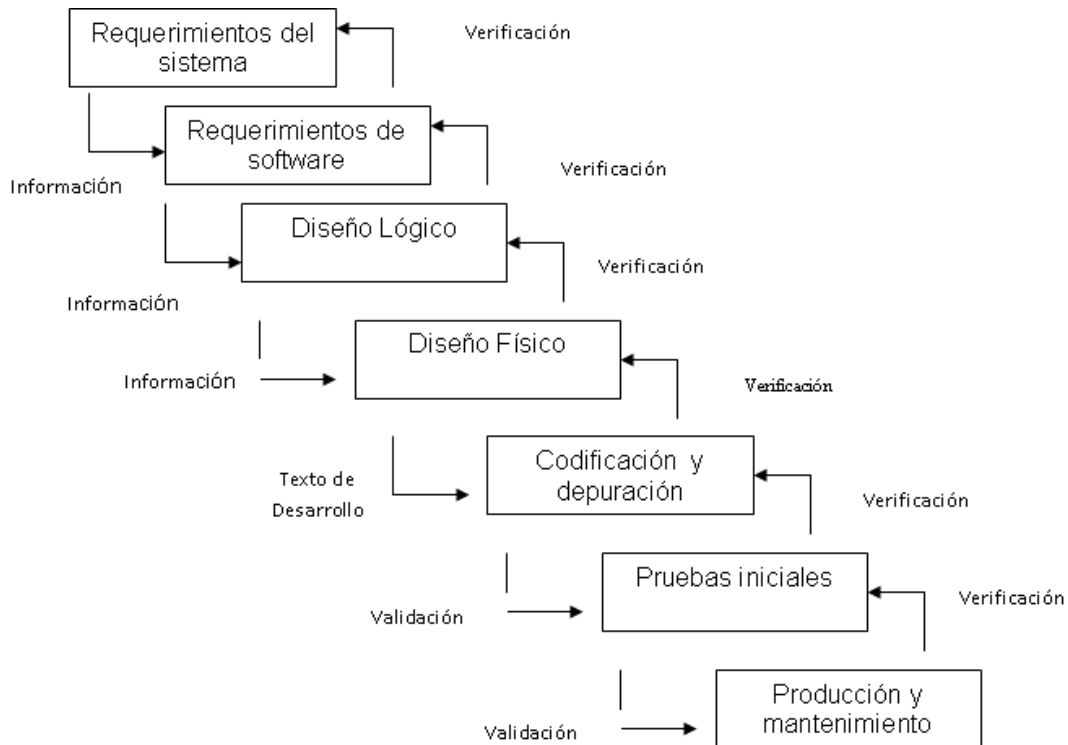
En la ingeniería de software los proyectos siempre tienen definidos un inicio y un final, en el modelo en cascada, el desarrollo esta dividido en fases, cada fase debe de completarse antes de iniciar la siguiente. Estas permiten a la persona encargada de desarrollar software, saber qué va a hacer y en qué fase se encuentra.

Las actividades que se realizan entre el inicio y el final de un una etapa son utilizados por las subsiguientes y debe de existir una comunicación constante entre ellas; de manera que se pueda validar y verificar la información que se obtiene y genera cada etapa. Al finalizar las etapas de un proyecto, se ha recolectado toda la información necesaria y se ha procesado y transformado en un producto o servicio final, al cual se le llama **Ciclo de Vida del Software**.

El Ciclo de Vida o Modelo en Cascada puede ser definido en siete fases:

- Requerimientos del sistema
- Requerimientos de software
- Diseño Lógico
- Diseño Físico
- Codificación y depuración
- Pruebas iniciales
- Producción y mantenimiento

**Figura 10. Ciclo de vida del software con comunicación entre etapas**



### 2.3.2 El modelo en espiral

El modelo en espiral, fue creado para utilizar las mejores características del modelo de prototipos y de ciclo de vida de software incluyendo un elemento que se encarga de analizar el riesgo en cada una de las interacciones.

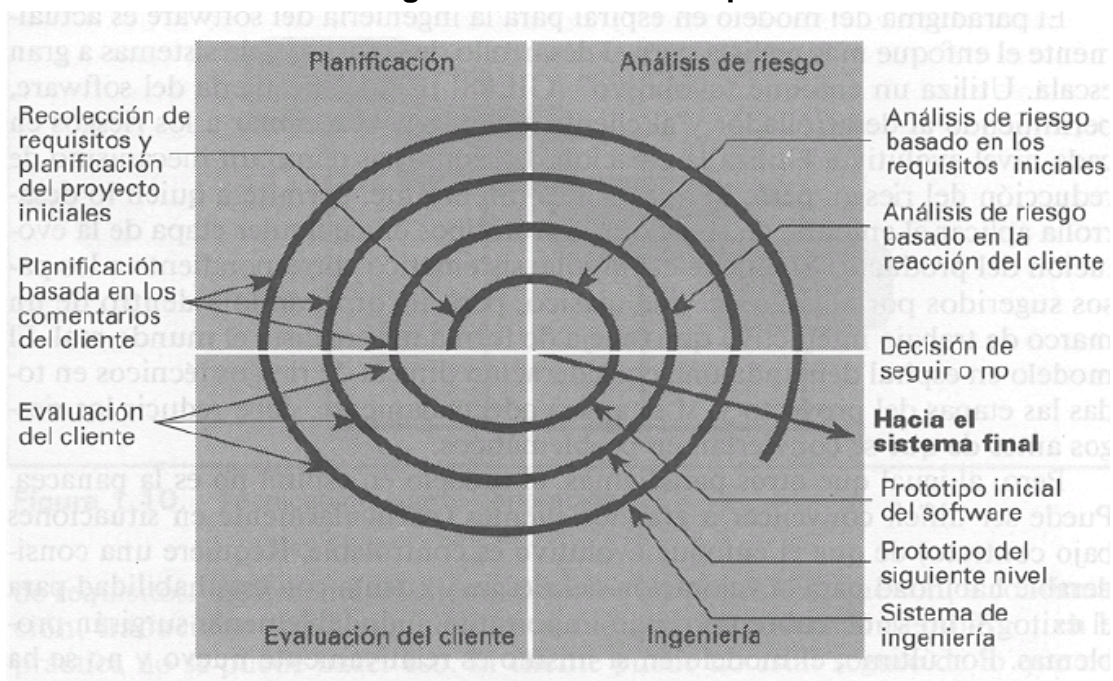
Este modelo utiliza cuatro cuadrantes, en la cual se definen cuatro actividades principales las cuales se describen a continuación:

- Planificación
- Análisis de riesgo
- Ingeniería
- Evaluación del cliente

Con cada interacción dentro de la espiral se va construyendo una nueva versión de software, cada vez mas completa.

El análisis de riesgo es un factor importante dentro del desarrollo de software, y el modelo de espiral provee la capacidad de poder evaluar riesgos y poder determinar si se debe continuar con el desarrollo, y si el riesgo es demasiado grande se puede dar por culminado el proyecto.

**Figura 11. Modelo en espiral**



El modelo en espiral es el enfoque más realista para el desarrollo de software y sistemas a gran escala. Dicho modelo utiliza el enfoque evolutivo que permite al cliente y al desarrollador entender y reaccionar a los riesgos en cada nivel de la espiral. El modelo en espiral demanda una consideración directa de riesgos técnicos en todas las etapas del proyecto y, si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en realidad.

### 2.3.3 El modelo desarrollo rápido de aplicaciones

Es un modelo de desarrollo de software, que utiliza como base el modelo en cascada, pero que enfatiza un ciclo de desarrollo extremadamente corto. Es funcional en el momento que se poseen claros los requisitos, y especialmente se puede dividir el sistema en módulos pequeños, ya que enfatiza en el uso de objetos o componentes que se pueden reutilizar.

El desarrollo rápido de aplicaciones, permite, al equipo crear un sistema completamente funcional dentro de períodos corto, normalmente de 60 a 90 días, frecuentemente con algunas sesiones.

### 2.3.4 El modelo de ensamblaje de componentes

Con el advenimiento de la tecnología orientada a objetos, la utilización de clases que encapsulan tanto datos como los algoritmos que se utilizan para manejar los mismos. Si se diseñan y se implementan adecuadamente, las clases<sup>12</sup> son reutilizables por las diferentes aplicaciones y arquitecturas de sistemas basados en computadora.

En primer lugar se identifica las clases candidatas examinando los datos que se van a manejar por parte de la aplicación. Si estas clases han sido creadas por programas anteriores se almacenan en una Biblioteca de Clases o depósito. Acto seguido, se determina cuáles de ellas ya existen a fin de reutilizarlas. Si no se precede a su creación.

---

<sup>12</sup> Una **clase** es la estructura de un objeto, es decir, la definición de todos los elementos de que está hecho un objeto. Un objeto es, por lo tanto, el "resultado" de una clase. En realidad, un objeto es una **instancia** de una clase, por lo que se pueden intercambiar los términos **objeto** o **instancia** (o incluso *evento*).

## **2.4 Técnicas**

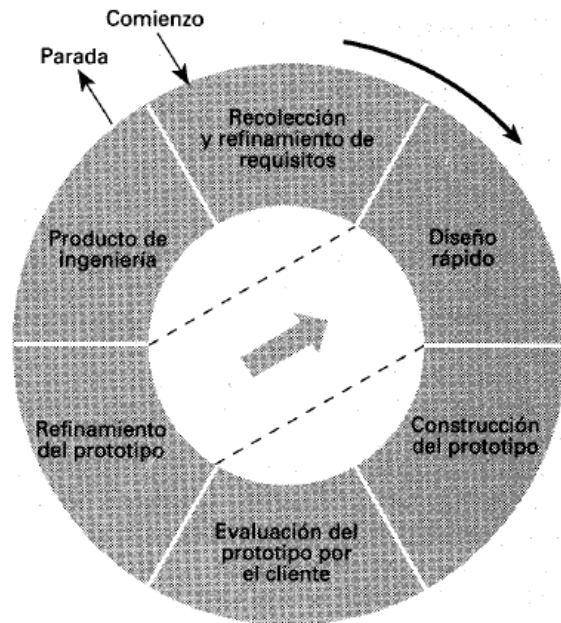
### **2.4.1 Técnica de construcción de prototipos**

Un cliente generalmente define un conjunto de objetivos para el software que desea, pero, en algunos casos no identifica los requisitos detallados, o en otros casos el programador no está seguro de la información o del proceso que realiza algún algoritmo o de la forma en que su programa interactuará con otro software (sistema operativo), o de la interacción hombre- máquina.

La construcción de prototipos facilita al programador la creación de un modelo que utilizará para evaluar los problemas antes mencionados, y le permitirá al desarrollador y cliente la definición de objetivos globales para el software, identificar los requisitos y las áreas en las cuales es obligatorio una mayor definición. De esto surgirá un diseño. Este representa los aspectos de entrada y salida que serán visibles para el usuario/cliente, y que lleva a la construcción de un modelo, el cual será el prototipo que el usuario/cliente evaluará y utilizará para redefinir los requisitos del software que en realidad se desarrolla. La interacción ocurre cuando el prototipo satisface las necesidades del cliente y a la vez permite que el desarrollador comprenda mejor lo que necesita hacer.

Aunque pueden aparecer problemas, la construcción de prototipos es un modelo efectivo para la Ingeniería del Software. La clave está en definir al comienzo las reglas, para que el desarrollador y el cliente sepan que el desarrollo de un prototipo sirve únicamente como mecanismo de definición de requisitos. Posteriormente debe ser descartado y debe construirse el software real, sin perder de vista la calidad.

**Figura 12. Construcción de prototipos**



#### **2.4.2 Técnicas de cuarta generación**

El término técnicas de cuarta generación (4GL's), es un amplio número de herramientas de software, que proveen al programador la habilidad para ejecutar instrucciones o comandos, facilitando al ingeniero de software, la creación de programas de una forma rápida. La herramienta generará automáticamente el código fuente basado en el conjunto de comandos o instrucciones que el programador brindo a la herramienta.

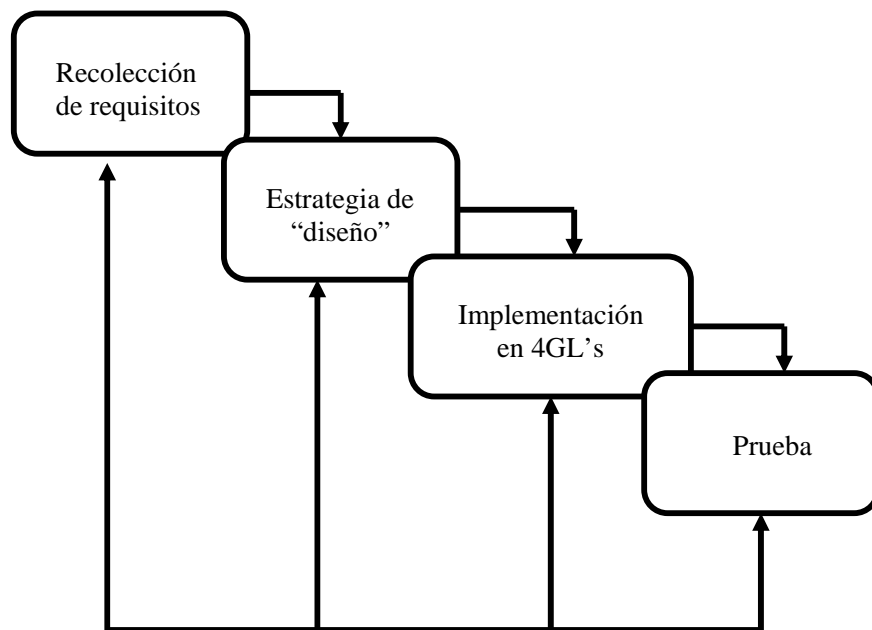
Actualmente existen diferentes herramientas de cuarta generación dentro de las cuales encontramos algunas que proveen: lenguajes no procedimentales para consulta a base de datos, generación de informes, manipulación de datos, interacción y definición de pantallas y generación de códigos, capacidades gráficas de alto nivel y capacidad de hojas de calculo.



Cada una de estas herramientas existen, pero solo son para dominios de aplicación muy específicos. No existe hoy disponible un entorno de herramientas de cuarta generación que pueda aplicarse con igual facilidad a todas las categorías de aplicaciones de software.

Las técnicas de cuarta generación tienen una completa participación en la etapa de desarrollo y programación, sin embargo, participan poco o nada, en la etapa de recopilación y análisis de la misma.

**Figura 13. Técnicas de cuarta generación**



## 2.5 Herramientas

### 2.5.1 Herramientas CASE

Como se ha comentado la ingeniería del software es la aplicación de principios de ingeniería para el desarrollo de productos de software, la cual tiene como fin, aumentar la productividad de los ingenieros de software y así

garantizar calidad en el producto y obtener un mejor control sobre el desarrollo de software.

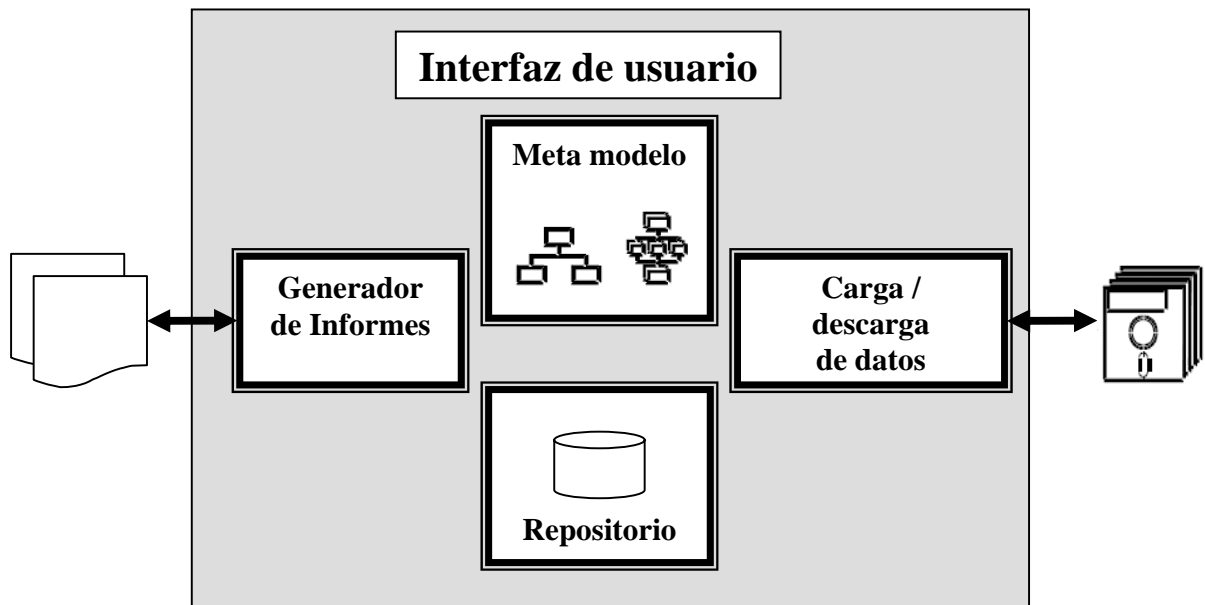
Las herramientas CASE (Computer Aided Systems Engineering) son la automatización de los procesos definidos por la ingeniería de software, para la agilización en el desarrollo, generación de código fuente, gestión de proyectos y la documentación.

Se puede decir que una herramienta CASE está compuesta de los siguientes elementos:

- **Repositorio:** también llamado diccionario, es aquí donde se almacenan los elementos definidos o creados por la herramienta.
- **Metamodelo:** define las técnicas y metodologías soportadas por la herramienta.
- **Generador de informes:** permite obtener toda la documentación que describe el sistema de información desarrollado.
- **Herramientas de carga/descarga de datos:** permite cargar repositorio de la herramienta, con datos provenientes de otros sistemas, o generar a partir de la propia herramienta esquemas de bases de datos, programas etc.
- **Interfaz de usuario:** que constará de editores de texto y herramientas de diseño gráfico, que permitan mediante la utilización de un sistema de ventanas, iconos y menús, definir diagramas, matrices etc.

- **Comprobación de errores:** facilidades que permiten llevar a cabo un análisis de exactitud, integridad y consistencia de los esquemas generados por la herramienta.

**Figura 14. Elementos de una herramienta CASE**



Existen muchas herramientas CASE disponibles en el mercado y fabricados por diferentes proveedores de software, y la diferencia de calidad entre cada una de ellas se encuentra en la verdadera integración, flexibilidad y facilidad de implementación en un ambiente de desarrollo. Por esto la ingeniería del software no puede depender en su totalidad de este tipo de herramientas.

## 2.6 Gestión de la ingeniería del software

La gestión de un proyecto de software esta centrado en las 3p's personal, problema y proceso. El gestor que olvida que el trabajo de la ingeniería del software es un esfuerzo humano, nunca tendrá éxito en la gestión

de proyectos. De igual manera un gestor que no fomenta una buena comunicación con el cliente desde el inicio de la evolución del proyecto, se arriesga a construir una elegante solución para un problema equivocado. Finalmente, si un administrador presta poca atención al proceso corre el riesgo de arrojar métodos, técnicas y uso de herramientas al vacío.

### **2.6.1 Personal**

Para lograr obtener el desarrollo de un producto exitoso, se debe contar con equipo altamente preparado y motivado, por lo cual, no hay que dejar por un lado el factor humano que se encuentre involucrado en dicho desarrollo.

A continuación, se examinan los participantes en el proceso de desarrollo:

#### **2.6.1.1 Los participantes**

Los participantes en el proceso de desarrollo de software se pueden clasificar en cinco categorías:

- **Gestores superiores:** definen los aspectos de negocios que a menudo tienen una influencia significativa en el proyecto.
- **Gestores de proyecto:** son los encargados de planificar, motivar organizar y controlar a los profesionales que realizan el trabajo de desarrollo de software.
- **Profesionales:** que proporcionan las capacidades técnicas necesarias para la ingeniería de un producto o aplicación.

- **Clientes:** son los que especifican los requisitos para la producto que se quiere desarrollar.
- **Usuarios finales:** son los que interactúan con el software una vez que se ha entregado.

Todos los proyectos de software están compuestos por los participantes que se mencionaron anteriormente. Para ser eficaz, el equipo del proyecto debe organizarse de manera que maximice las habilidades y capacidades de cada persona. Esta tarea recae en el jefe de equipo.

#### **2.6.1.2 Los jefes de equipo o gestor de proyecto**

Como se ha mencionado, todo proyecto de software tiene que ver con personas, esto indica, que la gestión de un proyecto de software es intensamente humana. Existen un numero de características importantes que debe cumplir un gestor de proyecto, estas características hacen la diferencia en la culminación de un proyecto exitoso, estas son:

- **Motivación:** la habilidad para motivar, y así obtener los mejores resultados del personal involucrado en el desarrollo.
- **Organización:** la habilidad para moldear procesos existentes o crear nuevos que permita al competo inicial en un proyecto final.
- **Ideas e innovación:** habilidades para motivar al personal para crear y hacer sentir a los desarrolladores creativos aun trabajando dentro de los límites establecidos.
- **Resolución de problemas:** poseer habilidades para la estructuración de soluciones y concentrarse en el problema a resolver.

- **Dotes de gestión:** un buen gestor debe tomar el control, tener confianza para asumir el control cuando se necesite.
- **Incentivos y logros:** todo gestor debe de brindar recompensas a la iniciativa y logros que se hayan logrado, y no penalizar si se corren riesgos controlados.
- **Influencia y construcción de espíritu de equipo:** el gestor debe ser capaz de controlar y promover en espíritu de equipo, aun en situaciones de estrés.

### 2.6.1.3 El equipo de trabajo

Los equipos de desarrollo de software no están conformados con personas que trabajan en forma independiente, los miembros del equipo están organizados para mejorar la obtención de productos de calidad.

La elección de una estructura apropiada para el proyecto depende de los antecedentes y estilos de trabajo de los miembros del equipo, del número de personas del equipo y los estilos de gestión de los clientes y desarrolladores.

Existen diferentes equipos de trabajo dentro de ellos tenemos:

- **Descentralizado democrático:** este equipo no cuenta con un jefe permanente, solo existen coordinadores de tareas a corto plazo. Las decisiones sobre problemas se hacen en consenso del grupo.
- **Descentralizado controlado:** en este equipo se cuenta con un jefe definido que coordina tareas y jefes secundarios que tienen responsabilidades de subtareas, la resolución de problemas sigue siendo

una actividad del grupo, pero la implementación de soluciones se reparte entre subgrupos por el jefe de equipo.

- **Centralizado controlado:** el jefe de equipo se encarga de la resolución de problemas a alto nivel, y la coordinación interna del equipo.

Como se puede observar, la selección del equipo de trabajo depende del tamaño del proyecto y de la complejidad, pero el factor más importante tiene que ver con los canales de comunicación, debido a que la comunicación brindará la herramienta más poderosa, el entendimiento, la confianza, y así generar un software de calidad.

## **2.6.2 El problema**

Un gestor de proyectos se enfrenta al problema de realizar un análisis detallado de requisitos, que proporcionen la información necesaria para la puesta en marcha del desarrollo, y así lograr una estimación y un plan organizado, que al principio no se logra a cabalidad por no poseerse información sólida o los requisitos pueden cambiar regularmente a medida que progresa el proyecto. Para obtener la información detallada para una buena estimación un gestor de proyectos debe tener un análisis por lo menos de unas semanas o meses.

Se debe estudiar el problema desde el inicio del proyecto, por lo menos se debe establecer el ámbito y delimitarlo.

### **2.6.2.1 Ámbito del software**

Es la primera actividad de gestión de un proyecto de software, debe definir respondiendo a lo siguiente:

- **Contexto** ¿cómo encaja el producto en un sistema y que delimitaciones tiene?
- **Objetivos de información** ¿Qué objetos de datos visibles al cliente se obtienen del software?
- **Función de rendimiento** ¿Qué función realiza el software para transformar la entrada en una salida?

El ámbito de un proyecto de software debe ser unívoca y entendible a niveles de gestión y técnico, y los enunciados del ámbito deben estar bien delimitados.

### 2.6.2.2 Descomposición del problema

Es una actividad que se asienta en el análisis de los requisitos del software. Durante el proceso de exposición del ámbito no se intenta descomponer el problema en su totalidad. Más bien, la descomposición se aplica en dos áreas principales:

- La funcionalidad que debe entregarse
- El proceso que se empleará para entregarlo

Al inicio de la planificación de un proyecto, el problema debe de descomponerse en problemas mas pequeños, que resultarán mejor manejables y de esa forma se puede lograr el entendimiento completo del problema a resolver.

Antes de iniciar la estimación del proyecto, se debe de refinar las funciones del software que se describen en el ámbito y de esa forma



proporcionar un mejor detalle para la estimación. Dado que ambos, el costo y las estimaciones de la planificación están orientadas funcionalmente y un pequeño grado de descomposición suele ser útil.

### 2.6.2.3 Problemas por los mitos del software

La mayoría de los profesionales del software, consideran que los mitos son actitudes erróneas que han causado serios problemas a los gestores de proyectos como a los técnicos. De esto, que las viejas actitudes se vuelven hábitos y son difíciles de modificar.

Existen tres tipos de mitos:

- **Mitos de gestión:** la mayoría de gestores se encuentra bajo presión, con el cumplimiento de los presupuestos, entregas a tiempo, hacer que no se retrase el proyecto y el mejoramiento de la calidad. Un mito de gestión se da en el momento en el que el gestor de proyecto cree que al utilizar algo que ya existe, como herramientas, generadores de manuales etc. hará que la calidad del producto se mejore y se de la entrega a tiempo, lo cual la mayoría de veces no es cierto.
- **Mitos del cliente:** la solicitud para el desarrollo de software, viene algunas veces de departamentos internos en la empresa y otras de alguien ajeno, al cual se conoce con el nombre cliente. En muchos casos, el cliente posee una visión diferente del desarrollo y una idea de software que no funciona, entregas después del tiempo, todo esto, hace que el cliente se cree una mala expectativa y quede insatisfecho, debido a que los gestores y responsables de desarrollo hacen poco para corregir dicha visión que el cliente posee.

- **Mitos de los desarrolladores:** muchos desarrolladores poseen ideas y técnicas de desarrollo que se han ido fomentando durante algunas décadas, lo cual provoca que esas viejas formas de desarrollo mueran fácilmente y se adapten nuevas, lo cual provoca una pobre gestión y malas prácticas técnicas, aun existiendo nuevas mejores formas de desarrollo y puesta en marcha de un proyecto de software que garantizan una mejor calidad y entregas en tiempo.

## 2.7 El proceso

El proceso de desarrollo de software es un conjunto de etapas, y cada una de ellas brinda un conjunto de pasos a seguir, y así obtener el producto deseado, una de estas etapas es la selección de la metodología a seguir, como se menciona en la sección 2.3 de este capítulo, dentro de ellas tenemos el modelo en cascada, espiral, prototipos etc. pero el problema es la selección del modelo apropiado para la ingeniería del software.

Se deben de cumplir algunas actividades durante la planificación del proceso, dentro de ellas se encuentran:

- Comunicación con el cliente
- Planificación
- Análisis de riesgos
- Ingeniería
- Evaluación del cliente

Como se puede observar, el proceso es la selección de la metodología adecuada para la elaboración de un proyecto y el seguimiento de sus fases permite ver el desarrollo, el estado actual del proyecto y que es lo que se espera obtener al final de dicho proceso, por lo que es uno de los factores más importante dentro de la gestión de un proyecto de software.

### 3. ISO PARA DESARROLLADORES DE SOFTWARE

#### 3.1 ¿Qué es ISO?<sup>13</sup>

ISO es la Organización Internacional de Estándares y no pertenece a ningún gobierno. Esta organización es el Cuerpo Mundial de la Federación Nacional de Estándares, dicha organización fue fundada en 1947; en la actualidad se encuentra en mas de 100 países y su fin es el promover el desarrollo de la estandarización mundial.

En el año 1987, fue adoptada la serie ISO 9000 como un estándar mundial para productos, el cual fue aprobado y respaldado por el estándar ISO, a este le concierne la administración de calidad y se convierte en un requisito para la competencia global. Todo esto, con el fin de asegurar la satisfacción de los clientes y comprometer a que las empresas apliquen requerimientos que regulen el producto y la realización de mejoras continuas a sus productos.

Toda empresa debe cumplir con un conjunto de requerimientos dependiendo de las actividades a las que se dedica y que van desde el diseño y desarrollo, a la producción, instalación y servicio. Para lo cual, debe seleccionar un sistema para certificarlo en ISO en cuanto a calidad se refiere, dicho sistema debe cumplir los procesos del negocio y la calidad para poder certificarse.

Aunque no se especifique para que tipo de producto esta orientada la norma ISO, puede ser aplicado a todo tipo de producto; el desarrollo y manutención de software no queda fuera de esto, puesto lleva involucrado diseño, desarrollo, manutención y su puesta en marcha, incluyendo la administración del control de calidad. El software es un producto final y como

---

<sup>13</sup> Internacional Organization for Standarization <http://www.iso.org/iso/home.htm>

tal, las empresas pueden certificarse en el estándar ISO y ofrecer su producto a calidad mundial.

### **3.2 Qué es una Norma**

Una norma es un documento que es accesible al público y que contiene especificaciones técnicas u otros criterios para que se usen como reglas, guías o definiciones de características, con el fin de asegurar que materiales, productos, procesos o servicios cumplen los requisitos especificados. Toda norma debe de estar aprobada por un organismo de normalización y no tiene carácter obligatorio.

### **3.3 Cómo trabajan las normas ISO 9000:**

Las normas ISO 9000 proporcionan a la empresa una serie de guías para la debida selección y uso de la ISO 9000, 9001, 9002,9003 y 9004. De la ISO 9001 a la 9003 son modelos que ayudan al aseguramiento de la calidad externa.

Dichas modelos fueron desarrollados para ser utilizados en situaciones contractuales, tales como aquellas entre un cliente y un proveedor. De igual manera las ISO 9004 proporciona guías para uso interno y desarrollar sistemas para garantizar la calidad de los negocios y el aprovechamiento de las oportunidades. Por supuesto, la decisión de que modelo seguir depende de la operación a la que se dedica la organización.

A continuación, se listan las diferentes normas para el aseguramiento de la calidad:

- **ISO 9000:** definiciones, conceptos y términos relacionados con calidad.

- **ISO 9001:** estándar general que comprende el aseguramiento de la calidad en el diseño, desarrollo, manufactura, instalación y servicio de los productos.
- **ISO 9002:** estándar que detalla el enfoque especialmente en manufactura e instalación de productos.
- **ISO 9003:** estándar que detalla la cobertura de inspección final y las pruebas necesarias para completar el producto.
- **ISO 9004:** provee los lineamientos para administrar un sistema de control de calidad para utilizar sistemas de auditoría de calidad.

La **ISO 9000** da las guías para facilitar la aplicación de la regla **ISO 9001** la cual es la encargada de aseguramiento de la calidad en el diseño, desarrollo, manufactura, instalación y servicio de los productos, garantizando la calidad para todo producto.

### **3.4 La certificación ISO**

Una certificación es el procedimiento, mediante el cual una tercera parte diferente e independiente del productor y el comprador, asegura por escrito que un producto, un proceso o un servicio cumple con los requisitos especificados y ayuda también a crear la confianza cliente-proveedor.

El acreditamiento será periódicamente supervisado, para asegurar que el sistema de calidad está siendo mantenido. Muchas certificaciones requieren de auditorías totales después de un tiempo específico que generalmente es tres o cuatro años, si en caso se incurriera en alguna falta dentro del mantenimiento del sistema de calidad, la certificación será suspendida o cancelada.

A continuación se enumeran los criterios básicos a tomar en cuenta para obtener una certificación:

1. Identificar los objetivos generales que la empresa quiere lograr.
2. Identifique lo que otras personas esperan.
3. Poseer información de la familia de normas ISO 9000.
4. El aplicar las Normas ISO en los sistemas de gestión.
5. Poseer ayuda en temas referentes del sistema de gestión de la calidad.
6. Establezca las situaciones actual: determine las diferencias existentes entre su sistema de gestión de la calidad y un sistema que cumpla (Auto evaluación o evaluación por una organización externa).
7. Determinar los procesos para que los productos lleguen a los clientes.
8. Desarrolle un plan para eliminar las diferencias existentes entre el sistema de la calidad actual y un sistema que cumpla, identificando las acciones necesarias para eliminar las diferencias existentes, asignando recursos y responsabilidades para llevar a un programa para completar las acciones necesarias.
9. Lleve a cabo un plan, que implemente las acciones identificadas y haga un seguimiento del programa.
10. Planear el llevar a cabo auditorías internas de forma periódicas.
11. Llevar a cabo auditorías por organismos certificados.
12. Poseer una visión que lograr una mejora continua.

Beneficios que se obtienen de la certificación **ISO 9000-3** para software:

- Mejor documentación de los sistemas.
- Se realiza un cambio de la cultura de la organización de forma positiva.
- Un incremento de eficiencia en la forma de hacer las cosas.
- Mejor percepción de lo que es la calidad.
- Una ampliación en la satisfacción del cliente final.
- Una reducción de auditorías por parte de los clientes.

- Optimizar el tiempo total del desarrollo.

Como puede observarse, para obtener la certificación se deben de cumplir varios requisitos previos, por lo cual es importante poseer la información adecuada antes de tratar de lograr una certificación y que el personal involucrado posea la información adecuada y actual para estar inmersa dentro del proceso de certificación.

### **3.5 ISO para desarrolladores de software:**

Tomando en cuenta las secciones anteriores, una norma marca pautas para la fabricación de productos, realización de un proceso o desarrollo de un servicio, etc. Las normas ISO no específica para qué tipo de producto está orientada y pueden ser aplicadas a cualquier producto, para lo cual existen diferentes normas ISO para el aseguramiento de calidad.

Existe una parte de la norma **ISO 9000**, la cual se conoce como **ISO 9000-3** y es la norma para quien desarrolla y mantiene software, oficialmente el nombre de esta regla es ***“Norma de administración de la calidad y aseguramiento de calidad – Parte 3: Lineamientos para la aplicación de ISO 9001 para el desarrollo, suministro y mantenimiento de software”***.<sup>14</sup>

Todo lo anterior con el fin que las empresas que desarrollan software alcancen:

- Un medio para cubrir las expectativas del cliente.
- Los beneficios de la calidad y obtener ventajas competitivas al tener un nivel para competir en el mercado.
- Establecer estrategias para reducir los costos en la producción.

---

<sup>14</sup> Cita textual del libro titulado ISO 9000 QS-9000 ISO 14000, Carlos González, pagina 65.

Las guías en esta parte de la ISO 9000, están desarrolladas con el fin de describir los controles sugeridos y los métodos necesarios para el desarrollo del software, el cual debe cumplir con los requisitos del cliente. Todo, con el fin de obtener la satisfacción del cliente en todas las etapas, desde las prácticas iniciales hasta el mantenimiento del producto.

### **3.6 Norma ISO-9000-3: Guías para la aplicación de ISO-9001 para el desarrollo, entrega y mantenimiento de software**

Esta parte de la ISO-9000 da las guías para facilitar la aplicación de la ISO-9001 para la organización, desarrollo, entrega y mantenimiento de software. Esta se creó con el fin de proveer una guía donde exista un contrato entre dos partes con el fin de demostrar la capacidad de poder desarrollar, entregar y mantener productos de software. De igual manera esta parte de la ISO-9000 está desarrollada con el fin de describir controles y los métodos necesarios para la producción de software el cumplimiento de los requisitos del cliente en todas las etapas desde las prácticas iniciales hasta la entrega final del producto.

Esta parte de la ISO-9000 **son aplicables en situaciones contractuales** para productos de software, cuando dentro del contrato se especifican los requerimientos de diseño y lo que se espera del producto terminado, especialmente en términos de rendimiento cuando se deba dar una adecuada demostración de ciertas habilidades por parte del proveedor del producto de software.

### **3.7 Secciones de la norma ISO 9000-3:**

El objetivo del ISO 9000-3 es proveer las especificaciones de cómo aplicar el ISO 9001 al desarrollo, abastecimiento y mantenimiento del software. Se compone de un conjunto de capítulos que facilitan la aplicación de ISO



9001, en las empresas que desarrollan, abastecen y realizan el mantenimiento de software. Está dividido en tres grupos: **los requerimientos de gestión en general, los requerimientos de las actividades del ciclo de vida de los proyectos y los requerimientos de las actividades de soporte.**

Las primeras tres secciones de la norma ISO-9000-3 (**SECCIÓN 0 Introducción, SECCIÓN 1 Alcance, SECCIÓN 2 Referencias normativas**) sólo contienen orientaciones generales, y los restantes contienen la información a detalle de la forma en que se debe de llevar a cabo la utilización de dicha norma.

A continuación se listan las restantes secciones de la norma **ISO-9000-3** que están numerados de la sección 3 a la 6: <sup>15</sup>

**SECCIÓN 3 Definiciones:** para los propósitos de la norma ISO-9000, las definiciones dadas en ISO-2382-1 e ISO-8402 aplicadas conjuntamente dan las siguientes definiciones:

- **Software:** creación intelectual que comprende programas, procedimientos, reglas y cualquier documentación asociada perteneciente a la operación del sistema de procesamiento.
- **Producto de software:** conjunto completo de programas de computadora, procedimientos, documentación asociada y diseño de datos que ha de ser entregados al usuario.
- **Elemento de software:** cualquier parte identificable del producto de software en cualquier etapa intermedia del proceso fina de desarrollo.
- **Desarrollo:** todas las actividades que han de ser realizadas para completar un producto de software.

---

<sup>15</sup> González, Carlos, Normas internacionales de administración de calidad, sistemas de calidad y sistemas ambientales. ISO 9000 QS-9000 ISO 1400, McGraw-Hill, P 65-74

- **Fase o Etapa:** un segmento definido de trabajo, sin implicar el uso de algún modelo o implicación de un período de tiempo en el desarrollo del producto de software.
- **Verificación del software:** es el proceso de evaluar el producto en una fase dada para asegurarse la consistencia y el correcto desarrollo del producto respecto a los estándares dados como entrada.
- **Validación del software:** es el proceso de evaluar el software para asegurar el cumplimiento de los requerimientos especificados.

## **SECCIÓN 4 Sistemas de Calidad – Marco de trabajo**

En el sistema de calidad la empresa debe establecer y mantener un sistema de calidad documentado, como medio para asegurarla y garantizar que se comprendan, implementen y mantengan en todos los niveles de la organización y debe de incluir:

- a) La preparación de procedimientos e instructivos coherentes con los requisitos de la norma internacional y con la política de calidad.
- b) La aplicación efectiva de los procedimientos y de las instrucciones documentadas del sistema de calidad.

### **SECCIÓN 4.1 Responsabilidad de la gestión**

#### **SECCIÓN 4.1.1 Responsabilidades administrativas del proveedor**

La gerencia del proveedor debe definir y documentar sus políticas para, y como compromiso de calidad. El proveedor debe asegurar que estas políticas se comprendan, implementen

y mantengan en todos los niveles de la organización. La responsabilidad, autoridad y la interrelación del personal que administra, desarrolla y verifica la calidad del trabajo deben ser definidas, particularmente para el personal que necesita la libertad organizacional y la autoridad para:

- a) Iniciar acciones.
- b) La identificación y documentar cualquier problema de que se presente en la calidad.
- c) Iniciar, recomendar y dar soluciones a través de los canales de comunicación definidos.
- d) La verificación de la implementación de las posibles soluciones.
- e) Controlar el proceso, la entrega o la instalación del producto que no fue de entera satisfacción para el cliente, hasta que las definiciones de inconformidad se corrijan.

#### **SECCIÓN 4.1.2 Responsabilidades administrativas de los clientes**

El cliente debe cooperar con el proveedor para dar toda la información necesaria en el menor tiempo y resolver los puntos pendientes. El cliente debe asignar un representante con la responsabilidad para tratar con el proveedor los asuntos contractuales. Este representante debe tener la autoridad para poder negociar los aspectos contractuales; éstos deben incluir, pero no limitados a ellos, los siguientes puntos:

- a) Definir al proveedor todos los requerimientos del sistema.
- b) Dar respuestas a las preguntas que el proveedor haga.
- c) El hacer la aprobación de las propuestas que el proveedor brinda.

- d) Dar como finalizado el contrato con el proveedor.
- e) Asegurar, que la estructura organizacional del proveedor lleve a cabo los acuerdos del contrato.
- f) Definir qué criterios y procedimientos son validos para la aceptación.
- g) Proceder con el software entregado que no está apto para su uso.

### **SECCIÓN 4.1.3 Reuniones de revisión**

Se deben realizar reuniones para revisiones regularmente entre los clientes y proveedores; éstas deben estar programadas para cubrir los siguientes aspectos, como mínimo:

- a) Comprobar que el software desarrollado por el proveedor cumple con los requerimientos convenidos.
- b) Realizar la verificación de los resultados.
- c) Aceptación de los resultados de las pruebas.

Los resultados de cada una de las actividades de las revisiones deben ser aprobados y documentados.

## **SECCIÓN 4.2 Sistemas de calidad**

### **SECCIÓN 4.2.1 Generales**

El proveedor debe establecer y mantener documentados los sistemas de calidad. Los sistemas de calidad deben estar integrados dentro de todo el ciclo de vida del software, con el fin de asegurarse que la calidad se construye como un proceso progresivo y no al final del proceso. La prevención de problemas

debe de tener mayor énfasis que la corrección después de que aparecen.

El proveedor debe asegurar la efectiva implementación de la documentación de los sistemas de calidad.

### **SECCIÓN 4.2.2 Documentación de los sistemas de calidad**

Todos los documentos del sistema de calidad, requerimientos y provisiones, deben estar claramente documentados en una forma clara y sistemática.

### **SECCIÓN 4.2.3 Plan de Calidad**

El proveedor debe preparar y documentar los planes de calidad, con el fin de implementar las actividades para cada elemento de desarrollo del software con base en los sistemas de calidad, y asegurar que sea comprendido y observado por todos los elementos de las organizaciones.

### **SECCIÓN 4.3 Auditorias internas del sistema de calidad**

El proveedor debe llevar a cabo un sistema de planificación y documentación de las auditorias internas de los sistemas de calidad, para verificar que todas las actividades se están cumpliendo de acuerdo con los requerimientos planeados y determinar la efectividad de los sistemas de calidad.

Las auditorias deben ser calendarizadas con base al estado y la importancia de cada una de las actividades a realizar. Las

auditorias y los seguimientos deben desarrollarse de acuerdo con los documentos realizados en el inicio del sistema de calidad.

Los resultados de las auditorias deben ser documentados y entregados al personal que tiene la autoridad en el área en que se realizó la auditoria. El personal administrativo responsable del área debe tomar su tiempo para poder efectuar las acciones correctivas y buscar las deficiencias reportadas por la auditoría.

#### **SECCIÓN 4.4 Acción correctiva**

El proveedor debe establecer, documentar y mantener procedimientos para:

- a) Investigar la causa de la inconformidad del producto o servicio para tomar la acción correctiva necesaria para prevenir su recurrencia.
- b) Analizar todos los procesos, operaciones, concesiones, registros de calidad, reportes y entrevistas a clientes para detectar y eliminar las causas potenciales de la inconformidad del producto o servicio.
- c) Iniciar acciones preventivas para poder evitar las repercusiones del problema encontrado.
- d) Aplicar controles para asegurar que las acciones correctivas están realizándose y que son efectivas.
- e) Implementar y guardar todos los cambios en los procesos resultantes de las acciones correctivas.

## **SECCIÓN 5 Sistemas de calidad – ciclo de vida de las actividades**

### **SECCIÓN 5.1 Generalidades**

El desarrollo de proyectos de software debe estar organizado e implementado a través de un modelo del ciclo de vida. Las actividades de calidad deben planificarse e implementarse con base en el modelo del ciclo de vida a que se utilizará en el desarrollo del software.

Esta parte de la ISO-9000 se desarrolló con el propósito que su aplicación sea independiente del modelo de ciclo de vida que se esta utilizando.

### **SECCIÓN 5.2 Revisiones del contrato**

El proveedor debe establecer y mantener procedimientos para las revisiones contractuales y para coordinación de las actividades a desarrollar. Cada contrato debe ser revisado por el proveedor para asegurar que:

- a) El alcance del contrato y los requerimientos están definidos y documentados.
- b) Los riesgos y planes de contingencia han sido identificados.
- c) La información propietaria está adecuadamente protegida.
- d) Las diferencia en los requerimientos sea resuelta.
- e) Se tiene la capacidad de poder entregar todo acorde a los requerimientos especificados en el contrato.

- f) La responsabilidad del proveedor en subcontratos está bien definida.
- g) La terminología está bien comprendida por todas las partes.
- h) El cliente tiene la capacidad de cumplir con todas las obligaciones contractuales.

De igual manera se debe tener un registro de todas las revisiones contractuales que se desarrollen.

### **SECCIÓN 5.3 Especificaciones de los requisitos del comprador**

Para poder iniciar el desarrollo del software, el proveedor debe tener un completo conjunto no ambiguo de los requerimientos funcionales, es decir, estos requerimientos deben incluir todos los aspectos necesarios para satisfacer las necesidades de los clientes. Estos requerimientos deben incluir, como mínimo, lo siguiente: rendimiento, confiabilidad, seguridad y privacidad. Dichos requerimientos deben ser lo suficientemente precisos y permitan la validación de la aceptación del producto.

La especificación de requerimientos debe ser entregado comúnmente en un documento; si no es este el caso, el proveedor debe especificar estos requerimientos con una estrecha colaboración del cliente, y el proveedor debe tener su aprobación antes de entrar a la siguiente etapa. La especificación de los requerimientos del cliente debe de estar sujeto a los controles de documentación y configuración administrativa como parte del desarrollo del documento.



De igual manera, todos las interfaces entre el producto de software y otro tipo de software y/o productos de hardware deben estar claramente especificados, ya sea directamente o por referencia, en los requerimientos del cliente.

### **SECCIÓN 5.4 Planificación del desarrollo**

El desarrollo del plan debe cumplir con lo siguiente:

- a) Definición del proyecto, que incluya los objetivos y como referencia proyectos similares del proveedor o del cliente.
- b) La organización de los recursos del proyecto, incluyendo la estructura del equipo de trabajo, responsabilidades, utilización de subcontratos y materia que ha de ser usado.
- c) El desarrollo de cada una de las etapas en el ciclo de vida.
- d) Calendarización del proyecto, identificando las tareas que se van a desarrollar, los recursos y el tiempo requerido para cada una de las interrelaciones.
- e) Identificación de planes afines, tales como:
  - Planes de calidad
  - Planes de configuración administrativas
  - Planes de integración
  - Planes de pruebas.

### **SECCIÓN 5.5 Planificación de calidad**

Como parte de la planificación del desarrollo, el proveedor debe preparar un plan de calidad. Este debe ser actualizado conforme se va desarrollando el proyecto y con la completa definición de los elementos de la fase que se inicia.

El plan de calidad debe estar formalmente revisado y aprobado por la organización y grupo de trabajo encargado de su implementación. El documento que describe el plan de calidad puede ser un documento independiente o formar parte de otro o estar compuesto por varios documentos.

El plan de calidad debe especificar o tener referencia de los siguientes elementos:

- a) Objetivos de calidad, expresables en términos medibles, si es posible.
- b) Definir los criterios de las entradas y las salidas para cada una de las fases.
- c) Identificar los tipos de pruebas, verificación y validación que se llevarán a cabo.
- d) Un plan detallado de las pruebas, verificaciones y validaciones que van a ejecutarse, incluyendo la Calendarización, recursos y autorizaciones.
- e) Responsabilidades específicas para actividades de calidad, tales como:
  - Revisiones y pruebas
  - Configuración administrativa y cambios de control
  - Control y corrección de fallas.

## **SECCIÓN 5.6 Diseño e implementación**

Las actividades de diseño e implementación son aquellas que transforman los requerimientos del cliente en un producto de software. Dada la complejidad de un producto de software, es muy importante que estas actividades se lleven a cabo de una manera

disciplinada, con el fin de brindar un producto acorde a las especificaciones y no basar la calidad en las validaciones y verificaciones.

### **SECCIÓN 5.6.1 Diseño**

Además de los requerimientos comunes de todas las actividades de la fase de desarrollo, los siguientes aspectos inherentes a las actividades de diseño se deben de tomar en cuenta:

- a) **Identificación de las consideraciones de diseño:** además de las especificaciones de entrada y salida; aspectos como reglas de diseño y definición de interfaces se deben de tomar en cuenta.
  
- b) **Diseño de una metodología:** un diseño sistemático de una metodología apropiada para el tipo de producto de software que será desarrollado debe diseñarse.
  
- c) **Uso de experiencias vividas:** la utilización de las lecciones aprendidas de diseños anteriores ayudará al proveedor a evitar ciertos problemas y darle un mejor uso a los recursos.
  
- d) **Procesos subsecuentes:** los productos deben diseñarse con el fin de facilitar las pruebas, mantenimiento y uso.

## SECCIÓN 5.6.2 Implementación

Además de los requerimientos comunes de las actividades de desarrollo, los siguientes aspectos deben considerarse en cada actividad de la implementación:

- a) **Reglas:** tales como reglas de programación, convenciones referentes a nombres, codificación y reglas sobre los comentarios y documentación en programas.
  
- b) **Metodologías de implementación:** el proveedor debe usar métodos y herramientas apropiadas para satisfacer los requerimientos del cliente.

Además, debe llevar a cabo revisiones para asegurar que los requerimientos se están alcanzando y los métodos se están utilizando correctamente. El diseño o la implementación no se deben dejar de realizar hasta que todas las deficiencias y posibles problemas estén completamente corregidos. Un registro de todas las revisiones debe de estar bien documentado.

## SECCIÓN 5.7 Ensayos y validación

Las pruebas son necesarias en todos los niveles, es decir, desde los elementos más simples del sistema hasta el sistema completo, Y para esto hay muchas técnicas que ayudan a la validación y a la integración. En algunos casos, validaciones, pruebas de campo y pruebas de aceptación pueden ser una misma actividad.

La documentación que describe el plan de pruebas y validaciones pueden ser un documento independiente o ser parte de otro documento o estar compuesto de varios documentos.

### **SECCIÓN 5.7.1 Planificación de las pruebas**

El proveedor debe establecer y revisar los planes de pruebas, especificaciones y procedimientos antes de empezar las actividades descritas en este plan. Se debe dar mayor importancia a:

- a) Planes para probar programas, integración de los sistemas, pruebas del sistema y pruebas de aceptación.
- b) Pruebas de datos, casos y resultados esperados.
- c) Tipos de pruebas a ser desarrolladas, por ejemplo, pruebas de funcionalidad, pruebas de alcances, pruebas de rendimiento etc.
- d) Pruebas de ambiente, herramientas y software.
- e) Criterios con los cuales los resultados de las pruebas serán analizados.
- f) Qué documentación tendrá el usuario.
- g) Personal requerido y los requerimientos de entrenamiento asociado al plan.

Se debe de prestar especial atención a los siguientes aspectos de las pruebas:

- a) Los resultados de las pruebas se deben documentar.
- b) Cualquier problema descubierto y su posible impacto en otros módulos o sistemas se debe anotar y notificar a

las personas responsables para que el problema sea monitoreado hasta que sea solucionado.

- c) Áreas de impacto por cualquier modificación deber ser encontradas y probadas nuevamente.
- d) La relevancia y adecuación de las pruebas se deben evaluar.
- e) La configuración del hardware y software, se debe considerar y documentar.

### **SECCIÓN 5.7.2 Validación**

Antes de ofrecer el producto para la entrega y aceptación del cliente, el proveedor debe validar su completa operación como un todo, si es posible, en condiciones similares a las descritas en el contrato o a las que se encontrará el producto.

### **SECCIÓN 5.8Aceptación**

Cuando el proveedor está listo para entregar el producto ya validado, el cliente debe juzgar si el producto es o no aceptable con los criterios de aceptación y especificación en el contrato. El método para el manejo de los problemas detectados, se debe de documentar.

Para la aceptación de cualquier tipo de producto se realizarán un conjunto de actividades, y dentro de esto el proveedor debe de colaborar con el cliente y así identificar y desarrollar lo siguiente:

- a) Calendarización.

- b)Cuál será el procedimiento que se seguirá en las evaluaciones.
- c) Con qué recursos (software y hardware) se cuenta.
- d) Qué criterios tomaran en cuenta para la aceptación.

### **SECCIÓN 5.9 Reproducción, entrega e instalación**

La reproducción es un paso que se debe llevar a cabo antes de la entrega del producto; esta actividad debe tomar las siguientes acciones o consideraciones:

- a) Número de copias de cada elemento de software que va a ser entregado.
- b) Tipo de medio en el cual se entregará, incluyendo el formato y la versión, en una forma clara y comprensible para el cliente.
- c) Estipulación sobre la documentación que se va a entregar, tales como manuales y guías de usuarios.
- d) Qué licencias se otorgarán a la empresa.
- e) Copias de respaldo, incluyendo un plan de recuperación para desastres.
- f) El período u obligaciones del proveedor, para la entrega.

#### **SECCIÓN 5.9.1 Entrega**

Se deben tomar todas las precauciones necesarias para verificar la correcta y completa entrega de las copias del producto de software.

## **SECCIÓN 5.9.2 Instalación**

Los roles, responsabilidades y obligaciones del proveedor y del cliente se debe establecer claramente, tomando mayor atención en lo siguiente:

- a) Calendarios, incluyendo horas extras y fines de semana o días festivos que se trabajarán.
- b) Colaboración del cliente (passwords, datos, etc.)
- c) Disponibilidad de las personas que realizarán la instalación.
- d) Disponibilidad de poder acceder el sistema y los equipos del cliente.
- e) La necesidad de validar como parte de cada instalación debe quedar registrada en los contratos.
- f) Un procedimiento formal para la aprobación final de cada instalación.

## **SECCIÓN 5.10 Mantenimiento**

Cuando el mantenimiento de los productos de software es requerido, después de la entrega e instalación inicial, se debe, estipular en los contratos. El proveedor debe establecer y mantener procedimientos para desarrollar las actividades del mantenimiento y verificar que dichas actividades cumplen los requerimientos especificados para el mantenimiento.

Las actividades del mantenimiento para los productos de software, se clasifican en tres grupos, que son:

- a) Resolución de problemas.



- b) La modificación en las interfaces brindadas.
- c) Expansión funcional o mejoramiento del rendimiento.

Los elementos que entran dentro del mantenimiento, y el período de tiempo en el que durará, se deben especificar en los contratos.

### **SECCIÓN 5.10.1 Plan de mantenimiento**

Todas las actividades de mantenimiento se deben de realizar y ser manejadas de acuerdo con el plan de mantenimiento. Dicho plan incluye lo siguiente:

- a) Alcance que tendrá el mantenimiento.
- b) Observar en qué estado está el producto al inicio.
- c) El soporte que se dará a la organización.
- d) Qué actividades se han planeado hacer.
- e) Los registros y los reportes de cada uno de los mantenimientos.

### **SECCIÓN 5.10.2 Tipos de actividades del mantenimiento**

Todos los cambios de software efectuados durante el mantenimiento, se deben hacer de acuerdo con los mismos procedimientos usados en el desarrollo del producto de software. Y los cambios deben también ser documentados de la misma forma como se realizaron los documentos para el control y la configuración administrativa.

A continuación se describen algunas de las actividades de mantenimiento:

- a) **Resolución de problemas:** la resolución de problemas implica la detección, análisis y corrección de los elementos de software que no son de entera satisfacción del cliente.
- b) **Modificaciones de interfaces:** las interfaces de usuario requiere modificaciones, y para lo cual se debe tener en cuenta que muchas veces esto implicara agregar más elementos de hardware o componentes que son controlados por el software.
- c) **Expansiones funcionales o mejoramiento del rendimiento:** esto regularmente es solicitado por el cliente en la etapa de mantenimiento.

### **SECCIÓN 5.10.3 Registro y reportes de mantenimiento**

Todas las actividades del mantenimiento han de quedar registradas en formatos predeterminados. Las reglas para la entrega de los reportes de dichas actividades se deben establecer y aprobar por el cliente y proveedor. Dicho registro debe incluir los siguientes puntos:

- a) Lista de las peticiones de asistencia de reportes de problemas que han sido recibidos y el estado actual de los mismos.
- b) Persona responsable a la que se acudió para poder realizar las acciones correctivas.
- c) Prioridades que se han agregado a cada una de las acciones correctivas.

- d) Los resultado de las acciones al momento de hacer una corrección.
- e) Estadísticas y ocurrencias de las fallas y actividades del mantenimiento.

El registro de las actividades del mantenimiento se debe utilizar para la evaluación y mejoramiento del producto de software, así como el mejoramiento de los sistemas de calidad.

### **SECCIÓN 6 Sistema de Calidad – Actividades de soporte**

La configuración administrativa provee mecanismos para identificar, controlar y llevar registros de las versiones de los elementos de software. Y en muchos casos, de algunos que aún están en uso y deben de ser mantenidas y controladas. La configuración administrativa debe realizar lo siguiente:

- a) Identificación de las versiones de cada elemento del software.
- b) Identificar los grupos y los elementos de software, que constituyen la versión completa de un producto de software.
- c) Identificar el estado del producto de software en desarrollo, su entrega e instalación.
- d) Controlar las actualizaciones simultáneas de un elemento de software, que se realizan por más de una persona.
- e) Coordinar la actualización de múltiples productos en uno o más lugares, si es requerido.

- f) Llevar registro de todas las acciones y cambios resultantes de los cambios requeridos, desde su inicio hasta la liberación del producto.

## **SECCIÓN 6.1 Planificación de la configuración administrativa**

El proveedor debe desarrollar e implementar un plan para la configuración administrativa, el cual debe incluir lo siguiente:

- a) La organización debe estar comprometida en la configuración administrativa y las responsabilidades asignadas a cada una de las personas que están dentro de la organización.
- b) Planear las actividades que se llevarán a cabo.
- c) Herramientas, técnicas y metodologías que serán utilizadas.
- d) El momento en el cual se debe entrar al estado de control.

### **SECCIÓN 6.1.1 Actividades de la gestión de la configuración administrativa**

#### **SECCIÓN 6.1.1.1 Identificación de la configuración y su seguimiento**

El proveedor a de establecer y mantener los procedimientos para identificar los elementos de software durante todas las fases, empezando con la fase de especificaciones hasta la entrega del producto. Estos procedimientos también se aplican después de la entrega del producto. Cada elemento de software debe tener una única identificación.

Los procedimientos se aplicarán para asegurar que los siguientes puntos pueden ser identificados para cada una de las versiones de los elementos de software:

- a) Especificaciones funcionales y técnicas.
- b) Identificar las herramientas de desarrollo que afectan las especificaciones técnicas y funcionales.
- c) Las interfaces, con otros elementos de software y hardware.
- d) Los documentos y archivos de computadora relacionados con el elemento de software.

La identificación de un elemento de software debe llevarse a cabo de tal manera que las relaciones entre cada uno de ellos puedan ser demostrados. Para productos ya entregados, debe haber procedimientos que faciliten su seguimiento.

#### **SECCIÓN 6.1.1.2 Control de cambios**

El proveedor debe establecer y mantener procedimientos para identificar, documentar, revisar y autorizar cualquier cambio en los elementos de software bajo la configuración administrativa. Todos los cambios a los elementos de software se deben llevar a cabo de acuerdo con estos procedimientos.

Antes de que un cambio sea aceptado en su totalidad, se deben identificar y examinar todos los efectos que estos cambios puedan provocar. Se deben proveer los métodos para realizar los cambios, a las personas

indicadas para poder llevar a cabo el seguimiento necesario.

### **SECCIÓN 6.1.1.3 Reporte de los estados de las configuraciones**

El proveedor debe establecer y mantener procedimientos para registrar, administrar y reportar los estados de los elementos de software, los cambios solicitados y la aprobación de implementación de los cambios realizados.

## **SECCIÓN 6.2 Control de documentos**

El proveedor debe establecer y mantener procedimientos para controlar todos los documentos relacionados en el contenido de esta parte de la norma ISO-9000. Este debe cubrir:

- a) La determinación de aquellos documentos que deben estar sujetos al control.
- b) La aprobación y distribución de los procedimientos.
- c) Cuáles serán los procedimientos al realizar un cambio.

### **SECCIÓN 6.2.1 Tipos de documentos**

Los procedimientos de control deben de aplicarse a los documentos relevantes como pueden ser:

- a) Documentación de la descripción de los procedimientos de los sistemas de calidad que se aplicarán dentro del ciclo de vida del software.
- b) Documentos de planificación que describen la planificación y el progreso de las actividades del proveedor y sus interacciones con el cliente.
- c) Documentación del producto, el cual incluye:
  - Entradas y salidas de las fases de desarrollo.
  - La validación y la verificación se debe de planear y documentar.
  - Documentar para el cliente.
  - Documentar la clase de mantenimiento.

Los documentos han de ser revisados y aprobados por las personas autorizadas previo a ser publicados.

Se deben desarrollar procedimientos para asegurar que:

- a) La publicación de los documentos apropiados son disponibles en las localidades apropiadas donde, la operación esencial del funcionamiento efectivo del sistema se debe desarrollar.

- b) Documentos obsoletos, son removidos de las localidades apropiadas de uso o publicación.

En los lugares, donde el documento estará en uso y se hará por medio de computadoras, se debe prestar especial atención sobre la aprobación, acceso y procedimientos de almacenamiento.

### **SECCIÓN 6.2.2 Cambio de documentos**

Los cambios a los documentos, deben ser revisados y aprobados por los mismos organismos funcionales que los desarrollaron, revisaron y aprobaron la revisión original, a menos que se designe a alguien para la realización de esta tarea. La organización designada debe tener acceso a toda la documentación anterior para que pueda basar su revisión y aprobación.

Los documentos deben ser impresos o publicados nuevamente después de que se realicen todos los cambios.

### **SECCIÓN 6.3 Registros de calidad**

El proveedor debe establecer y mantener procedimientos para la identificación, colección, ordenamiento, almacenamiento, mantenimiento y disposición de los registros de calidad. Cada uno, ha de mantenerse para demostrar que se han alcanzado los requerimientos de calidad y su efectiva. De igual manera, el proveedor de poseer registros de calidad de los subcontratos realizados.



Los registros de calidad han de ser entendibles e identificables con relación al producto. Estos deben de almacenarse y mantenerse de tal forma que sean fácilmente recuperables y se estar en un ambiente adecuado con el fin de minimizar su deterioro, daño o pérdida. Los registros de calidad deben estar disponibles para su evaluación por el cliente o su representante, durante un período de tiempo. De igual forma, hay que tener un control de las personas que han realizado consultas a dichos registros.

## **SECCIÓN 6.4 Medición**

### **SECCIÓN 6.4.1 Medición del producto**

Los métricos o medidas deber ser reportados y usados para administrar el proceso de desarrollo y entrega, que es específico para cada uno de los productos de software. Los métodos seleccionados para cuantificar el proyecto, se deben describir para que los resultados puedan ser utilizados.

El proveedor de productos de software debe coleccionar y actuar con base en medidas cuantitativas de calidad, de los productos de software. Estas medidas deben ser utilizadas con el propósito siguiente:

- a) Obtener información y reportar los valores de las mediciones periódicamente.
- b) Identificar el estado actual de rendimiento en cada uno de los métricos.

- c) Tomar acciones correctivas si los niveles de los métricos salen de los rangos establecidos.
- d) Ha de establecerse metas que ayuden al mejoramiento tomando como base los métricos.

### **SECCIÓN 6.4.2 Proceso de medición**

El proveedor debe tener medias cuantitativas de la calidad de los procesos de desarrollo y entrega. Estos métricos deben reflejar:

- a) Si el proceso de desarrollo se está llevando correctamente con base en los hitos y a los objetivos de calidad planteados en los programas.
- b) Si el proceso de desarrollo es efectivo en la reducción de la probabilidad de falla.

Lo mas importante, en este momento, es conocer los niveles que se van a utilizar para el proceso de control y mejoramiento y no el valor específico de los métricos. La opción de los métricos se debe ajustar al proceso que se está utilizando, si es posible, para que tengan un impacto directo en la calidad de producto de software que se está entregando.

## **SECCIÓN 6.5 Reglas, prácticas y acuerdos**

El proveedor debe suministrar reglas, prácticas y acuerdos para cumplir efectivamente con los sistemas de calidad dados en esta parte de la ISO-9000. El proveedor debe revisar continuamente estas reglas, prácticas y acuerdos.

## **SECCIÓN 6.6 Herramientas y técnicas**

El proveedor debe utilizar herramientas y técnicas apropiadas para poder seguir las guías de los sistemas de calidad dadas en esta parte de la ISO-9000. Estas herramientas y técnicas deben ser efectivas en el proceso de desarrollo, así como también para propósitos administrativos.

## **SECCIÓN 6.7 Entregas**

El proveedor debe asegurar que el producto o servicio entregado esté conforme a los requerimientos. Los documentos que se entregarán deben tener la información claramente descrita que detallen perfectamente el producto o servicio entregado. Al mismo tiempo, se debe revisar y aprobar estos documentos para verificar si están correctos antes de su entrega.

### **SECCIÓN 6.7.1 Evaluación de subcontratos**

El proveedor debe seleccionar a los subcontratistas en función de su habilidad para cumplir los requerimientos de los subcontratos, incluyendo los requerimientos de calidad. Se debe llevar registros las empresas subcontradas que han sido seleccionadas.

La selección de un subcontratista, se debe hacer con base en el tipo de producto o servicio que la empresa realizará, tomando en cuenta también, los registros de los servicios o productos realizados anteriormente por dichas empresas.

### **SECCIÓN 6.7.2 Validación del producto entregado**

El proveedor es responsable por la validación del trabajo de los subcontratistas. Esto puede requerir que sea necesario la verificación de los diseños y se otras revisiones, con base en los sistemas de calidad que se vayan a tomar en cuenta, si es necesario, se deberá incluir en los contratos de los subcontratistas. Cualquier requerimiento de las pruebas de aceptación del trabajo de los subcontratistas por parte del proveedor también deben ser incluidos.

En el contrato se especificará, que el proveedor o su representante debe tener el derecho de determinar en el inicio o en el final, que el producto entregado esté conforme a los requerimientos establecidos. La validación del producto entregado no quita la responsabilidad del proveedor de entregar un producto aceptable o las subsiguientes entregas o revisiones.

Cuando el proveedor o su representante realicen la validación con el premiso del subcontratista, éste no se debe utilizar para verificar la efectividad de los sistemas de calidad del subcontratista.

## **SECCIÓN 6.8 Producto de software incluido**

El proveedor puede incluir o usar productos de software de terceras personas o de productos anteriores desarrollados por el mismo. Al igual, se debe establecer y mantener procedimientos para validar, proteger y mantener dichos productos.

Los productos proveedor/cliente que se encuentren no aptos para su utilización deben ser registrados y reportados, la validación por parte del cliente no elimina la responsabilidad del proveedor de entregar un producto aceptable.

## **SECCIÓN 6.9 Capacitación**

El proveedor debe establecer y mantener procedimientos para identificar las necesidades de entrenamiento de todo el personal que desarrolla actividades que afectan la calidad. Los trabajadores que desarrollan tareas específicas deben ser calificados con base en una apropiada educación, entrenamiento y/o experiencia, si es requerido.

Los temas seleccionados deben ser determinados con base en las herramientas específicas, técnicas, metodologías y recursos de hardware, que han de ser utilizados en la administración y desarrollo del producto de software. El entrenamiento puede ser necesario para desarrollar habilidades y conocimientos de áreas específicas dentro del producto de software, que se va a desarrollar. De igual manera ha de llevarse registros de la experiencia ganada y de los entrenamientos dados.

Como puede observarse, la norma ISO 9000-3, tiene como fin, garantizar el cumplimiento de los requisitos del cliente, garantizando desde las entrevistas el cumplimiento de las fases desarrollo y que llevarán a cabo cumpliendo con los procesos establecidos al momento de la certificación.

## 4. MÉTRICAS DEL SOFTWARE

Uno de los problemas mas grandes con lo que se enfrenta un gestor de proyectos; es de predecir si un proyecto será exitoso. Muchos de los gestores orientados a metas únicamente toman el tiempo, costo y rendimiento como parámetros independientes para medir el éxito, pero éstos no identifican si el proyecto está correctamente administrado, o satisface las necesidades del cliente, y si está cumpliendo los estándares o normas locales o internacionales de calidad.

Se puede decir entonces, que éxito de un proyecto de software está medido por las acciones o comportamientos de tres grupos: el grupo de desarrollo, la empresa cliente, y la empresa desarrolladora.

De ahí, la importancia que todo desarrollador conozca las técnicas que se utilizan para el control, evaluación y medición de un proyecto, con el objeto que el producto sea de alta calidad y además cumpla con normas y estándares que garantizan la satisfacción de todas las personas que están involucradas con el servicio o producto que se va a realizar.

Existen varias razones para tener una medida de un producto.

1. Para lograr indicar calidad que un producto.
2. Obtener que tan productiva es el personal.
3. Para evaluar los beneficios, derivados del uso de métodos y herramientas de la ingeniería de software.
4. Lograr tener una línea que servirá de base para la estimación.
5. Para justificar el porqué usar nuevas herramientas

#### 4.1 Actividades del proceso de medición<sup>27</sup>

Antes de empezar con el proceso de medición, que ayude a tomar decisiones, es importante comprender los principios básicos de la medición, la cual posee las siguientes actividades:

- **Formulación:** La obtención de medidas y métricas del software apropiadas para la representación del software que se desarrollará.
- **Recolección:** El mecanismo empleado para acumular los datos que se necesitan para obtener las métricas formuladas.
- **Análisis:** es calcular las métricas y aplicar herramientas matemáticas.
- **Interpretación:** poseer la capacidad para evaluar los resultados en un esfuerzo por conseguir una buena visión de la calidad.
- **Retroalimentación:** Las recomendaciones que se derivan de la interpretación de métricas y su transmisión a los miembros del equipo.

Las métricas del software solo serán útiles si están caracterizadas de manera efectiva, y se validan para probar su valor. Los siguientes principios son representativos para validar las métricas:

- Una métrica debe tener propiedades matemáticas deseables: es decir, el valor debe estar en un rango significativo y en una escala racional.
- Cuando una métrica representa una característica de software, tiende a aumentar o disminuir cuando se encuentran rasgos positivos o negativos en dicha característica.

---

<sup>27</sup> <http://rios-vazquez.blogspot.com/2009/02/unidad-2-el-proceso-del-software-y.html>  
<http://www.monografias.com/trabajos55/proceso-de-desarrollo-software/proceso-de-desarrollo-software2.shtml>



- Cada métrica debe validarse empíricamente en una amplia variedad de contextos antes de publicarse o ampliarse en la toma de decisiones y además, una métrica debe medir el factor de interés.

Aunque la formulación, caracterización y validación son críticas, la recopilación y el análisis son las actividades que dirigen el proceso de medición.

#### 4.2 ¿Qué es un métrico?

Antes de entrar en detalle con el significado de métrico, se debe tomar en cuenta que **medida**, **medición** y **métrico** son términos que suelen utilizarse de manera intercambiable y es importante observar que existen pequeñas diferencias entre ellos.

En términos de la Ingeniería del Software una **medida** proporciona una indicación cuantitativa, cantidad, dimensión, capacidad o tamaño de algún atributo de un producto o proceso y **medición** es el acto de determinar una medida y de igual manera un **métrico** se define como “**una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado**”<sup>28</sup>, o (métrico: del metro o de la medida<sup>29</sup>).

De lo anterior, un métrico es un número que se asocia con una idea, es decir, es una indicación medible de algún aspecto cuantitativo (por ejemplo, el número promedio de errores encontrados en cada revisión o prueba).

Con todo esto, un ingeniero de software recopila medidas y desarrolla métricos para obtener los indicadores, y un **indicador**<sup>30</sup> es un métrico o combinación de métricos que proporcionan conocimientos acerca del proceso

---

<sup>28</sup> Según glosario de estándares del IEEE [IEE93]

<sup>29</sup> Gran diccionario Salvat, Mallorca España, 1989.

<sup>30</sup> <http://www.definicion.org/indicador>

del software, un indicador proporciona conocimientos que permiten al jefe de proyecto o a los ingenieros de software ajustar el proceso, el proyecto o el producto para que las cosas mejoren.

### 4.3 Clasificación de las métricas<sup>31</sup>

La clasificación de una métrica de software refleja o describe la conducta del software. A continuación se muestra una breve clasificación de métricas de software:

- **Métricas de complejidad:** son todas las métricas de software que definen de una u otra forma la medición de la complejidad; tales como: volumen, tamaño, nidación, costo (estimación), agregación, configuración, y flujo.
- **Métricas de calidad:** son aquellas definen la calidad del software; como exactitud, estructuración, pruebas, mantenimiento, reutilización, cohesión, acoplamiento. Y estas se convierten en puntos críticos en el diseño, codificación, pruebas y mantenimiento.
- **Métricas de competencia:** son las que se usan para medir las actividades y productividad de los programadores. En esta área ha sido el avance, a pesar de haber una intensa investigación al respecto.
- **Métricas de desempeño:** corresponden a las métricas que miden la conducta e interacción de módulos y otros sistemas, bajo la supervisión del sistema operativo o hardware.

---

<sup>31</sup> <http://www.monografias.com/trabajos55/proceso-de-desarrollo-software/proceso-de-desarrollo-software2.shtml>

- **Métricas estilizadas:** son las métricas de experimentación y de preferencia; Por ejemplo: estilo de código, indentación, las convenciones denominando de datos, las limitaciones, etc. De ningún modo deben de confundirse con las métricas de calidad.
- **Variedad de métricas:** orientadas a la portabilidad, localización y consistencia.

Dichas clases de métricas ayudan a fortalecer la idea de que no solamente una métrica puede ser utilizada para ver la complejidad y la calidad del producto terminado conocido como software.

#### 4.4 Diferentes enfoques de las métricas

Se han propuesto cientos de métricas para el software, pero no todas proporcionan suficiente soporte práctico para su desarrollo. Algunas, demandan mediciones que son demasiado complejas, otras son difíciles de entenderlas, y otras violan las nociones básicas intuitivas de lo que realmente es el software de alta calidad. Es por eso, que se han definido una serie de atributos que deben acompañar a las métricas efectivas de software, por lo tanto la métrica obtenida y las medidas que conducen a ello deben cumplir con las siguientes características fundamentales:

- **Simple y calculable:** Debe ser relativamente fácil de aprender y su cálculo no debe exigir cantidades anormales de tiempo o esfuerzo.
- **Empírica e intuitivamente persuasivas:** La métrica debe de satisfacer las nociones intuitivas del ingeniero acerca del atributo del producto que se está evaluando.

- **Consistentes y objetivas:** La métrica siempre debe arrojar resultados que no permitan ambigüedad alguna.
- **Independientes en el uso de unidades y dimensiones:** El cálculo matemático de la métrica debe emplear medidas que no lleven a combinaciones extrañas de unidades.
- **Independientes del lenguaje de programación:** Las métricas deben basarse en el modelo de análisis o diseño, o en la estructura del propio programa.
- **Mecanismos efectivos para la retroalimentación:** La métrica debe llevar a un producto final de las mas alta calidad

Tomando en cuenta lo anterior, lo que se desea controlar debe ser medible, y que esta medición no sea influenciada por ningún factor externo o interno. No obstante que la mayoría de las métricas de software compensan las características anteriores, algunas de las métricas usualmente empleadas no cumplen una o dos características.

De igual manera, se debe llevar un registro completo de los métricos, este debe de llevar como mínimo: el nombre de la persona que la lleva a cabo, fecha en que se hizo y el nombre de la persona que está desarrollando la tarea o proceso medido.

#### **4.5 Diseño de la métrica**

Antes de empezar con el proceso de cálculo de métricas, se debe de planear los pasos para diseñar las métricas, a continuación se listan los pasos para definir las:

**Paso 1 :Definiciones claras:** El primer paso en el diseño de una métrica es dar una definición estándar para las entidades y sus atributos a ser medidos, para que no existan ambigüedades de interpretación, debido a que otra gente podría interpretar estas palabras en su propio contexto con significados que pueden variar de nuestra intención al definirlos.

**Paso 2: Definir el modelo:** El siguiente paso es definir un modelo para la métrica. En términos sencillos, se define en el **cómo** calcular la métrica. Cuando se crea un modelo de medición de software se necesita ser pragmáticos.

Si se trata de incluir todos los elementos que afectan al atributo o que caracterizan a la entidad, el modelo utilizado llegará a ser demasiado complicado.

Ser pragmático significa ser práctico y no tratar de crear un modelo perfecto y tan solo tomar lo más importante. Recordar que el modelo puede ser siempre modificado para incluir mas niveles de detalle en el futuro.

**Paso 3: Establecer un criterio de conteo:** El siguiente paso en el diseño de una métrica es dividir al modelo en sus métricas primitivas (cuantificables en forma directa) y la definición del criterio de conteo para medir cada primitiva. En este paso, se debe definir la forma del conteo para la medición de la métrica. Las métricas primitivas y su criterio de conteo definen el primer nivel de los datos que necesitan ser recolectados para implantar la métrica.

Para ejemplificar, esto se va usar el modelo de minutos de caída por servidor por año. Una de las métricas primitivas para este modelo es el número de servidores, dado de criterio de conteo:

- ❖ "servidores" al final
- ❖ "servidores" al inicio + "servidores" al final/ 2.
- ❖ También podría ser ("servidor" en cada día) / número de días

**Paso 4: Decidir ¿Qué es bueno?** una vez que se a decidido que medir y como medir, se tiene que decidir que hacer con el resultado. ¿Es 5 muy poco o 25 sobre pasa el límite? ¿se debe crecer o no?.

Uno de los primero lugares para empezar a ver "¿Qué es bueno?", es con los clientes. La mayoría de veces, los requerimientos del usuario son los que determinarán los valores de algunas métricas. En el momento que se ha tomado la decisión de "¿Qué es bueno?", se pueden echar andar las medidas necesarias. Si se está "excediendo" los valores deseados, o si las acciones correctivas no son necesarias.

**Paso 5: Reporte de la métrica:** El siguiente paso es decidir es como hacer el reporte la métrica. Esto incluye la definición del formato, la obtención de los datos, mecanismos de distribución y disponibilidad. El formato del reporte se diseña de acuerdo a lo que se quiera dar a conocer, es por eso que se debe preguntar lo siguiente:

- ¿Está la métrica incluida en una tabla con otras métricas para un período?
- ¿Sé añade como último valor en una gráfica de tendencia que muestran los valores de la métrica para varios períodos?.

En el reporte de métricas, se deberá realizar cuatro pasos que se describen a continuación:

- **El ciclo de obtención de datos:** define que a menudo la métrica requiere de vistas de los datos y en qué momento estos elementos están disponibles para el cálculo.
- **El ciclo de reporte:** define con qué frecuencia el reporte es generado y cuando se prepara para su distribución.
- **Los medios de reporte:** informan como se entregaran las métricas.
- **Definiendo como se distribuirá:** Se determina que persona recibirá una copia y las restricciones de acceso a la misma,

**Paso 6: Calificadores adicionales:** El paso final en el diseño de una métrica es el determinar calificadores adicionales. Una buena métrica es la que es genérica. Esto significa que la métrica es válida para todos los niveles de calificadores que se puedan adicionar.

Los calificadores adicionales necesitan ser definidos como parte del diseño de las métricas ya que estos determinan el segundo nivel de los requerimientos de la recolección de datos. Ninguna discusión de selección y diseño de métricas de software sería completa sin tener en cuenta el cómo las mediciones **afectan a la gente y cómo la gente afecta a las mediciones.**

Con tan solo medir, esto afecta el ambiente de las personas que serán medidos. La gente quiere ser bien vista, por lo que van a querer que las mediciones sean buenas.

La mejor manera de prevenir problemas con el factor humano al trabajar con métricas es seguir algunas de las siguientes reglas básicas:

- **No hacer mediciones del individuo:** las mediciones de productividad individual son los ejemplos clásicos de estos errores. Si se midiera la productividad en líneas de código por hora, la gente se concentraría en su propio trabajo y excluiría al equipo de trabajo, o se enfocaría en realizar programación con líneas extras de código. Es por eso que se recomienda enfocarse en el proceso y en el producto, no en la gente.
  
- **No ignorar los datos:** un camino seguro para acabar un programa de métricas es no olvidar los datos cuando se toman las decisiones, ya que estos "dan sustento a la gente cuando sus reportes emplean información útil a la organización". Si las metas que se establecen y se comunican no son respaldadas con acciones entonces la gente en la organización se desempeñará basándose en el ambiente y no en las metas. Lo que debería realizarse, es:
  - **Seleccionar las métricas basándose en objetivos:** para tener métricas que cumplan con nuestra necesidad de información, se debe seleccionar métricas que proporcionen información que den respuesta a las preguntas.
  
  - **Proveer retroalimentación:** para que los miembros del equipo se mantengan informados de los resultados.
  
  - **Tener compromiso:** con comprometer a la gente en alcanzar las metas, genera que las personas dentro del equipo logren un sentimiento de propiedad, por esto el participar en definir de las métricas acrecentara ese sentimiento.



## 4.6 Panorama de las métricas del producto

Aunque en la actualidad se han propuesto una amplia variedad de métricas, se lista a continuación algunos aspectos que se debe tomar en cuenta dentro de las áreas del modelo de desarrollo de software, independientemente del modelo que se elija.

- **Métricas para el modelo de análisis:** estas métricas atienden varios aspectos específicamente del modelo de análisis e incluyen:

**Funcionalidad entregada:** proporciona una medida indirecta de la funcionalidad que se empaqueta con el software.

**Tamaño del sistema:** mide el tamaño general del sistema, definido desde el punto de vista de la información disponible como parte del modelo de análisis.

**Calidad de la especificación:** proporciona una indicación de la especificidad o el grado en que se ha completado la especificación de los requisitos.

- **Métricas para el modelo de diseño:** Estas métricas cuantifican los atributos del diseño de manera tal que le permiten al ingeniero de software evaluar la calidad del diseño. La métrica incluye:

**Métricas arquitectónicas:** proporcionan un indicio de la calidad del diseño.

**Métricas al nivel de componente:** miden la complejidad de los componentes del software y otras características que impactan la calidad.

**Métricas de diseño de la interfaz:** se concentran principalmente en la facilidad de uso.

**Métricas especializadas en diseño orientado a objetos:** miden características de clases, además de las correspondientes a comunicación y colaboración.

- **Métricas para el código fuente:** Estas métricas miden el código fuente y se usan para evaluar su complejidad, además de la facilidad con que se mantiene y prueba, entre otras características:

**Métricas de complejidad:** miden la complejidad lógica del código fuente.

**Métricas de longitud:** proporciona un indicio del tamaño del software.

- **Métricas para pruebas:** Estas métricas ayudan a diseñar casos de prueba efectivos y a evaluar la eficacia de las pruebas:

**Métricas de cobertura de instrucciones y ramas:** lleva al diseño de casos de prueba que proporcionan cobertura del programa.

**Métricas relacionadas con los defectos:** se concentran en encontrar defectos y no en las propias pruebas.

**Efectividad de la prueba:** proporciona un indicio en tiempo real de la efectividad de las pruebas aplicadas.

**Métricas en proceso:** métricas relacionadas con el proceso que se determinan a medida que se aplican las pruebas.

#### **4.6.1 Métricas para el modelo de análisis**

Estas métricas examinan el modelo de análisis con la intención de predecir **el tamaño** del sistema resultante. El tamaño es, en ocasiones un indicador de la complejidad del diseño y casi siempre resulta un indicador de un mayor esfuerzo de codificación, integración y prueba.

##### **4.6.1.1 Métrica basada en la función**

Utilizan una medida de la funcionalidad; ésta no se puede medir directamente, se debe derivar indirectamente por medio de medidas directas.

###### **4.6.1.1.1 Puntos de Función (PF)<sup>32</sup>:**

Se usa de manera efectiva como medio para medir la funcionalidad que entrega un sistema, todo esto con base en datos históricos, la idea es que examinemos una especificación del sistema, estas se derivan con una relación empírica según las medidas contables (directas) del dominio de información del software y las evaluaciones de la complejidad de software.

Para que se debe utilizar los Puntos de Función:

1. Estimar el costo o esfuerzo requerido para diseñar, codificar y probar el software.
2. Estimar el número de errores que podrían aparecer durante las pruebas.

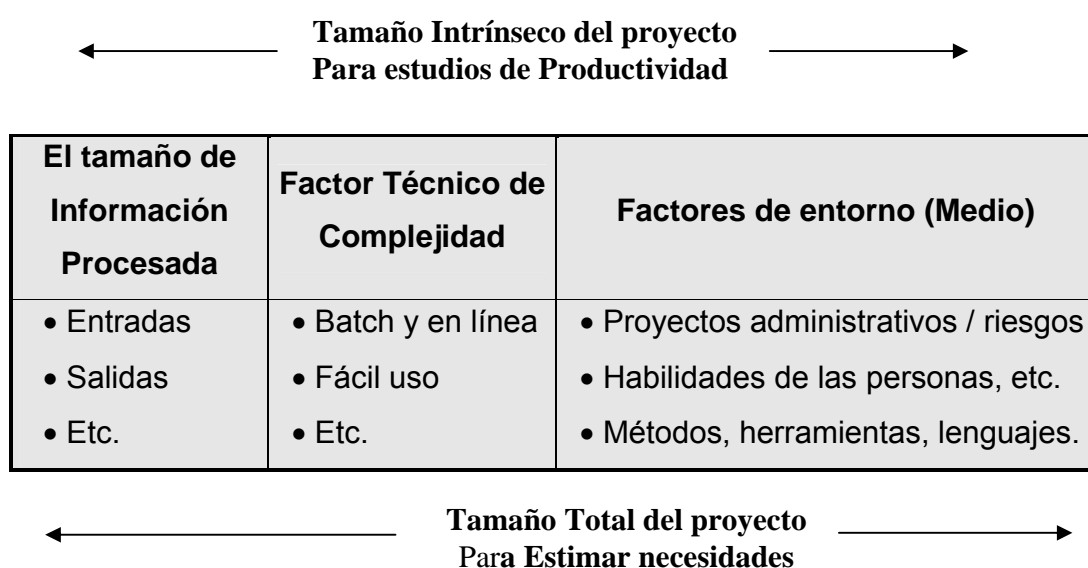
---

<sup>32</sup> <http://www.sc.ehu.es/jiwdcoj/mmis/fpa.htm>

3. Pronosticar el número de componentes, líneas de código proyectada, o ambas en el sistema.

El tamaño de la tarea de diseño y desarrollo de un sistema de cómputo es determinado por el producto de tres factores mostrados en la figura 15:

**Figura 15 Tamaño de diseño y desarrollo de un sistema de cómputo<sup>33</sup>**



- **El tamaño de información procesada:** esta es una medida de las entradas y salidas de la información y del proceso que lleva en el sistema.
- **Factor técnico de complejidad:** en éste toma en cuenta la medida de varias técnicas y otros factores implicados en el desarrollo y en el implemento de la información procesada requerida.
- **Factores de entorno (o medio):** éste es el grupo de factores que surge del entorno del proyecto típicamente valorado en proyectos con riesgo de

<sup>33</sup> Charles R. Symons, '98]

medidas. Incluye habilidades, experiencia y motivaciones del personal involucrado y de los métodos, lenguajes y herramientas usadas por el equipo del proyecto.

**El método de Punto de Función (PF)** consiste en componentes de un sistema que se clasifican en cinco tipos:

**Número de entradas (EE):** Cada entrada externa se origina en un usuario o es transmitida desde otra aplicación y proporciona distintos datos orientados a la aplicación o información de control. Dicha información, suele utilizarse para actualizar archivos lógicos internos, las entradas deben distinguirse de las consultas, que se encuentran por separado.

**Número de salidas externas (SE):** Cada salida externa se deriva en el interior de la aplicación y proporciona información al usuario. En este contexto, salida externa alude informes, pantallas, mensajes de error. Los datos dentro de los informes no se contarán como elementos individuales.

**Número de consultas externas (CE):** Una consulta externa se define como la entrada en línea que lleva a la generación de alguna respuesta inmediata por parte del software, en la forma de salida en línea.

**Número de archivos lógicos internos (ALI):** Cada archivo lógico interno es un agrupamiento lógico de datos que reside dentro de los límites de las aplicaciones y que se mantiene mediante entradas externas.

**Número de archivos de interfaz externos (AIE):** Cada archivo de interfaz externo es un agrupamiento lógico de datos externo a la aplicación pero que proporciona datos que podrían usarse en ésta.

Dependiendo del número de elementos de datos (**conteo**), estos se denominan como “simple”, “promedio” o “complejo”. Las organizaciones que usan métodos de puntos de función desarrollan criterios para determinar si una entrada determinada es simple, promedio o compleja.

Cada componente es el número dado de puntos dependiendo en tipo y complejidad (**tabla 1**), seguido se debe de multiplicar el número de elementos obtenidos de cada una de las entradas (**cantidad**) por el tipo de factor de ponderación que se haya elegido (simple, promedio o complejo) y la suma para todos los componentes es expresado en “Puntos funcionales sin ajustar”.

**Tabla I. Calculo de Puntos de Función**

		Factor de Ponderación			Total	
Descripción (Entradas)	Conteo	Simple	Promedio	Complejo		
Entradas Externas (EE)	3	*	3	4	6	9
Salidas Externas (SE)	2	*	4	5	7	10
Consultas Externas (CE )	2	*	3	4	6	8
Archivos de Lógica Interna (ALI)	1	*	7	10	15	15
Archivos de Interfaz Externa (AIE)	4	*	5	7	10	20
Total Puntos de Función sin ajustar ( <b>Conteo Total</b> )						<b>62</b>

Para calcular los puntos de función se usa la siguiente relación:

$$PF = \text{conteo total} * [0.65 + (0.01 * \sum(F_i))]$$

Donde **conteo total** es la suma de todas las entradas de PF obtenidas de la Tabla I, y para el calculo de  $\Sigma(F_i)$  es necesario utilizar los *factores de ajuste* Tabla II, para dar respuesta a las 14 preguntas que se muestran en la Tabla III, asignándoles un valor de 0 a 5 dependiendo del grado de influencia que posea en el sistema, y que brindará el grado total de influencia el cual se convierte en el **Factor Técnico de Complejidad** de la Figura 15.

**Tabla II. Factores de ajuste para puntos de función**

Valores de Respuesta	Factor de ajuste
No presente o no influencia	0
Influencia insignificante o incidental	1
Influencia moderada	2
Influencia promedio o medio	3
Influencia significativa	4
Influencia esencial o fuerte a través de	5

**Tabla III. Preguntas para el cálculo del factor de ajuste**

No.	Pregunta	Ponderación [0-5]
1	¿El sistema requiere respaldo y recuperación confiable?	
2	¿Se requieren comunicaciones de datos especializadas para transferir información a la aplicación, ní obtenerla de ella?	
3	¿Hay funciones distribuidas de procesamiento?	
4	¿El desempeño es crítico?	
5	¿El sistema se ejecutará en un entorno existente que tiene un uso pesado de operaciones?	
6	¿El sistema requiere entrada de datos en línea?	

7	¿La entrada de datos en línea requiere que la transacción de entrada se construya en varias pantallas u operaciones?	
8	¿Los ALI se actualizan en línea?	
9	¿Las entradas, las salidas, los archivos o las consultas son complejos?	
10	Es complejo el procesamiento interno?	
11	¿El código diseñado será reutilizable?	
12	¿Se incluyen la conversión e instalación en el diseño?	
13	¿Está diseñado el sistema para instalaciones múltiples en diferentes organizaciones?	
14	¿La aplicación está diseñada para facilitar el cambio y para que el usuario lo use fácilmente?	
Total de ponderación $\Sigma(F_i)$		[Valor]

Quando se calculan los puntos de función, éstos se utilizan de forma análoga a las LDC (Líneas de Código) para normalizar medidas de productividad, calidad y otros ámbitos de software, como por ejemplo errores, defectos, costo.

Por lo cual, estos datos ayudarán a los Ingenieros de Software a evaluar el grado en el que han completado sus actividades de revisión y prueba.

#### 4.6.1.1.2 Métricas de tamaño del sistema (La métrica Bang<sup>34</sup>)

Puede emplearse para desarrollar una indicación del tamaño del software a implementar como consecuencia del modelo de análisis. La forma de la implementar o el tamaño del sistema son independientes para La Métrica Bang.

<sup>34</sup> Desarrollada por Tom DeMarco 91, <http://www.mitecnologico.com/Main/LaMetricaBang>



Para calcular el desarrollador del software debe evaluar primero un conjunto de primitivas (elementos del modelo de análisis que no se subdividen más en el nivel de análisis). Las primitivas se determinan evaluando el modelo de análisis y desarrollando cuentas para los siguientes elementos:

- **Primitivas funcionales (Pfu<sup>35</sup>)** transformaciones (burbujas) que aparecen en el nivel, en el diccionario de datos.
- **Objetos**
- **Relaciones (RE<sup>36</sup>)** Las conexiones entre objetos de datos.
- **Transiciones (TR<sup>37</sup>)** El número de transacciones de estado en el diagrama de transición de estado.

Además de las seis primitivas nombradas arriba, se determinan medidas adicionales para:

- **Primitivas modificadas de función manual (PMFu):** Funciones que caen fuera del límite del sistema y que deben modificarse para acomodarse al nuevo sistema.
- **Elementos de datos de entrada (EDE<sup>38</sup>):** Aquellos elementos de datos que se introducen en el sistema.
- **Elementos de datos de salida (EDS:)** Aquellos elementos de datos que se sacan en el sistema.
- **Elementos de datos retenidos (EDR<sup>39</sup>):** Aquellos elementos de datos que son retenidos (almacenados) por el sistema.

---

<sup>35</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

<sup>36</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

<sup>37</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

<sup>38</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

<sup>39</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

- **Muestras (tokens) de datos (TCi):** Las muestras de datos (elementos de datos que no se subdividen dentro de una primitiva funcional) que existen en el límite de la *i*-ésima primitiva funcional (evaluada para cada primitiva).
- **Conexiones de relación (Rei):** Las relaciones que conectan el *i*-ésimo objeto en el modelo de datos con otros objetos.

Es desarrollador de esta métrica sugiere que la mayoría del software se puede asignar a uno de los dos dominios siguientes, **dominio de función** o **dominio de datos**, dependiendo de la relación RE/PFu.

Las aplicaciones de **dominio de función**<sup>40</sup> (encontrados comúnmente en aplicaciones de ingeniería y científicas) hacen hincapié en la transformación de datos y no poseen generalmente estructuras de datos complejas.

Las aplicaciones de **dominio de datos**<sup>41</sup> tienden a ser modelos complejos de datos.

- Si  $RE / Pfu < 0,7$  implica una aplicación de dominio de función.
- Si  $0,8 < RE / Pfu < 1,4$  implica una aplicación híbrida.
- Si  $RE / Pfu > 1,5$  implica una aplicación de dominio de datos.

Como diferentes modelos de análisis harán una participación del modelo con mayor o menor grado de refinamiento, se sugiere que se emplee una cuenta medida de muestras (*token*) por primitiva para controlar la uniformidad de la participación a través de muchos diferentes modelos dentro del dominio de una aplicación.

---

<sup>40</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

<sup>41</sup> <http://www.mitecnologico.com/Main/LaMetricaBang>

$$TC_{avg} = TC_i/P_{fu}$$

Una vez que se ha calculado la métrica Bang, se puede emplear el historial anterior para asociarla con el esfuerzo y el tamaño. De igual manera, se sugiere que las organizaciones construyan sus propias versiones de tablas IPFuC y IOBC para calibrar la información de proyectos completos de software.

#### 4.6.1.1.3 Métrica para la calidad de la especificación<sup>42</sup>

Este método se basa específicamente en que existe una lista de características con las cuales puede evaluarse la calidad del modelo de análisis y la correspondiente especificación de requisitos: *especificidad (falta de ambigüedad), grado de avance, corrección, facilidad de comprensión, facilidad de verificación, consistencia interna y externa, facilidad para alcanzar los objetivos, concisión, facilidad para darle seguimiento, facilidad para modificarse, precisión y facilidad de reutilización.*

Además, se observa que las especificaciones de alta calidad deben estar almacenadas electrónicamente, ser ejecutables o por lo menos interpretables, estar anotadas por importancia relativa, ser estables, tener indicada la versión, estar organizadas, incluir referencias cruzadas y especificarse con el grado de detalle correcto.

Aunque, al parecer, muchas de estas características tienen una naturaleza cualitativa, se sugiere que cada una puede representarse empleando una o más métrica.

Hay que suponer que hay  $n_r$  requisitos en una especificación, de modo que:

---

<sup>42</sup> Davis [Dav93], <http://www.mitecnologico.com/Main/MetricasModeloDeAnalisis>

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

donde  $n_f$  es el número de requisitos funcionales y  $n_{nf}$  el de no funcionales (como el desempeño).

Para determinar la especificidad (falta de ambigüedad) de los requisitos, se sugiere una métrica basada en la consistencia de la interpretación de los revisores de cada requisito:

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

donde  $n_{ui}$  es el número de requisitos que todos los revisores interpretaron de la misma manera. Cuanto más cercano este el valor de Q a 1, menor será la ambigüedad de la especificación.

El **grado de avance** de los requisitos funcionales se determina al calcular la relación

$$Q_2 = n_u / [n_j * n_s]$$

donde  $n_u$  es el número de requisitos de función única;  $n_j$  el número de entradas (estímulos) definidos o implícitos en la especificación, y  $n_s$  es número de estados especificados.

La relación Q2, mide el porcentaje de funciones necesarias que se han especificado para un sistema. Sin embargo, no se atienden requisitos que no son funcionales. Para incorporarlos a una métrica general del grado de avance, se de considerar el grado de validación de los requisitos:

$$Q_3 = n_c / [n_c + n_{nv}]$$

donde  $n_c$  es el número de requisitos que se han validado como correctos, y  $n_{nv}$  el de requisitos que aún no se validan.

#### **4.6.2 Métricas para el modelo de diseño**

Sería inconcebible que el diseño de un nuevo avión, un nuevo chip de computadora o un nuevo edificio de oficinas se realizara sin definir las medidas de diseño, sin determinar las métricas de diversos aspectos de calidad del diseño y si usarlas para guiar la manera en que evoluciona el diseño.

Es de notar que al diseñar sistemas complejos muchas veces se avanza sin medición. Las métricas para software, como otras métricas, no son perfectas; muchos expertos argumentan que se necesita más experimentación hasta que se puedan emplear bien las métricas de diseño. Es inaceptable el diseñar sin ningún factor de medición.

Continuando se mostraran algunas métricas orientadas al diseño. Aunque ninguna es perfecta, pueden proporcionarle al diseñador una mejor visión interna y así el diseño evolucionara a un mejor nivel de calidad.

##### **4.6.2.1 Métricas de diseño**

Éstas, se concentran en las características de la estructura del programa dándole énfasis a la estructura y en la eficiencia de los módulos. Estas métricas son de “**caja negra**”, en el sentido de que no se requiere ningún conocimiento del trabajo interno de un componente de software en particular.

Los autores Card y Glass<sup>43</sup> definen tres medidas de la complejidad del diseño del software:

- Estructural
- De Datos
- Del Sistema

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

$$S(i) = f_{out}^2(i)$$

donde  $f_{out}(i)$  es la expansión del módulo  $i$ .

La **complejidad de los datos  $D(i)$**  es una indicación de la complejidad en una interface interna de un módulo  $i$  y es definida:

$$D(i) = v(i) / [f_{out}(i) + 1]$$

donde  $v(i)$  es el número de variables de entrada y salida que se pasan al módulo  $i$  o se reciben de este.

Al final,  **$C(i)$  o la complejidad del sistema**. Se define como la suma de las complejidades estructurales y de los datos, y se procesa así

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

A medida que crecen los valores en la complejidad, esta aumenta globalmente. Esto hace que la probabilidad del esfuerzo aumente.

---

<sup>43</sup> Pressman 98

Existen varias *métricas de morfología*<sup>44</sup> simples (por ejemplo, forma) que permiten comparar diferentes arquitecturas de programa mediante un conjunto de dimensiones directas.

#### 4.6.2.2 Métricas al nivel de componente

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen medidas de la cohesión, acoplamiento y complejidad del módulo. Estas tres medidas pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de componentes.

Las métricas presentadas son de “caja blanca” en el sentido que requieren conocimiento del trabajo interno del módulo en cuestión. Las métricas de diseño en los componentes se pueden aplicar una vez que se ha desarrollado un diseño procedimental. Esta puede atrasarse hasta poseer el código fuente.

##### 4.6.2.2.1 Métricas de cohesión

Existen diferentes métricas cuya función es el indicar la cohesión (union) de un módulo<sup>45</sup>. Las métricas se definen con cinco conceptos y medidas:

- **Porción de datos:** Dicho simplemente, una porción de datos es una marcha atrás, a través de un módulo que busca valores de datos que afectan a la localización del módulo en el que empezó la marcha atrás. Debería resaltarse que se pueden definir tanto *porciones de programas* (que se centran en enunciados y condiciones) como *porciones de datos*.

---

<sup>44</sup> Fenton 91

<sup>45</sup> Bieman y Ott [Hamdi '99]

- **Símbolos léxicos (tokens) de datos:** Las variables definidas para un módulo pueden definirse como señales de datos para el módulo.
- **Señales de unión:** Es el conjunto de las señales que son encontradas en uno o más porciones de datos.
- **Señales de super-unión:** *Son las señales de los datos que son comunes de los módulos.*
- **Cohesión:** No es más que la unión entre señales de los módulos.

Los autores Bieman y Ott desarrollaron métricas para cohesiones funcionales fuertes (CFF), cohesiones funcionales débiles (CFD), y pegajosidad (el grado relativo con el que las señales de unión ligan juntas porciones de datos) Estas métricas se pueden interpretar de la siguiente manera:

Todas estas métricas de cohesión tienen valores que van desde 0 a 1. Tienen un valor de 0 cuando un procedimiento tiene más de una salida y no muestra ningún atributo de cohesión indicado por una métrica particular.

Un procedimiento sin señales de super-unión, sin señales comunes a todas las porciones de datos, no tiene una cohesión funcional fuerte (no hay señales de datos que contribuyan a todas las salidas).

Un procedimiento sin señales de unión, es decir, sin señales comunes a más de una porción de datos (en procedimientos con más de una porción de datos), no muestra una cohesión funcional débil y ninguna adhesividad (no hay señales de datos que contribuyan a más de una salida).

La cohesión funcional fuerte y la pegajosidad se obtienen cuando las métricas de Bieman y Ott toman un valor máximo de 1.



Esta es descripción breve de métricas anteriores. Sin embargo, como ilustración del carácter de dichas métricas y es cohesión funcional fuerte:

$$\mathbf{CFF(i) = SU(SA(i))/señales (i)}$$

donde **SU(SA(i))** denota señales de super-uni6n (el conjunto de señales de datos que se encuentran en todas las porciones de datos de un m6dulo i). Como la relaci6n de señales de super-uni6n con respecto al n6mero total de señales en un m6dulo i aumenta hasta un valor m6ximo de 1, la cohesi6n funcional del m6dulo tambi6n aumenta 1.

#### 4.6.2.2 Métricas de acoplamiento

El acoplamiento de m6dulo proporciona una indicaci6n de la “conectividad” de un m6dulo con otros m6dulos, datos globales y entorno exterior. Existe una métrica para el acoplamiento del m6dulo que combina el acoplamiento de flujo de datos y de control: acoplamiento global y acoplamiento de entorno.

Las medidas necesarias para calcular el acoplamiento de m6dulo se definen en t6rminos de cada uno de los tres tipos de acoplamiento apuntados anteriormente.

Para el acoplamiento de flujo de datos y de control:

$d_i$  = n6mero de par6metros de datos de entrada

$c_i$  = n6mero de par6metros de control de entrada

$d_o$  = n6mero de par6metros de datos de salida

$c_o$  = n6mero de par6metros de control de salida

Para el acoplamiento global

gd = número de variables globales usadas como datos

gc = numero de variables globales usadas como control

Para el acoplamiento de entorno:

w = número de módulos llamados (expansión)

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

Usando estas medidas, se define un indicador de acoplamiento de módulo,  $m_c$ , de la siguiente manera:

$$m_c = k/M$$

donde  $k = 1$  es una constante de proporcionalidad.

$$m = d_i + a * c_i + d_o + b * c_o + g_d + c * g_c + w + r$$

donde:

$$a=b=c=2$$

Cuanto mayor es el valor de  $m_c$ , menor es el acoplamiento de módulo. Por ejemplo, si un módulo tiene un solo parámetro de entrada y salida de datos, no accede a datos globales y es llamado por un solo módulo entonces:

$$m = 1/(1+0+1+0+0+0+ 1+0) = 1/3 = 0.33$$

Se debería esperar que un módulo como éste presentara un acoplamiento bajo. De ahí que, un valor de  $m_c = 0.33$  implica un acoplamiento bajo. Por el contrario, si un módulo tiene 5 parámetros de salida y 5 parámetros

de entrada de datos, un número igual de parámetros de control, accede a 10 elementos de datos globales y tiene una concentración de 3 y una expansión de 4,

$$m = 1/(5 + 2.5 + 5 + 2.5 + 1 + 0 + 21 + 4) = 0.02$$

el acoplamiento implicado es grande.

Para que aumente la métrica de acoplamiento a medida que aumenta el grado de acoplamiento, se puede definir una métrica de acoplamiento revisada:

$$C = 1 - mc$$

donde el grado de acoplamiento no aumenta linealmente un valor mínimo en el rango de 0,66 hasta un máximo que se aproxima a 1.0.

#### **4.6.2.2.3 Métricas de complejidad**

Se pueden calcular una variedad de métricas del software para determinar la complejidad del flujo de control del programa. Muchas de éstas se basan en una representación denominada grafo de flujo, un grafo es una representación compuesta de nodos y enlaces. Cuando se dirigen los enlaces, el grafo de flujo es un grafo dirigido.

Las métricas de complejidad pueden emplearse para predecir información sobre la fiabilidad y mantenimiento de sistemas software, también realimentan la información durante el proyecto de software para ayudar a controlar la actividad de diseño, en las pruebas y mantenimiento, proporcionan información sobre los módulos software para ayudar a resaltar las áreas de inestabilidad.

La métrica de complejidad más ampliamente usada para el software es la **complejidad ciclomática**, proporciona una medida cuantitativa para probar la dificultad y una indicación de la fiabilidad última.

Hay estudios experimentales, que indican una fuerte correlación entre la métrica de **complejidad ciclomática** y el número de errores que existen en el código fuente, así como el tiempo requerido para encontrar y corregir dichos errores.

La complejidad ciclomática puede emplearse para proporcionar una indicación cuantitativa del tamaño máximo del módulo. Recogiendo datos de varios proyectos de programación reales, ha averiguado que una complejidad ciclomática de 10 parece ser el límite práctico superior para el tamaño de un módulo, cuando la complejidad ciclomática de los módulos excedían ese valor, resultaba extremadamente difícil probar adecuadamente el módulo.

#### **4.6.2.3 Métricas de la interfaz**

Aunque existe una significativa cantidad de literatura sobre el diseño de interfaces hombre-máquina, se ha publicado relativamente poca información sobre métricas que proporcionen una visión interna de la calidad y facilidad de empleo de la interfaz.

La métrica de conveniencia de la representación es una métrica muy utilizada para diseño en interfaces hombre-maquina. Una interfaz gráfica de usuario (IGU) usa entidades de representación, **iconos, gráficos, gráficos, texto, menús, ventanas y otras** para ayudar al usuario a completar tareas.

Una tarea Utilizando Interfaz gráfica de usuario (IGU), este debe moverse de interfaz a otra.

Para un diseño de una IGU específica, se pueden asignar costos a cada secuencia de acciones de acuerdo con la siguiente relación:

$$\text{Costos} = \Sigma[\text{frecuencia de transición } (k_i) \times \text{costos de transición } (k_i)]$$

donde  $k$  es la transición  $i$  específica de una entidad de representación a la siguiente cuando se realiza una tarea específica. Esta suma se da con todas las transiciones de una tarea en particular o conjunto de tareas requeridas para conseguir alguna función de la aplicación.

El costo puede estar caracterizado en términos de tiempo, retraso del proceso o cualquier otro valor razonable, tal como la distancia que debe moverse el ratón entre entidades de la representación.

La conveniencia de la representación se define como:

$$CR = \text{Costo de la representación óptima}$$

$$CR = 100 \times [(CR) / (\text{costo de la representación propuesta})]$$

donde CR = para una representación óptima.

Para calcular la representación óptima de una IGU, la superficie de la interfaz (el área de la pantalla) se divide en una cuadrícula. Cada cuadro de la cuadrícula representa una posible posición de una entidad de la representación.

Para una cuadrícula con  $N$  posibles posiciones y  $K$  diferentes entidades de representación para colocar, el número posible de distribuciones se representa de la siguiente manera:

$$\text{Número posible de distribuciones} = [N! / (K! * (N - K)!)] * K!$$

La CR se emplea para valorar diferentes distribuciones propuestas de IGU y la sensibilidad de una representación en particular a los cambios en las descripciones de tareas (por ejemplo, cambios en la secuencia y/o frecuencia de transiciones).

Es importante apuntar que la selección de un diseño de IGU puede guiarse con métricas tales como CR, pero el árbitro final debería ser la respuesta del usuario basada en prototipos de IGU.

Puede haber una posibilidad de éxito si se prefiere la interfaz basándose exclusivamente en la opinión del usuario ya que el rendimiento medio de tareas de usuario y su satisfacción con la IGU están altamente relacionadas.

#### **4.6.2.4 Métricas de diseño orientado a objetos**

El Software Orientado a Objetos (OO) es fundamentalmente distinto del software que se desarrolla utilizando métodos convencionales. Las métricas para sistemas OO deben de ajustarse a las características que distinguen el software OO del software convencional.

Estas métricas hacen hincapié en el encapsulamiento, la herencia, complejidad de clases y polimorfismo. Por lo tanto las métricas OO se centran en métricas que se pueden aplicar a las características de encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos que hagan única a esa clase.

Como en todas las métricas los objetivos principales de las métricas OO se derivan del software convencional: comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto.

En un tratamiento detallado de las métricas del software para sistemas orientados a objetos, existe una descripción de las nueve características distintivas y mensurables de un diseño orientado a objetos:<sup>46</sup>

- **Tamaño:** el tamaño se define a partir de cuatro conceptos:
  - **Población** que se mide al tomar un conteo estático de entidades orientada a objetos como clases u operaciones.
  - **Volumen** son idénticas a las de la población, pero se recopilan dinámicamente. *Longitud* es una medida de una cadena de elementos de diseño interconectados como la longitud de un árbol de herencia.
  - **Funcionalidad** brindan un parámetro indirecto del valor que se entrega al cliente en sistemas orientada a objetos.
  
- **Complejidad:** como el tamaño, hay muchos conceptos diferentes de la complejidad del software, la complejidad se considera desde el punto de vista de las características estructurales, al examinar la manera en que se interrelacionan las clases de un diseño orientado a objetos.
  
- **Acoplamiento:** las conexiones físicas entre los elementos de un diseño orientado a objetos, como el número de colaboraciones entre clases, que representan el acoplamiento dentro de un sistema orientado a objetos.
  
- **Suficiencia:** un componente de diseño es suficiente si refleja plenamente todas las propiedades del objeto de dominio de la aplicación que se está modelando, o si la clase posee las características que debe tener.

---

<sup>46</sup> Whitmire 97

- **Grado de avance:** el grado de avance considera varios puntos de vista, pero indica indirectamente el grado en que puede reutilizarse el componente de diseño.
- **Cohesión:** el grado de cohesión de una clase se determina al examinar el grado en que el conjunto de propiedades que posee es parte del dominio del problema o el diseño.
- **Primitivismo:** es el grado en que una operación es atómica (es decir, la operación no puede construirse a partir de una secuencia de otras operaciones contenidas dentro de una clase). Una clase que muestra un alto grado de primitivismo sólo encapsula operaciones primitivas.
- **Similitud:** esta medida indica el grado en que dos o más clases son similares en cuanto a su estructura, función comportamiento o propósito.
- **Volatilidad:** la volatilidad de un componente de diseño orientado a objetos mide la probabilidad de que ocurra un cambio, ya que los cambios de diseño ocurren cuando los requisitos se modifican o cuando las modificaciones se presentan en otra parte de una aplicación.

#### 4.6.2.4.1 Métricas orientadas a clases

Se sabe que la clase es la unidad principal de todo sistema orientado a objetos. Por consiguiente, las medidas y métricas para una clase individual, la jerarquía de clases, y las colaboraciones de clases resultarán sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño. Se ha visto que la clase encapsula a las operaciones (procesamiento) y a los atributos (datos).



La clase suele ser el “predecesor” de las subclases que heredan sus atributos de operaciones y suele colaborar con otras clases. Todas estas características ayudan a formar un grupo de métricas que se utilizan especialmente para las clases.

Uno de los conjuntos de métricas de software orientado a objetos, a los que se hace referencia más ampliamente es la basadas en clases, a las cuales suele aludirse con el nombre de conjunto de métricas CK para sistemas orientados a objetos.

### **Métodos ponderados por clase (MCP)**

Los métodos ponderados por clase (**MPC**), son aquellos en donde se definen **n** métodos de complejidad **C1, C2..., Cn**, para una clase C. La métrica de complejidad específica que se selecciona debe de normalizarse de tal modo que la complejidad nominal para un método tome el valor 1.0.

$$MPC = \sum C_i$$

para  $i = 1$  a  $n$ .

La complejidad y su número es un indicador de la cantidad de esfuerzo que se necesita para implementar y verificación de una clase. Además, si es mayor el número de métodos esto hará más complejo será el árbol de herencia.

Finalmente, a medida que el número de métodos crece para una clase dada, es más probable que se vuelva cada vez más específico de la aplicación, imitando por tanto su potencial de reutilización. Por todas estas razones, MPC debería de mantener un valor tan bajo como sea razonable.

Aun cuando podría parecer relativamente sencillo desarrollar un contador del número de métodos de una clase, el problema es en realidad más complejo

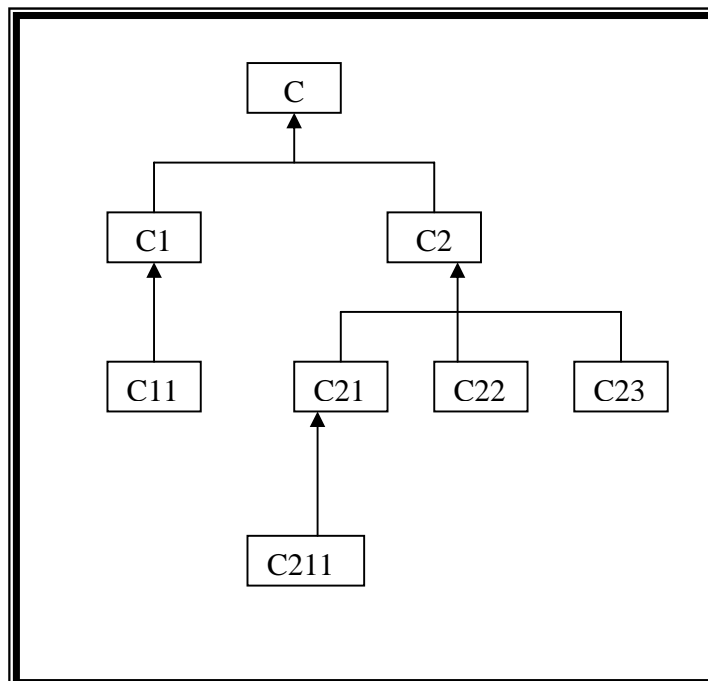
de lo que parece. Debe desarrollarse un enfoque de conteo consistente para los métodos.

### Árbol de Profundidad de Herencia (APH)

Esta es la que da la longitud máxima desde el nodo focalizado hasta la raíz de él árbol. Si se toma como referencia la figura 4.2, el valor de APH para la jerarquía de clases resultante es 4.

Con el crecimiento de APH, existe la probabilidad de que clases en niveles inferiores hereden una cantidad grande métodos. De esto hace difícil la predicción del comportamiento de alguna clase. Una jerarquía de clases lleva también a una mayor complejidad de diseño. Viendo el lado positivo, el poseer grandes valores en un árbol de profundidad de herencia ayuda a reutilizar más métodos.

**Figura 16. Jerarquía de clases**



## **Número de Descendientes (NDD)**

Las subclases que son inmediatamente subordinadas a una clase son denominan descendientes. En la Figura 16 la clase C2 tiene tres descendientes las subclases C21, C22 y C23.

A medida que crece el número de descendientes, se incrementa la reutilización, pero también es cierto, que a medida que crece NDD, la abstracción representada por la clase predecesora puede verse diluida.

Entonces, existe la posibilidad de que algunos de los descendientes no sean realmente miembros propios de la clase predecesora. Si el número de descendientes crece, las pruebas crecerán.

## **Respuesta para una clase (RPC)**

El conjunto de métodos que pueden ser ejecutados como respuesta a mensajes recibidos por un objetos es conocido como conjunto de respuestas o RPC y es definida como el numero de métodos que existen en ese conjunto de respuestas.

A medida que crece RPC, el esfuerzo necesario para la comprobación crece, porque la sucesión de comprobación va creciendo y también la complejidad global de diseño de la clase crece.

#### 4.6.2.4.2 Métricas orientadas a objetos: la colección de métricas para el diseño orientado a objetos<sup>47</sup>

Existe un conjunto de métricas para diseño orientado a objetos que proporcionan indicadores cuantitativos para las características del diseño orientado a objetos. A continuación se presenta una pequeña muestra de estas:

##### **Método de Factor de Herencia (MFH)**

El grado en que la arquitectura de clases de un sistema orientado a objetos usa la herencia para métodos u operaciones y atributos se define como:

$$MFH = \frac{\sum M_i(C_i)}{\sum m_a(C_i)}$$

Donde la sumatoria se presenta desde  $i=1$  hasta  $T_c$ ,  $T_c$  se define como el número total de clases en la arquitectura;  $C_i$  es una clase dentro de la arquitectura y

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

Donde

$M_a(C_i)$  = el número de métodos que pueden invocarse en asociación con  $C_i$ .

$M_d(C_i)$  = el número de métodos declarados en la clase  $C_i$ .

$M_i(C_i)$  = el número de métodos heredados (y no redefinidos) en  $C_i$ .

El valor de HFH es un indicativo del impacto de la herencia en el software orientado a objetos.

---

<sup>47</sup> Harrison, Counsell y Nithi '98

## Factor de Acoplamiento (FA)

El acoplamiento es una indicación de las conexiones entre elementos de un diseño orientado a objetos. El conjunto de métricas del diseño orientado a objetos define el acoplamiento de la siguiente manera:

$$FA = \frac{\sum_i \sum_j es\_cliente(C_i, C_j)}{(T_C^2 - T_C)}$$

Donde las sumatorias van desde  $i=1$  hasta  $T_C$  y desde  $j=1$  hasta  $T_C$ . La función

$Es\_cliente = 1,$

si y sólo si existe una relación entre la clase cliente,  $C_c$ , y la clase servidor,  $C_s$ , y  $C_c$  diferente  $C_s$

$=0,$  en cualquier otro caso.

Aunque muchos factores afectan la complejidad, la facilidad de comprensión y el mantenimiento del software, resulta razonable concluir que, a medida que aumenta el valor de FA, también aumentará la complejidad del software orientado a objetos, de igual manera como consecuencia, es posible que resulten afectadas la facilidad de comprensión y mantenimiento, junto con la posibilidad de reutilización.

### 4.6.3 Métricas para el código fuente

Las medidas fueron desarrolladas por Maurice Halstead para determinar una medida cuantitativa de la complejidad directa de los operarios y operandos en el módulo.

Entre las métricas de software más avanzadas, éstos son indicadores fuertes de complejidad de código, existen la evidencia de que las medidas de

Halstead son útiles durante el desarrollo, para valorar la calidad de código en aplicaciones computablemente-densas. Las medidas de Halstead se basan en cuatro números de escalar, que serán derivados directamente del código fuente del programa, tales como:

$n_1$  = el número de distintos operadores que aparecen en un programa

$n_2$  = el número de distintos operandos que aparecen en un programa

$N_1$  = el número total de veces que aparece el operador

$N_2$  = el número total de veces que aparece el operando

Halstead usa las medidas primitivas para desarrollar expresiones para la **longitud** global del programa; **volumen** mínimo potencial para un algoritmo; el **volumen real** (número de bits requeridos para especificar un programa); el **nivel del programa** (una medida de la complejidad del software); **nivel del lenguaje** (una constante para un lenguaje dado); y otras características tales como **esfuerzo de desarrollo**, **tiempo de desarrollo** e incluso el **número esperado de fallos** en el software.

Halstead expone que la longitud  $N$  se puede estimar como:

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

y el volumen de programa se puede definir como:

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

Se debería tener en cuenta que  $V$  variará con el lenguaje de programación y representa el volumen de información.

En teoría debe de existir un mínimo volumen para un algoritmo. Halstead define una relación de volumen L como la relación volumen de la forma más compacta de un programa respecto al volumen real del programa. Por tanto L, debe ser siempre menor de uno. En términos de medidas primitivas, la relación de volumen se puede expresar como:

$$L = 2/n1*n2/N2$$

Halstead propuso que todos los lenguajes pueden categorizarse por nivel de lenguaje, y que variará según los lenguajes. Halstead creó una teoría que suponía que el nivel de lenguaje es constante para un lenguaje dado.

#### **4.6.4 Métricas para las pruebas**

Aunque se ha escrito mucho sobre métricas del software para pruebas, la mayoría de las métricas propuestas se concentran en el proceso de pruebas, no en las características técnicas de las pruebas mismas. En general, los responsables de las pruebas deben confiar del análisis, diseño y de código hecho, que les guíen en la elaboración y ejecución los casos a poner a prueba en el sistema.

El esfuerzo global de las pruebas puede predecirse utilizando las métricas basadas en la función. Se pueden juntar y correlacionar varias características a nivel de proyecto (p ej.: esfuerzo y tiempo las pruebas, errores descubiertos, número de casos de prueba producidos) de proyectos anteriores con el número de puntos de función producidos por un equipo del proyecto. El equipo puede después predecir los valores “esperados” de estas características del proyecto actual.

La métrica BANG puede proporcionar una indicación del número de casos de prueba necesarios para examinar las medidas primitivas. El número

de primitivas funcionales (Pfu), elementos de datos, (ED), objetos (OB), relaciones (RE), estados (ES) y transiciones (TR) pueden emplearse para predecir el número y tipos de pruebas de la caja negra y blanca del software. Por ejemplo, el número de pruebas asociadas con la interfaz hombre-máquina se puede estimar examinando:

- El número de transiciones (TR) contenidas en la representación estado-transición del IHM y evaluando las pruebas requeridas para ejecutar cada transición.
- Cuantos objetos se mueven en las interfaces.
- Cuantos elementos son introducidos o que el sistema genere.

Las métricas de diseño de alto nivel proporcionan información sobre facilidad o dificultad asociada con la prueba de integración y la necesidad de software especializado de prueba (como: uso de matrices y o controladores).

La complejidad ciclomática (una métrica de diseño de componentes) se encuentra en el núcleo de las pruebas de caminos básicos, un método de diseño de casos de prueba. Además, se pueden elegir módulos candidatos para realizárseles pruebas más profundas si se utiliza la complejidad ciclomática. 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.



Si se utiliza la definición del volumen  $V$  de un programa, y usando  $NP$  como el nivel del programa, el esfuerzo del software se calculará;

$$NP = 1 / [(n1 / 2) * (N2 / n2)]$$

$$e = V / NP$$

El porcentaje del esfuerzo global de pruebas a asignar un módulo  $k$  para estimar usando la siguiente relación:

$$\text{porcentaje de esfuerzo de pruebas } (k) = e(k) / \sum e(i)$$

donde  $e(k)$  se calcula para el módulo  $k$  usando las ecuaciones anteriores y donde la sumatoria en el denominador de la ecuación de “porcentaje de esfuerzo” es la sumatoria del esfuerzo de la ciencia del software a lo largo de todos los módulos del sistema.

Existen tres medidas que proporciona una indicación qué tan compleja es una prueba. Una de ellas es la **amplitud de la prueba** que proporciona una indicación de cuantos módulos se han probado. La **profundidad de la prueba** es una medida que ayuda determinar la amplitud de los caminos independientes. Se puede tener un número exacto de caminos sumando la complejidad ciclomática de la totalidad de los módulos del programa.

Finalmente, a medida que se van haciendo las pruebas y se recogen los datos de los errores, se pueden emplear los *perfiles de fallos* para dar prioridad y categorizar los errores descubiertos.

El grado de prioridad indica que tan severo es un problema. Al hacer una categorización de fallos se puede hacer una descripción del error, y de esa manera que se puedan llevar a cabo análisis estadísticos de los errores.

#### 4.6.5 Métricas de mantenimiento

Se han propuesto métricas diseñadas explícitamente para actividades de mantenimiento. El estándar IEEE 982.1-1988<sup>48</sup> sugiere un **índice de madurez del software (IMS)** que proporciona una indicación de la estabilidad de un producto de software (basada en los cambios que ocurren con cada versión del producto) Se determina la siguiente información:

*Mr* = número de módulos en la versión actual

*Fc* = número de módulos en la versión actual que se han cambiado

*Fa* = número de módulos en la versión actual que se han añadido

*Fd* = número de módulos de la versión anterior que se han borrado en la versión actual

El índice de madurez del software se calcula de la siguiente manera:

$$\text{IMS} = [\text{Mr} - (\text{Fa} + \text{Fc} + \text{Fd})] / \text{Mr}$$

A medida que el IMS se aproxima a 1.0 el producto se empieza a estabilizar. El índice de madurez del software se puede emplear como métrica de la planificación de mantenimiento al software. El tiempo medio para producir una versión de un producto software puede correlacionarse con el IMS desarrollándose modelos empíricos para el mantenimiento.

---

<sup>48</sup> <http://odysseus.ieee.org/ieeesearch/query.html?qt=ieee+software+maintenance&charset=iso-8859-1>

## **5 TÉCNICAS DE ESTIMACIÓN DE COSTOS DE SOFTWARE**

Dentro de la mayor parte de organizaciones, la estimación de costos de la programación se basa en experiencias pasadas. Los datos históricos se usan para identificar los factores de costo y determinar la importancia relativa de los diversos factores dentro de la organización.

Los datos empíricos que soportan la mayoría de los modelos de estimación se obtienen de una muestra limitada de proyectos. Es por eso, que estos modelos de estimación no son adecuados para toda clase de software y todos los entornos de desarrollo. Por consiguiente, los resultados obtenidos de dichos modelos se deben utilizar con prudencia.

Un modelo de estimación debe calibrarse para reflejar las condiciones locales. El modelo debe probarse mediante la aplicación de datos recopilados a partir de proyectos completados, colocar los datos en el modelo y luego comparar los resultados reales con los ingresados. Si la concordancia es insuficiente, el modelo debe afinarse y ponerse a prueba nuevamente antes de que se pueda utilizar.

### **5.1 Juicio de un experto**

La técnica más utilizada para la estimación de costos es el uso del juicio experto, este se basa en la experiencia, en el conocimiento anterior y en el sentido comercial de uno o mas individuos dentro de la organización.

La mayor ventaja del juicio de un experto que es la experiencia, puede llegar a ser su debilidad; debido a que puede confiarse de que el proyecto sea similar al anterior pero bien puede suceder que haya olvidado algunos factores que ocasionan que el sistema nuevo sea significativamente diferente; o quizás, el que realiza la estimación no tenga experiencia en ese tipo de proyecto.

Para compensar de alguna manera ese tipo de problemas, los grupos de expertos algunas veces tratan de llegar a un consenso en el estimado, con lo cual se trata de minimizar las fallas individuales y la falta de familiaridad en proyectos particulares neutralizando las tendencias personales y el deseo de ganar un contrato a través de una estimación optimista.

## 5.2 La estructura de los modelos de estimación

Un modelo común de estimación se da utilizando análisis de regresión sobre los datos compilados de proyectos anteriores de software. La estructura, global de tales modelos adquiere la forma de:

$$E = A + B * (e_v)^C$$

donde  $A$ ,  $B$  y  $C$  son constantes conseguidas empíricamente,  $E$  es el esfuerzo en meses/persona, y  $e_v$  es la variable de estimación (de Líneas de código o Puntos de Función).

Además de la dependencia señalada en la ecuación, la mayoría de los modelos de estimación tienen algún componente de ajuste del proyecto que permite ajustar  $E$  debido a otras características del proyecto tales como: complejidad del problema, experiencia del personal, entorno de desarrollo.

Entre los muchos modelos de estimación orientados a Líneas de Código se encuentran los siguientes:

**Tabla IV. Modelos de estimación orientados a líneas de código<sup>29</sup>**

$E=5.2 * (KLDC)^{0.91}$	<b>Modelo de Walston-Felix</b>
$E = 3.2 * (KLDC)^{1.05}$	<b>Modelo Simple de Boehm</b>
$E=5.288 * (KLDC)^{1.047}$	<b>Modelo Doty para KLDC &gt; 9</b>
$E= 5.5 + 0.73 * (KLDC)^{1.16}$	<b>Modelo de Bailey-Basisli</b>

También se han propuesto los modelos orientados a Puntos Función. Entre estos modelos se incluyen:

**Tabla V. Modelos de estimación orientados a puntos de función<sup>30</sup>**

$E = -91.4+0.355 PF$	<b>Modelo de Albrecht y Faffney</b>
$E = -37 + 0.96 PF$	<b>Modelo de Kemerer</b>
$E = -12.88 + 0.405 PF$	<b>Modelo de regresión para proyecto pequeño</b>

Una rápida exploración de los modelos listados arriba indica que cada uno traerá un resultado diferente para el mismo valor de KLDC y PF. Los modelos de estimación se deben evaluar para necesidades particulares.

### **5.3 El Modelo COCOMO**

Existe una jerarquía de modelos de estimación de costos, dentro de ella existe un modelo conocido como **COCOMO**<sup>31</sup>, el cual se volvió de los mas ampliamente utilizado y estudiados de los modelos de estimación.

<sup>29</sup> Roger S. Pressman 2005

<sup>30</sup> Roger S. Pressman 2005

<sup>31</sup> <http://es.wikipedia.org/wiki/COCOMO>

COCOMO es una jerarquía de modelos de estimación de costes de software que incluye submodelos:

- **Modelo 1:** El modelo COCOMO *básico* calcula el esfuerzo (y el costo) del desarrollo de software en función del tamaño del programa, expresado en las líneas estimadas de código (LDC).
- **Modelo 2:** El modelo COCOMO *intermedio* calcula el esfuerzo del desarrollo de software en función del tamaño del programa y de un conjunto de “conductores de costo” que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.
- **Modelo 3:** El modelo COCOMO *avanzado* incorpora todas las características de la versión intermedia y lleva a cabo una evaluación del impacto de los conductores de costo en cada fase (análisis, diseño, etc.) del transcurso de ingeniería del software.

Los modelos COCOMO están establecidos para tres prototipos de proyectos de software:<sup>32</sup> que empleando la terminología de Boehm son:

1. **Modo orgánico:** aquellos proyectos de software que son respectivamente pequeños y sencillos en donde trabajan pequeños equipos que poseen buena experiencia en la aplicación, sobre un conjunto de requisitos poco rígidos.
2. **Modo semiacoplado:** son los proyectos de software intermedios hablando de tamaño y complejidad, en donde los equipos tienen diversos niveles de experiencia, y además deben satisfacer requerimientos poco o medio rígidos.

---

<sup>32</sup> B. W. Boehm '81

3. **Modo empotrado:** son proyectos de software que deben ser desarrollados en un conjunto de hardware, software y restricciones operativas muy restringido.

Las ecuaciones de estimación del esfuerzo de desarrollo tienen la forma:

$$E = a_i S^{b_i} m(X)$$

con

- S el número de miles de líneas de código fuente.
- $m(X)$  es un multiplicador que depende de 15 atributos (Tabla VI).

**Tabla VI. Factores multiplicadores de esfuerzo en COCOMO<sup>33</sup>**

<b>Factor de Multiplicación</b>	<b>Intervalo de los valores</b>
<b>Atributos del producto</b>	
Confiabilidad requerida	0.75 a 1.40
Tamaño de la base de datos	0.94 a 1.16
Complejidad del producto	0.70 a 1.65
<b>Características de la máquina</b>	
Limitantes en el tiempo de ejecución	1.00 a 1.66
Limitaciones en memoria principal	1.00 a 1.56
Volatilidad de la virtualidad en la maquina	0.87 a 1.30
Tiempo de entrega de programas	0.87 a 1.15
<b>Características del personal</b>	
Capacidad de los analistas	1.46 a 0.71
Capacidad de los programadores	1.42 a 0.70
Experiencia en programas de aplicación	1.29 a 0.82
Experiencia en máquinas virtuales	1.21 a 0.90
Experiencia en lenguajes de programación	1.14 a 0.95
<b>Características del proyecto</b>	
Uso de técnicas modernas de programación	1.24 a 0.82
Uso de herramientas de programación	1.24 a 0.83
Tiempo requerido para el desarrollo	1.23 a 1.10

<sup>33</sup> Richard Fairley, Ingeniería del software 1993



Y los coeficientes  $a_i$  y  $b_i$  son respectivamente:

**Tabla VII. Constantes para calcular el esfuerzo del modelo COCOMO<sup>34</sup>**

Modo	Básico		Intermedio	
	$a_i$	$b_i$	$a_i$	$b_i$
<b>Orgánico</b>	2.4	1.05	3.2	1.05
<b>Semiencajado</b>	3.0	1.12	3.0	1.12
<b>Empotrado</b>	3.6	1.2	2.8	1.2

### 5.3.1 Modelo Básico

Este modelo trata de estimar, de una manera rápida y más o menos burda, la mayoría de proyectos pequeños y medianos. Se consideran tres modos de desarrollo en este modelo: orgánico, semiencajado y empotrado.

#### 5.3.1.1 Modo orgánico

En este modo, un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía de unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles de líneas (medio), mientras que en los otros dos modos el tamaño varía de pequeño a muy grandes (varios cientos de miles de líneas). En este modo, al igual que en los otros, el coste se incrementa a medida que el tamaño lo hace, y el tiempo de desarrollo se alarga.

Se utilizan dos ecuaciones para determinar el esfuerzo de personal y el tiempo de desarrollo. El coste es

---

<sup>34</sup> B. W. Boehm '81

$$K_m = 2.4 S_k^{1.05}$$

donde  $K_m$  se expresa en personas-mes y  $S_k$  es el tamaño expresado en miles de líneas de código fuente. El tiempo de desarrollo se da por

$$t_d = 2.5 K_m^{0.38}$$

donde  $K_m$  se obtiene de la ecuación anterior y  $t_d$  es el tiempo de desarrollo en meses. Estas ecuaciones se han obtenido por medio de ajustes de curvas realizado por Boehm en TRW sobre 63 proyectos.

### **5.3.1.2 Modo empotrado**

En este modo, el proyecto tiene unas fuertes restricciones, que pueden estar relacionadas con el procesador y el interface hardware. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Las estimaciones de tiempo y coste se basan en las mismas ecuaciones que en el modo orgánico, pero con diferentes constantes. Así, el coste se da por

$$K_m = 3.6 S_k^{1.20}$$

y el tiempo de desarrollo por

$$t_d = 2.5 K_m^{0.32}$$

### 5.3.2 Modelo Intermedio

En este modelo se introducen 15 atributos de coste para tener en cuenta el entorno de trabajo. Estos atributos se utilizan para ajustar el coste nominal del proyecto al entorno real, incrementando la precisión de la estimación.

#### 5.3.2.1 Ecuaciones nominales de coste

Para cada modo de desarrollo, los 15 atributos del coste intervienen como multiplicadores en el coste nominal,  $K_n$ , para producir el coste ajustado.

Las ecuaciones nominales de coste para el modelo intermedio son:

modo orgánico	$K_n = 3.2 S_k^{1.05}$
modo semiencajado	$K_n = 3.0 S_k^{1.12}$
modo empotrado	$K_n = 2.8 S_k^{1.20}$

Hay que notar que:

- los exponentes son los mismos que los del modelo básico, confirmando el papel que representa el tamaño.
- los coeficientes de los modos orgánico y empotrado han cambiado, para mantener el equilibrio alrededor del semiencajado con respecto al efecto multiplicador de los atributos de coste.

#### 5.3.2.2 Atributos de coste

Estos atributos tratan de capturar el impacto del entorno del proyecto en el coste de desarrollo. De un análisis estadístico de más de 100 factores que

influyen el coste, Boehm retuvo 15 de ellos para COCOMO (Tabla VI).

### 5.3.3 Modelo avanzado

Este modelo, puede procesar todas las características del proyecto para construir una estimación. Existen dos principales:

1. Multiplicadores de esfuerzo sensitivos a la fase. Algunas fases se ven más afectadas que otras por los atributos. El modelo detallado proporciona un conjunto de multiplicadores de esfuerzo para cada atributo. Esto ayuda a determinar la asignación del personal para cada fase del proyecto.
2. Jerarquía del producto a tres niveles. Se definen tres niveles de producto. Estos son módulo, subsistema y sistema. La cuantificación se realiza al nivel apropiado, esto es, al nivel al que es más susceptible la variación.

### 5.4 La ecuación del software

La **ecuación del software**<sup>35</sup> es un modelo multivariable que asume una distribución específica del esfuerzo a lo largo de la vida de un proyecto de desarrollo de software. El modelo se ha obtenido a partir de los datos de productividad para unos 4,000 proyectos actuales de software. Un modelo de estimación tiene esta forma:

$$E=[LDC * B^{0.333} / P]^3 * (1/t^4)$$

Donde

E= esfuerzo en personas-mes o personas-año

---

<sup>35</sup> Norman E. Fenton'91

t = duración del proyecto ya sea en meses o años

B = "factor especial de destrezas"<sup>36</sup>

P = "parámetro de productividad" que refleja:

- Madurez global del proceso y de las prácticas de administración
- La amplitud hasta donde se utilizan correctamente las normas de la ingeniería del software.
- El nivel de los lenguajes de programación utilizados
- Las habilidades y la experiencia del equipo del software.
- La complejidad de la aplicación.

Es importante señalar que la ecuación del software tiene dos parámetros dependientes:

1. La estimación del tamaño en LDC (Líneas de Código)
2. Una indicación de la duración del proyecto en meses o años.

Para simplificar el proceso de estimación y utilizar una forma más común para el modelo, existe un conjunto de ecuaciones obtenidas de la ecuación del software<sup>37</sup>.

Un tiempo mínimo de desarrollo se define como:

$$t_{min} = 8.14 (LDC/PP)^{0.43} \text{ en meses para } t_{min} > 6 \text{ meses}$$

---

<sup>36</sup> B aumenta lentamente conforme "Crecen las necesidad de integración, las pruebas, la garantía de calidad, la documentación y las habilidades de gestión"

<sup>37</sup> Pressman sexta edición

$E = 180Bt^3$  en personas-mes para  $E \geq 20$  personas-mes (t en años)

### 5.5 El modelo CMM<sup>38</sup> (Modelo de Capacidad y Madurez)

El Modelo de Capacidad y Madurez o **CMM (Capability Maturity Model)**, es un modelo de evaluación de los procesos de una organización. Fue desarrollado inicialmente para los procesos relativos al software por la Universidad Carnegie-Mellon para el SEI (Software Engineering Institute<sup>39</sup>).

El **Modelo de Capacidad y Madurez** en la Ingeniería de Sistemas fue publicado por el SEI en noviembre de 1995. Está dedicado a las actividades de ingeniería de sistemas. "CMM" es una marca registrada del SEI.

A partir de noviembre de 1986 el SEI, a requerimiento del gobierno federal de los Estados Unidos de América, desarrolló una primera definición de un modelo de madurez de procesos en el desarrollo de software, que se publicó en septiembre de 1987.

Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso (KPA - *Key Process Area*). Para cada área de proceso define un conjunto de buenas prácticas que habrán de ser:

- Definidas en un procedimiento documentado
- Provistas (la organización) de los medios y formación necesarios
- Ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas)
- Medidas
- Verificadas

---

<sup>38</sup> [http://es.wikipedia.org/wiki/Modelo\\_de\\_Capacidad\\_y\\_Madurez](http://es.wikipedia.org/wiki/Modelo_de_Capacidad_y_Madurez)

<sup>39</sup> **Federally Funded Research and Development Centers (FFRDCs)**

A su vez estas Áreas de Proceso se agrupan en cinco "niveles de madurez", de modo que una organización que tenga institucionalizadas todas las prácticas incluidas en un nivel y sus inferiores, se considera que ha alcanzado ese nivel de madurez.

Los niveles son:

- **Inicial.** Las organizaciones en este nivel no disponen de un ambiente estable para el desarrollo y mantenimiento de software. Aunque se utilicen técnicas correctas de ingeniería, los esfuerzos se ven minados por falta de planificación. El éxito de los proyectos se basa la mayoría de las veces en el esfuerzo personal, aunque a menudo se producen fracasos y casi siempre retrasos y sobrecostos. El resultado de los proyectos es impredecible.
- **Nivel repetible.** En este nivel las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos, existen unas métricas básicas y un razonable seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente.
- **Definido.** Además de una buena gestión de proyectos, a este nivel las organizaciones disponen de correctos procedimientos de coordinación entre grupos, formación del personal, técnicas de ingeniería más detalladas y un nivel más avanzado de métricas en los procesos. Se implementan técnicas de revisión por pares (*peer reviews*).
- **Gestionado.** Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad.
- **Optimizado.** La organización completa está volcada en la mejora

continúa de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

Así es como el modelo CMM establece una medida del progreso conforme avanza, en niveles de madurez. Cada nivel a su vez cuenta con un número de áreas de proceso que deben lograrse. El alcanzar estas áreas se detecta mediante la satisfacción o insatisfacción de varias metas claras y cuantificables.

Con la excepción del primer Nivel, cada uno de los restantes Niveles de Madurez está compuesto por un cierto número de Áreas Claves de Proceso, conocidas a través de la documentación del CMM por su sigla inglesa: KPA.

Cada KPA identifica un conjunto de actividades y prácticas interrelacionadas, las cuales cuando son realizadas en forma colectiva permiten alcanzar las metas fundamentales del proceso. Las KPA's pueden clasificarse en 3 tipos de proceso: **Gestión, Organizacional e Ingeniería.**

Los cinco niveles definidos por el SEI se obtienen como consecuencia de evaluar las respuestas de un cuestionario de evaluación basado en el CMM. Los resultados del cuestionario se filtran en un único grado numérico que proporciona una indicación de la madurez del proceso de una organización.

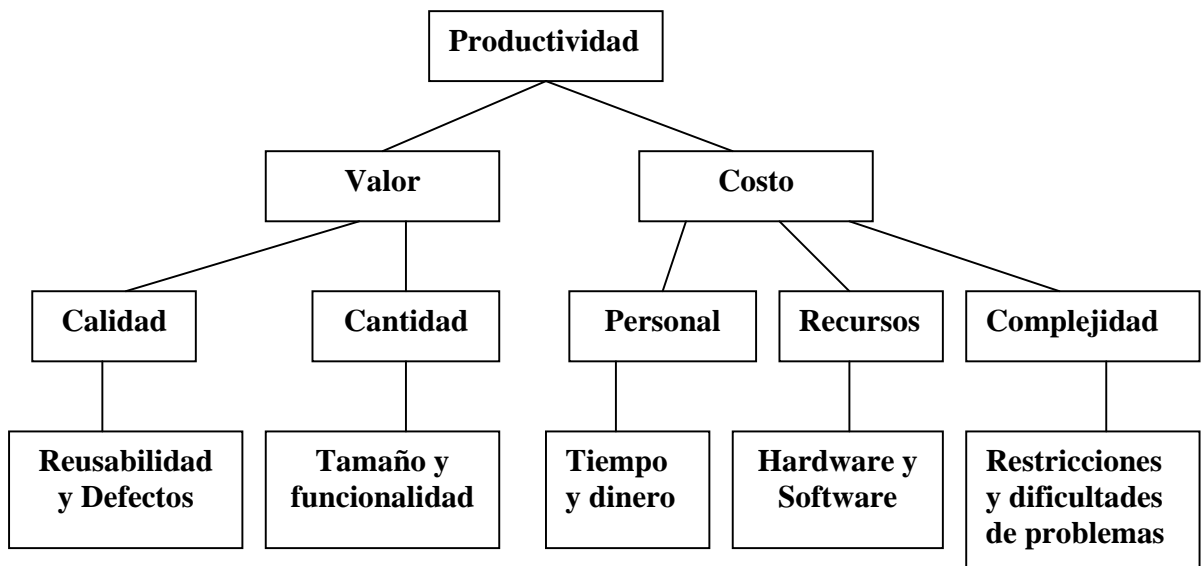
A pesar de la aceptación internacional, el CMM cuenta con críticas. Las críticas más serias son referentes a la validez de la escala de cinco niveles ya que no existe evidencia convincente de que las empresas que se encuentran en el nivel mas alto (nivel cinco) produzcan software de mejor calidad.



## 5.6 Modelo de productividad.

El modelo de productividad<sup>40</sup>, representada en la figura 17, hace énfasis en el hecho que la productividad es una función tanto del valor como de costo. En donde el valor se ve afectado por factores de calidad como de cantidad y los costos por los recursos, complejidad y personal.

Figura 17. Modelo de productividad



## 5.7 Estimación para proyectos orientados a objetos

Vale la pena completar los métodos convencionales de estimación de costo del software con un enfoque diseñado explícitamente para software orientado a objetos. Algunos autores sugieren el enfoque siguiente:<sup>41</sup>

<sup>40</sup> Fenton '91

<sup>41</sup> Lorenz y Kidd

1. Desarrollar estimaciones aplicando la descomposición de esfuerzo, análisis de Puntos de Función y cualquier otro método que sea aplicable en aplicaciones convencionales.
2. Aplicar el modelado de análisis orientado a objetos, desarrollar casos de uso y determinar un conteo. Reconocer que el número de casos de uso puede cambiar conforme avance el proyecto.
3. A partir del modelo de análisis, determinar el número de clases clave.
4. Categorizar el tipo de interfaz para la aplicación y desarrollar un multiplicador para las clases de soporte.

**Tabla VIII. Tipo de interfaz y multiplicador**

<b>Tipo de Interfaz</b>	<b>Multiplicador</b>
Sin IUG (Interfaz de Usuario Gráfica)	2.0
Interfaz del usuario basada en texto	2.25
Con IUG	2.25
IUG Compleja	3.0

5. Multiplicar el número total de clases (clave + soporte) por el número promedio de unidades de trabajo por clase. Los autores Lorenz y Kidd sugieren de 15 a 20 personas-día por clase.
6. Comprobar de manera cruzada la estimación basada en clase al multiplicar el número promedio de unidades de trabajo por caso de uso.

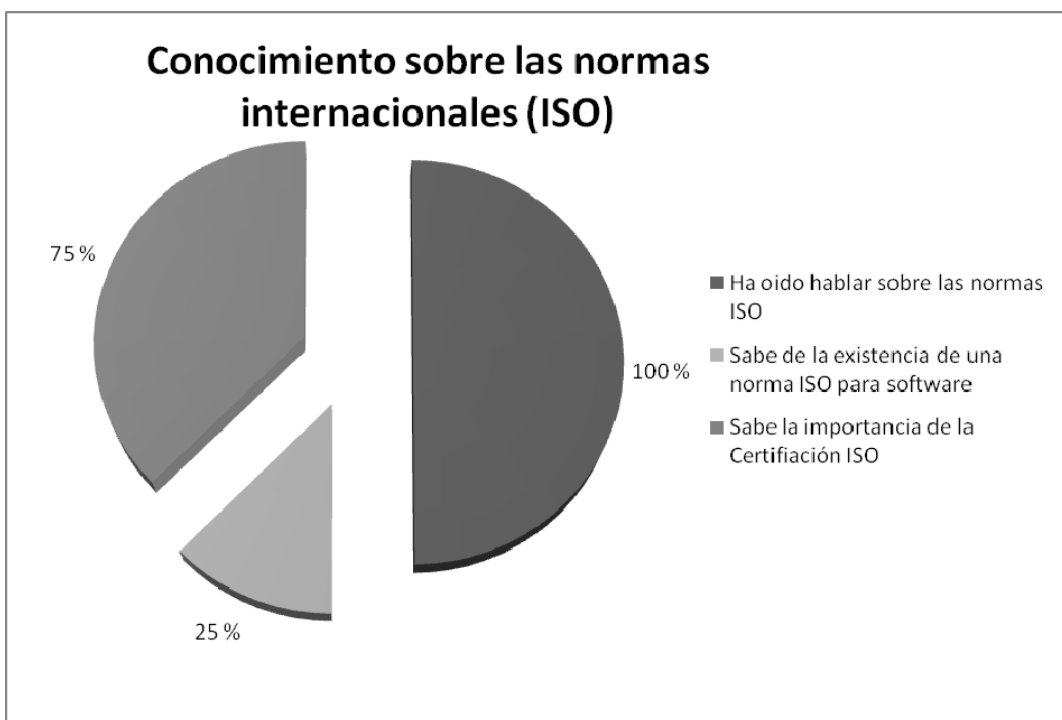
Por lo general, las estimaciones precisas de un proyecto emplean al menos dos de tres técnicas, al comparar las estimaciones obtenidas con la aplicación de diferentes técnicas, el planificador tiene más probabilidad de calcular y lograr tener con exactitud la duración del proyecto de software sin dejar por un lado la calidad.



## 6 TRABAJO DE CAMPO

La realización del trabajo de campo se llevó a cabo con la toma de datos a través de una encuesta, para algunas empresas que se dedican al desarrollo de software, los resultados son los siguientes:

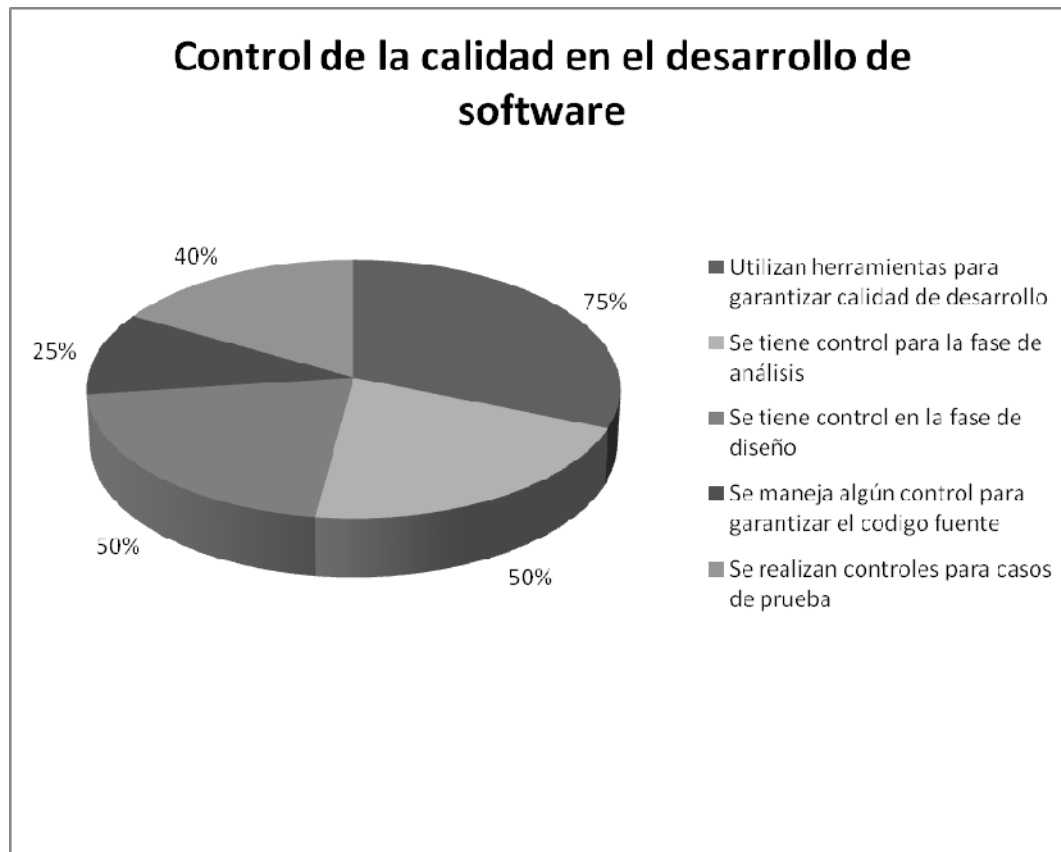
**Figura 18. Gráfico de conocimiento sobre las normas internacionales (ISO).**



**Fuente: Elaborado por Walter Vásquez**

Como puede observarse, la mayoría de los encuestados ha oído sobre las normas internacionales, pero, tan solo un 25% tenía conocimiento de la existencia de una norma internacional que fuera aplicable en el desarrollo de software, y el 75% sabe de la importancia de estar certificado internacionalmente y que se vería como una ventaja competitiva en un mercado globalizado.

**Figura 19. Gráfico de control de calidad en el desarrollo de software.**



**Fuente: Elaborado por Walter Vásquez**

Puede observarse que el 75% de las empresas utilizan herramientas que les ayudan a garantizar la calidad, y existe un porcentaje igual de 50% en las etapas análisis y diseño que ayudaran a fijar los cimientos de proyecto a desarrollar, se puede observar que una vez controladas las fases anteriores existe un porcentaje de 40% dedicado a las fases de pruebas esto debido a que se han exigido en otras fases mayor atención, y por último, se tiene un 25% en la realización de controles al momento del desarrollo de código fuente.

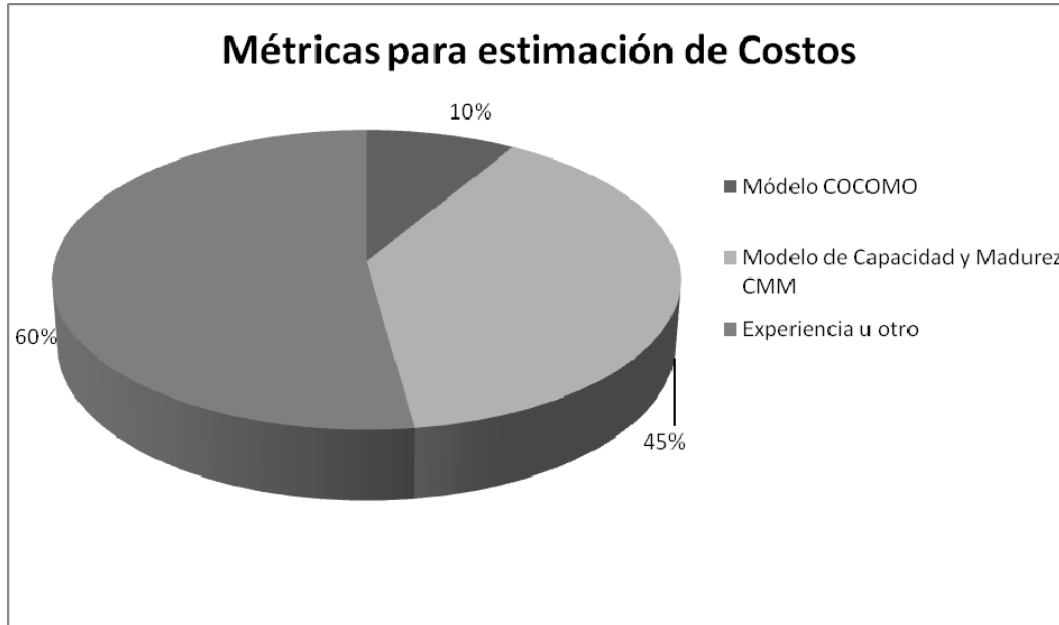
**Figura 20. Gráfico para medir la importancia de la métrica.**



**Fuente: Elaborado por Walter Vásquez**

Se observa, que en las fases de análisis y diseño existe un porcentaje del 73% indicado que es donde las empresas ponen mayor atención al momento hacer mediciones y con un 71% está la determinación de costos y tiempo, lo cual indica que la calidad del producto a entregar no está determinado por el costo ni el tiempo pero si es importante tener presente que se aproxime a las expectativas de de lo que la empresa espera obtener, y con un 70% se encuentra que es importante medir la reutilización de código lo cual permitirá que el tiempo de entrega se reduzca y el proceso de pruebas sea menor debido a que el código ha sido probado con anterioridad de esto se ve que el porcentaje de pruebas es de un 68%.

**Figura 21. Gráfico del uso de métricas conocidas.**

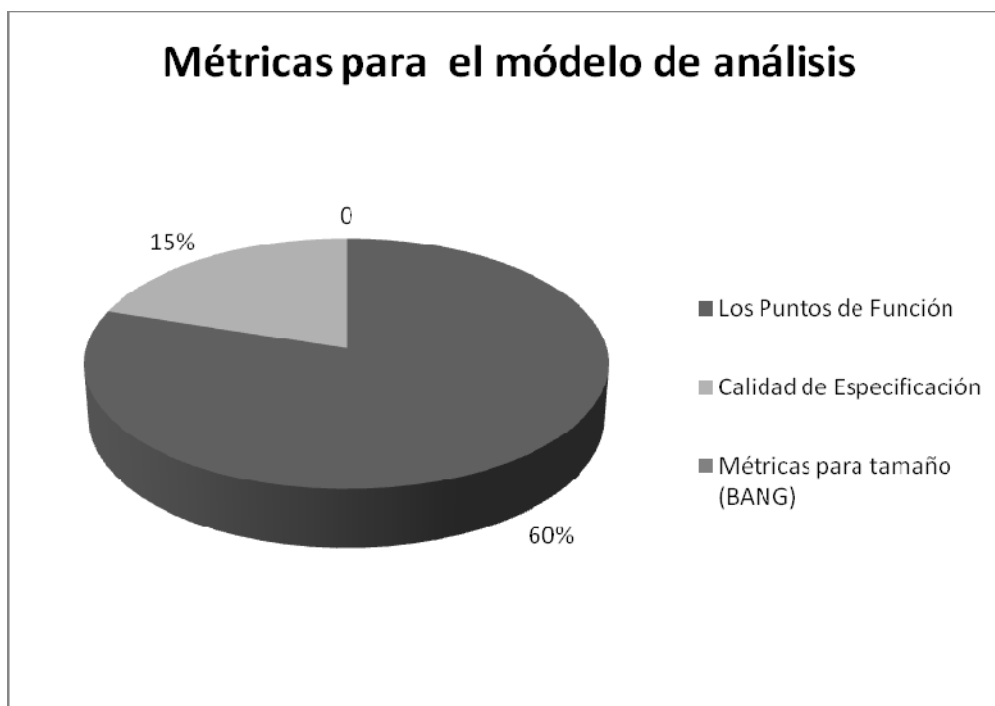


**Fuente: Elaborado por Walter Vásquez**

Como puede observarse en el gráfico, el porcentaje mayor lo tiene la experiencia y uso de otras técnicas, esto indica que las personas hacen uso de su experiencia al hacer el cálculo de los costos o utilizan algún método posiblemente no conocido o implementado directamente por los mismos, y existe mayor conocimiento de la utilización del Modelo de Capacidad y Madurez del software o CMM debido a que se basa en características que para las empresas tienen como parámetro de medición y que lo hace fácil de medir, y con 10% se encuentra el modelo COCOMO que la por ser difícil de aplicar presenta un porcentaje bastante bajo.



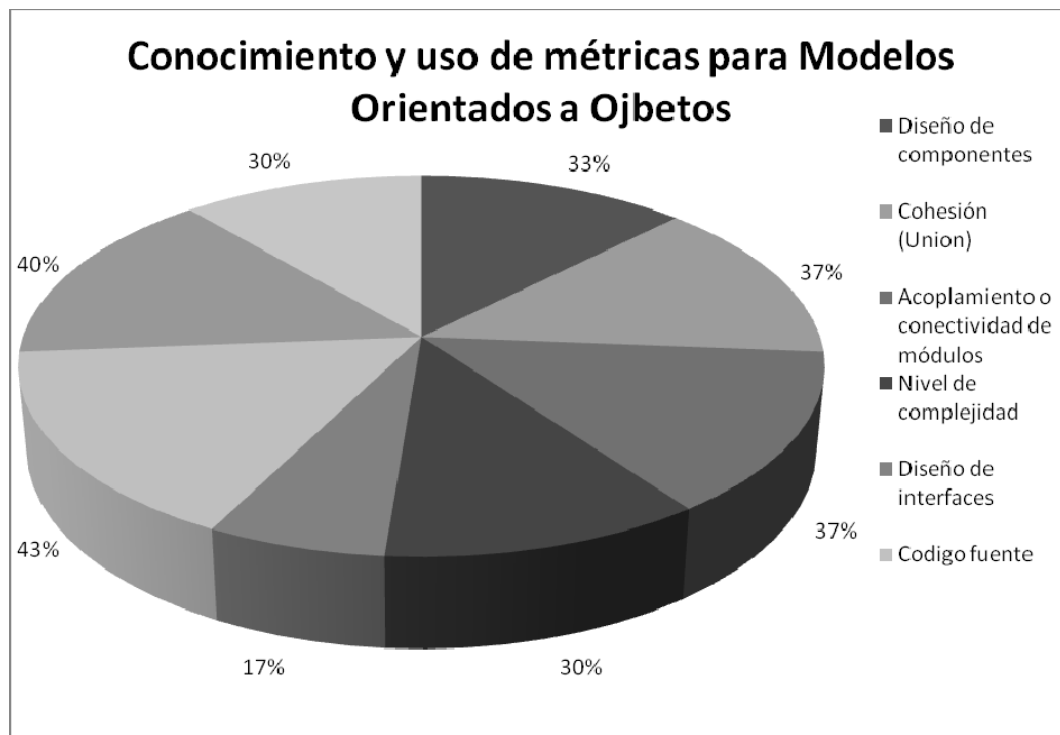
**Figura 22. Gráfico de métricas aplicables al modelo de análisis.**



**Fuente: Elaborado por Walter Vásquez**

Dentro de las métricas utilizadas en el modelo de análisis existen varias dentro de las cuales la conocida como Puntos de Función tiene un 60% de conocimiento lo cual indica que la mayoría de empresas utilizan dicha métrica debido a su fácil comprensión y aplicación, y con un menor porcentaje la Calidad de Especificación y con 0% la métrica Bang.

**Figura 23. Gráfico de conocimiento y uso de métricas en los modelos orientados a objetos.**



**Fuente: Elaborado por Walter Vásquez**

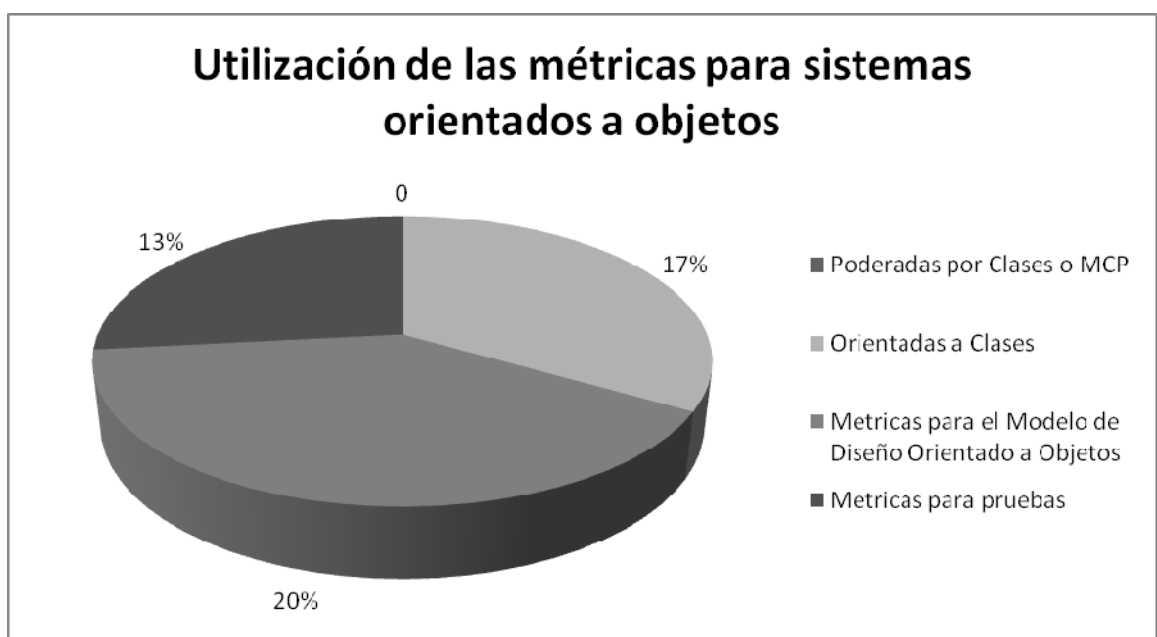
Para modelos orientados a objetos no existe mucho conocimiento por ser muy nuevo, los porcentajes más altos están en un 40% y 43% para las pruebas y código fuente, lo cual indica que se tiene el conocimiento que dicho código puede ser heredado para utilizarlo en nuevos módulos y garantizar a través de las pruebas que eso pueda ser llevado a cabo.

Puede observarse, que la Cohesión y el Acoplamiento tienen un porcentaje del 37%, lo cual indica que se conoce que la interacción y comunicación entre los componentes es tomada como parte vital de los modelos orientados a objetos, y con 30% se encuentran el nivel de complejidad

y pruebas indicando que no se conoce mucho como medir el nivel de complejidad y el mantenimiento.

Y con un 17% se encuentra el Diseño de Interfaces mostrando el poco conocimiento que existe de cómo obtener valores confiables al momento de diseñar interfaces para modelos orientados a objetos.

**Figura 24. Gráfico de uso de métricas para diseño orientados a objetos.**



**Fuente: Elaborado por Walter Vásquez**

Observando la gráfica, lo primero que viene a la vista es que las Métricas ponderadas por clases es la que menor conocimiento presenta con un porcentaje de 0, indicando el poco interés que despierta medir el Software orientado a objetos, el porcentaje más alto se encuentra en las métricas para el modelo de diseño orientado a objetos con un 20%, indicando que todos saben cómo realizar un Diseño Orientado a Objetos aun cuando el porcentaje es bajo, con un tercer lugar se encuentra que la aplicación del Métricas

Orientadas a Clases poseen un 17%, esto indica que en alguna medida se aplica la verificación de las integraciones, herencias y encapsamiento en clases, y con un 13% se encuentra la aplicación de Métricas para pruebas, mostrando que no deja de ser importante la aplicación de pruebas a los Sistemas Orientados a Objetos.

Tomando en cuenta la información obtenida dentro del trabajo de campo, es vital observar que todas las empresas han mostrado tener conocimiento en el uso de técnicas para garantizar la entrega de productos de calidad, de igual manera, se han enfocado en las fases de Análisis y Diseño como elemento principal de garantizar el producto final, todo esto sin dejar por un lado la fase de Pruebas que le garantizaría el cumplimiento de los requerimientos.

## CONCLUSIONES

1. Los conceptos de calidad no deben de ser tomados como aspectos que aplican únicamente en el trabajo, debido a que estos constituyen una forma de vida y comportamiento.
2. Toda administración de un proyecto es un proceso humano, por lo cual no debe dejarse por un lado la aplicación eficiente de los conceptos de planificación, organización, motivación y control.
3. Los modelos y estándares que se utilizan para garantizar la entrega de un producto de calidad, en este caso para la Ingeniería del Software, se centran algunos en el producto terminado con un enfoque en las pruebas y en la aplicación de métodos correctivos y otros, en los procesos de desarrollo de software iniciando las revisiones y correcciones desde las fases iniciales del desarrollo.
4. Existen diferentes factores que influyen en la calidad del proceso de desarrollo de software, tales como el tamaño de la organización, el recurso humano, la tecnología, el conocimiento, el tiempo, el esfuerzo, el recurso económico, la interrelación y el grado de influencia que poseen, todos estos son determinantes en la entrega del producto final.
5. La ingeniería de software es una de las áreas que debe renovarse constantemente, todo esto debido a la evolución que rápidamente lleva la tecnología (Software, Hardware), lo cual implica que un profesional del software debe mantenerse actualizado en dicha materia.

6. Los métodos tratados en este trabajo de graduación no están orientados directamente sobre la utilización de una metodología en especial, ya que cada una está orientada a una fase diferente dentro de la etapa de desarrollo de software, hay algunas directamente ligadas a las fases de análisis, otras a las fases de diseño, sin olvidar las existentes para las fases de desarrollo y pruebas, cuyo objetivo primordial es garantizar un producto final de calidad.
  
7. Al entregar un producto de software el trabajo no termina ahí, la evolución que las empresas generalmente hacen que continuamente se esté evaluando y modificando el software, para lo cual siempre se debe de estar realizando procesos de mejora y que lleva inmerso la aplicación de técnicas que ayuden a que esos cambios funcionen y sean de calidad.
  
8. La existencia de una cultura institucional en las organizaciones es un factor que muy pocas veces puede medirse e incluso hasta subjetivo, pero existe, y dentro del desarrollo de un proyecto, las actitudes, valores y costumbres del personal involucrado son elementos que participan en cada una de las etapas del ciclo de vida del software, por lo cual, la creación de un sistema de información producirá cambios en el comportamiento de los trabajadores de una empresa, es por ello que los involucrados (clientes y desarrolladores) deben de estar preparados para esto.

## RECOMENDACIONES

1. En la implementación de un sistema de control de calidad, la Junta Directiva y Gerencia de cualquier organización deben de estar comprometidas plenamente con el sistema y brindar todo el apoyo para que esto pueda funcionar.
2. La calidad no es solo establecer requisitos o insistir en que las actividades se hagan bien, no se trata de que algo sea funcional, tiene que ver con la forma en que un proyecto se dirige, en las dependencias y las interacciones, e acciones de las personas para conseguir los objetivos establecido y que las actividades se realicen de manera ordenada, enérgica y correcta.
3. Al momento de realizar el reclutamiento de personal debe cuidarse que el candidato tenga los niveles académicos adecuados para el trabajo a realizar, y capacidad de trabajo en equipo que garantice una integración adecuada en el desarrollo y la aplicación capacitaciones que el usuario pueda aplicar para garantizar la calidad del producto que el equipo va a desarrollar.
4. Que las empresas desarrolladoras de software tengan por lo menos una metodología que ayude a medir la calidad del producto a entregar, sin dejar por un lado el presupuesto constante para realizar dichas actividades.

5. Tener como objetivo el certificarse con las normas internacionales ISO para lograr competir dentro de un mercado creciente y tenerlo como una ventaja competitiva en un mundo globalizado.



## BIBLIOGRAFÍA

1. Larios Gutiérrez, Juan José, Hacia un modelo de calidad. Grupo Editorial Iberoamericana, 1989. 160 p.
2. Silva, Roberto Bravo, Calidad Total. UNED, San José Costa Rica, 1998. 340 p.
3. Dan Ciampa, Calidad Total: Guía para su implantación. Addison-Wesley Iberoamericana, E.U.A., 1993. 286 p.
4. Giltow Howard S. y Shelly J. Gitlow., Como manejar la calidad y la productividad con el método Deming: Una guía práctica para mejorar su posición competitiva. Grupo Editorial Norma, Colombia, 1994.280 p.
5. Pressman, Roger, Ingeniería del software: Un enfoque practico. McGraw-Hill, Mexico. 1993. 824 p.
6. Fairley, Richar, Ingeniería de software, McGraw-Hill, México, 1993. 390 p.
7. Lawrence Pfleeger, Shari, Ingeniería de software. Prentice Hall, Brasil, 2002. 759 p.
8. González, Carlos, Normas internacionales de administración de calidad, sistemas de calidad y sistemas ambientales. ISO 9000 QS-9000 ISO 1400, McGraw-Hill.

## Referencias electrónicas

9. <http://sgp.cna.gob.mx/Publico/Diccionarios/Glosario.htm> (febrero 2008)
10. <http://www.gestiopolis.com/recursos/documentos/fulldocs/ger1/ctabud.htm> (marzo 2008)
11. <http://www.aulafacil.com/administracionempresas/Lecc-9.htm> (marzo 2008)
12. <http://www.geocities.com/gehg48/Teorias1.html> (abril 2008)
13. [www.wikipedia.com](http://www.wikipedia.com) (febrero 2008)
14. <http://www.tuobra.unam.mx/publicadas/041229173633-Reingeni.html> (abril 2008)
15. [http://books.google.com.gt/books?hl=es&lr=&id=g\\_nweMjueSkC&oi=fnd&pg=PR20&dq=%22Stoner%22+%22Administraci%C3%B3n%22+&ots=-OnGRH6GC&sig=e7c0G005twPH4zMhrF0XPKOM\\_pQ#PPA7,M1](http://books.google.com.gt/books?hl=es&lr=&id=g_nweMjueSkC&oi=fnd&pg=PR20&dq=%22Stoner%22+%22Administraci%C3%B3n%22+&ots=-OnGRH6GC&sig=e7c0G005twPH4zMhrF0XPKOM_pQ#PPA7,M1) (mayo 2008)
16. [http://sistemas.itlp.edu.mx/tutoriales/procesoadmvo/tema6\\_1.htm](http://sistemas.itlp.edu.mx/tutoriales/procesoadmvo/tema6_1.htm) (junio 2008)
17. [www.wikipedia.com](http://www.wikipedia.com) (junio 2008)
18. <http://www.monografias.com/trabajos14/control/control.shtml> (agosto 2008)
19. [http://es.wikipedia.org/wiki/Modelo\\_de\\_Capacidad\\_y\\_Madurez](http://es.wikipedia.org/wiki/Modelo_de_Capacidad_y_Madurez) (septiembre 2008)

20. <http://www.wordreference.com/definicion/evoluci%F3n> (septiembre 2008)

21. <http://definicion.de/gerencia/> (octubre 2008)