UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

**ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES**
**CAMPUS "ACATLÁN"**

# TÉCNICA DE ASEGURAMIENTO DE CALIDAD ESTADÍSTICA: UNA HERRAMIENTA QUE CONTRIBUYE A MEJORAR LA CALIDAD DEL SOFTWARE. CASO PRÁCTICO EN LA FASE DE MANTENIMIENTO DE UN SOFTWARE.


SEMINARIO TALLER EXTRACURRICULAR

QUE PARA OBTENER EL TÍTULO DE

**LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN**

PRESENTA

**MARÍA DEL PILAR QUIROGA GARCÍA**

ASESOR:
LIC. MARITZA NOVA JUÁREZ

ENERO 2004

***A mis padres**, quienes me dieron la oportunidad de vivir y su ejemplo de tenacidad.*

***A mis hermanos**, quienes siempre confiaron en que lograría todo lo que me propusiera.*

***A mis sobrinos**, para que no olviden que los sueños cuando se desean de verdad se pueden hacer realidad.*

***A todos los profesores**, quienes me enseñaron y apoyaron desde la educación básica hasta llegar a concluir este trabajo.*

***A mi príncipe azul**, quien ha estado, está y estará en mis sueños, esperando que éstos muy pronto se hagan realidad para siempre.*

***A esos angelitos** que están todavía en el cielo y que algún día compartirán esta felicidad.*

# INDICE

# INTRODUCCIÓN

Con base en las experiencias que se han obtenido durante los 12 años laborados en el área de sistemas en diferentes empresas (pequeñas, medianas y grandes), desarrollando software para diferentes áreas (contable-administrativas, inventarios, gubernamentales, ventas) se ha concluido que en México, hoy en día, el desarrollo del software continúa realizándose de forma artesanal, es decir, que cada software tiene las características particulares de quien lo realiza ya que no se siguen estándares y mucho menos una metodología; se desarrolla el software bajo presión debido a la falta de Planificación; la mayor parte del día laboral se ocupa en corregir lo que se hizo mal en el día, la semana, el mes o el año anterior; a veces las fechas de entrega de un software son tan rígidas que para cumplirlas se sacrifica la funcionalidad y la calidad del mismo; y finalmente no existen indicadores objetivos para medir la calidad del software; todas estas situaciones degradan la calidad del software.

En este trabajo se mostrará que en el proceso de desarrollo de software, es posible y muy útil, la definición de medidas que permitan determinar de manera cuantitativa su calidad. La existencia de medidas apropiadas facilita el mejoramiento continuo y nos ayuda a enfocar nuestros esfuerzos en las áreas potenciales de mejora.

Aunque conocidas hace varios años en Estados Unidos, las técnicas y los métodos relacionados con la medición del software han recibido recientemente notoria atención en el mundo empresarial. En México, a menos que alguien se interese en el tema, la medición del software es, en general, desconocida y por consecuencia no practicada, además de que se piensa que solo está reservada para empresas con personal altamente capacitado. Este trabajo tiene como objetivo presentar una de esas técnicas de una forma práctica y mostrar que es aplicable en cualquier empresa.

Para alcanzar dicho objetivo, en este trabajo se mostrará cómo la utilización correcta de la "Técnica de Aseguramiento de Calidad Estadística" proporciona una mejora en la calidad del software. La propuesta es analizar, mediante dicha técnica, las actividades realizadas durante el mantenimiento del software en una empresa, a fin de aplicar medidas correctivas a las que implican mayores recursos. Esto permitirá al personal de mantenimiento dedicarse a otras actividades.

Para este trabajo se eligió la Técnica de Aseguramiento de Calidad Estadística principalmente por las siguientes dos razones:

a) Permite crear un método para mejorar las fases del ciclo de vida que introducen errores en el software.

b) La recolección de datos para aplicar la técnica de Aseguramiento de Calidad Estadística es "sencilla" porque se basa en los defectos o errores que presenta el software.

De los 12 años de experiencia que se tienen en el desarrollo de sistemas, los últimos 3 han sido en el área de Mantenimiento y particularmente es en ésta donde se observa si las fases de análisis, diseño, desarrollo y pruebas fueron realizadas con calidad, ya que en la fase de Mantenimiento el sistema ya está siendo usado por el usuario final y todos los defectos que se le presenten ocasionarán que el personal encargado del mantenimiento tenga que corregirlos lo más pronto posible, ocasionando esto a su vez, más defectos. Esta cadena de defectos puede hacer que el mantenimiento de un software se convierta en una actividad tediosa y desgastante.

La aplicación de la técnica de Aseguramiento de Calidad Estadística en la empresa, para el caso práctico, involucró más de 8 meses de actividades adicionales a las cargas de trabajo diarias. Debido a que la empresa no contaba con datos de los defectos del software fue necesario diseñar un formato para recolectar los datos para la aplicación de la técnica, una vez diseñado el formato se estableció un método para el llenado del mismo, se invirtió tiempo para recolectar los datos de los defectos detectados y finalmente se aplicó la técnica.

Este trabajo está dirigido a todas las personas que ya se dedican a la fabulosa tarea de desarrollar software y a aquellas que tienen conocimientos en Ingeniería de Software y que están interesados en desarrollarlo con calidad. Se pretende hacer consciencia de que un software parchado y lleno de defectos, no solo va a ser más costoso de mantener, sino fundamentalmente es incapaz de proporcionar un nivel de satisfacción a los clientes y usuarios.

El caso práctico sustentado en este trabajo se aplicó a una empresa transnacional, sin embargo, fue necesario ocultar el origen de los datos para proteger información confidencial de la misma.

Este trabajo se estructuró en cuatro capítulos:

En el primer capítulo se introducen los conceptos básicos para definir explícitamente lo que significa "la Calidad del Software" y se describen los factores que tienen impacto determinante en la calidad en el momento de crear un software.

En el segundo se describe la importancia de obtener medidas objetivas del estado de calidad del software, exponiendo los atributos a ser medidos, los modelos de evaluación de calidad conocidos y el proceso que se sigue para obtener estas medidas.

En el tercero se describen las actividades del Aseguramiento de Calidad del Software y en específico sus mecanismos de medición y de recolección de datos,

detallando los pasos para la aplicación de la técnica del Aseguramiento de Calidad Estadística.

Finalmente en el cuarto capítulo se aplica, durante la fase de Mantenimiento de un software, la técnica de Aseguramiento de calidad Estadística para recolectar datos que permitan medir la calidad del mismo.

# CAPÍTULO 1

# LA CALIDAD DEL SOFTWARE

"La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia."
Ruskin, John

En este capítulo se presentan distintos conceptos relacionados con la calidad, empezando por su definición formal y las distintas definiciones que expusieron los clásicos, considerando como tales a aquellos autores que más influencia han tenido en el desarrollo de la calidad.

Hay que tener presente que los significados que en la vida ordinaria se dan a este vocablo son muy diversos, por lo que, para usarlo con precisión, es necesaria una definición formal y por lo tanto se analizará el concepto de calidad según la norma ISO[1] 8402.

## 1.1. Calidad.

Parece que hoy en día hablar de calidad está de moda, es un término que muchas empresas usan en sus anuncios para dar la sensación de que su producto es mejor que otros de la misma especie.

El interés por la calidad tuvo sus inicios en el sector industrial como una estrategia para resolver problemas de variabilidad de productos y sus dificultades de producción interna.

Para definir la calidad de una manera técnica es necesario mencionar un interés por la satisfacción del cliente y por la mejora continua en los procesos, en las personas y, como resultado, en el producto.

El diccionario de Webster's Revised Unabridged[2] nos indica que la calidad es:

1. La condición de existencia de tal o cual característica que lo distingue de otras; carácter; tipo; rango.
2. Lo que hace o ayuda a hacer algo tal y como es; propiedad, característica o atributo que distingue; virtud, capacidad, o poder peculiar; trato distintivo.

Usando como base la definición anterior, podemos entender por qué si se solicita a diferentes personas que definan la palabra calidad probablemente den como respuesta alguno de los siguientes ejemplos: la calidad para un fabricante de zapatos sería "producir zapatos sin defectos", para un estudiante podría ser "hacer las cosas bien", para un padre de familia sería "llevar una vida honorable"; todas estas definiciones están calificando propiedades de algo y nos muestran que el concepto de Calidad no es absoluto.

Si bien en el lenguaje cotidiano pueden permitirse vaguedades en las definiciones y coexistir múltiples acepciones de la palabra "calidad", en el lenguaje técnico se

[1] ISO: siglas empleadas para designar a la Organización Internacional para la Estandarización, en inglés "International Organization for Standardization".
[2] Free on-line dictionary of computing (09-02-2002) IEEE. http://www.hiperdictionary.com/dictionary/IEEE
IEEE: siglas empleadas para designar al "Institute of Electrical and Electronics Engineers".

tiene que ser mucho más preciso. A continuación se expondrán algunas definiciones formales del término "calidad":

**W. Edwards Deming** propuso la idea de calidad como conformidad con requisitos y confiabilidad en el funcionamiento. Para este autor resulta fundamental el compromiso de mejora constante.

**Philip B. Crosby**, otro autor norteamericano, ofrece una orientación de las organizaciones hacia la prevención del error y lograr "hacer las cosas bien a la primera".

**Joseph M. Juran** propone la calidad como la adecuación del producto al uso. Esta definición sugiere determinar quiénes son los clientes, cuáles son sus necesidades, desarrollar los productos o servicios que las satisfagan, evaluar el logro alcanzado, actuar para reducir la diferencia e introducir mejoras hasta donde seamos capaces.

**Kaoru Ishikawa** recomienda como herramientas para la calidad los métodos estadísticos, fomentar la comunicación y no interrumpir la cadena proveedor-cliente.

**Armand V. Feigenbaum**, a quien se debe el término calidad total, tiene como objetivo principal de calidad la satisfacción del cliente.

Todos estos precursores han tenido una influencia directa y notoria en el desarrollo del concepto actual de calidad y en la creación de estrategias y herramientas para implementarla en las empresas.

La definición de calidad que servirá de base para esta trabajo será la que enuncia la norma ISO[3] 8402, que se encuentra en el diccionario en línea de IEEE[4]:

**"Calidad es la totalidad de rasgos y características de una entidad que le confieren la capacidad de satisfacer necesidades implícitas o explícitas".**

Buscando la definición de "entidad" encontramos que es "aquello que se puede describir y considerar individualmente".

Así pues, los productos y los servicios son entidades, pero también lo son las actividades, los procesos, las organizaciones, los sistemas, las personas o cualquier combinación de los anteriores.

Usando la definición de la ISO 8402 se puede hablar de la calidad de un producto, de la calidad de un proceso, o de la calidad de un sistema, lo cual es una novedad

---

[3] ISO: siglas empleadas para designar a la Organización Internacional para la Estandarización, en inglés "International Organization for Standardization".

[4] Free on-line dictionary of computing (09-02-2002) IEEE. http://www.hiperdictionary.com/dictionary/IEEE. IEEE: siglas empleadas para designar al "Institute of Electrical and Electronics Engineers".

respecto a los conceptos tradicionales de calidad, que solamente eran aplicables a productos.

## 1.2. Software de Calidad.

Después de definir el concepto de calidad surge la siguiente pregunta ¿Cómo se aplica éste concepto al desarrollo de Software?.

Para abordar el tema de la calidad del software es necesario explicar que éste es un producto con características muy peculiares, ya que es un elemento lógico y no físico.

A continuación, se explicarán las características del software listadas en [PRES2002] y en [PIAT2000]:

1. *El software se desarrolla, no se fabrica en el sentido clásico del término.*

   Cuando se usa el término "fabricar" generalmente se entiende como la construcción de algo físico por medios mecánicos, es decir, la construcción de un edificio o de un chip; y cuando se usa el vocablo "desarrollar" se percibe como el crecimiento o modificación de algo para alcanzar otro estado. Este crecimiento se ve claramente durante el "desarrollo" del software ya que éste va tomando formas diferentes mientras pasa de la definición inicial del requerimiento hasta convertirse en una serie de pantallas con las que el usuario puede interactuar.

2. *Se trata de un producto lógico, sin existencia física.*

   Cuando se describe al software como una entidad sin existencia física, se entiende que el producto final no es un objeto que el cliente puede llevarse en su bolsillo, es el análisis, diseño y desarrollo de instrucciones para que la computadora realice una serie de tareas.

3. *No se degrada con el uso.*

   Todos los objetos físicos que se usan en la vida diaria como el reloj, el auto, la ropa, etc. se desgastan por el uso. Al ser el software un producto lógico no sufre ese deterioro por muy intenso que sea su uso, por lo que se puede afirmar que en los casos en los que sea necesaria una reparación del software será porque desde su origen ya tenía un defecto que corregir.

   En [PRES2002] se resalta que el software no se degrada con el uso pero que sí sufre un deterioro debido al mantenimiento del mismo, ya que en esta fase se realizan cambios al software que pueden introducir nuevos

defectos, los cuales necesitarán de otros cambios que pueden generar otros defectos, y así sucesivamente. Con base en la experiencia adquirida al laborar en el área de sistemas se tiene que aceptar que este deterioro es real.

4. *El software se entrega conscientemente con defectos.*

Como consumidor no se adquiere un aparato de sonido o un televisor que sea entregado con una lista de errores conocidos, en cambio, cuando se entrega un software a un cliente, éste último ya tiene en mente que el producto que se le está entregando seguramente será reemplazado en un futuro por una versión en la que se corrijan los defectos del que se esta adquiriendo.

5. *La mayoría del software se desarrolla a la medida.*

Este punto se refiere específicamente a la reutilización de componentes de software en programas diferentes y que a la fecha es limitada, ya que la mayoría de las veces se desarrollan a la medida de las necesidades.

Para continuar con la definición de *calidad de software* se necesita tener un concepto formal de lo que se va a entender por software a lo largo de este trabajo.

El diccionario en línea de IEEE[3] define ***software*** como ***"los programas, procedimientos o reglas escritas y la documentación asociada a la operación del sistema computacional"***. Tomando como base esta definición se concluye que el software no sólo es código, es la unión de todas sus fases de desarrollo.

Ahora ya se puede exponer la definición de ***"Calidad de Software"***.

La definición oficial de la calidad del software es la del estándar IEEE Std. 610-1990, que se menciona en [PIAT2000]:

**"Grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario".**

Para entender y asimilar la definición anterior de "Calidad de Software", se necesita explicar primero el término "requisitos especificados": en el área de sistemas estos requisitos son el resultado de la fase del análisis de requerimientos del proceso de desarrollo del software, y pueden ser requisitos funcionales, de seguridad, de interfaz gráfica, etc.

El segundo término a explicar es "necesidades o expectativas del cliente": éstos son los requisitos implícitos que no se obtienen durante el análisis del desarrollo

[3] Free on-line dictionary of computing (09-02-2002) IEEE. http://www.hiperdictionary.com/dictionary/IEEE. IEEE: siglas empleadas para designar al "Institute of Electrical and Electronics Engineers".

del software pero que el usuario desea obtener. Evidentemente, mientras mejor se realice la fase de análisis, menos requisitos de este tipo quedarán sin especificarse explícitamente.

Entonces, se resume que durante este trabajo se entenderá como **"Calidad de Software" el grado en que un sistema cumple con los requisitos implícitos y explícitos del cliente**.

Tomando como base el concepto anterior de Calidad de Software, se explicará el término ***"Software de Calidad"*** y se definirá como **"el software que cumple con los requisitos explícitos e implícitos del cliente en un alto grado y este último puede ser expresado cuantitativamente"**.

La mayoría de los que se dedican a crear software han afirmado alguna vez, de una forma subjetiva y con las mejores intenciones, que el software en el que participaron era de Calidad, se menciona de "forma subjetiva" porque no se tienen bases cuantitativas en las cuales apoyarse para asegurar que realmente el software que realizaron fue de calidad en sus fases de análisis, diseño, desarrollo y pruebas.

## 1.3. Factores que afectan la Calidad del Software.

Cuando se trata de mejorar la calidad de un producto de manufactura se piensa inmediatamente en aumentar la calidad del proceso ya que éste se considera el factor crítico de mejora del primero. Estas ideas de mejora de procesos para aumentar la calidad del producto fueron aportadas por el Ingeniero W. E. Deming [SOMM2002].

Para los productos intangibles, como lo es un Software, que provienen de un proceso que es básicamente intelectual y difícil de automatizar, no es obvio que con la sola mejora del proceso mejore el producto. En [SOMM2002] se listan los cuatro factores principales que afectan la Calidad del Software: (1) *Calidad del personal*, (2) *Calidad del proceso*, (3) *Tecnología de desarrollo* y (4) *Costo y tiempo*, los cuales se explicarán a continuación:

### 1) Calidad del personal.

El personal de una empresa, es el factor más importante para la Calidad del Software, y se enfatiza "el personal de la empresa" porque ésta frase no sólo se refiere al personal del área de sistemas sino también al personal del área de recursos humanos, del área de atención al cliente, del área administrativa, del área de ventas, etc., ¿por qué se involucra a todo el personal de la empresa? porque todos, tarde o temprano, tendrán una participación en el desarrollo del

software, sobretodo, si la empresa genera internamente sus propios sistemas para las diferentes áreas.

En una empresa, en donde el personal del departamento de sistemas tiene calidad pero el personal del área que va a definir y a usar el software no tiene calidad, pueden ocurrir, entre otras, las siguientes situaciones:

a) La fase de análisis de requerimientos puede quedar incompleta debido a que la persona encargada de definir los requisitos explícitos del software no será capaz o no querrá definirlos, tal vez porque tenga miedo de que el software, una vez implementado, pueda reemplazar sus actividades o porque desconozca el objetivo de la creación del mismo; la falta de datos en esta fase inicial afectará a las fases siguientes y puede resultar muy costoso en tiempo, presupuesto y en esfuerzo al ir adecuando la definición original dada hasta llegar a la funcionalidad requerida.

b) El software es implementado en la empresa, y el usuario final introduce "basura", entonces el software procesará y entregará "basura", lo cual, desde el punto de vista de cualquier persona, degrada la calidad del software.

De la misma forma podemos encontrar empresas en las que su personal del departamento de sistemas no tiene calidad, esto dará como resultado desarrollos de software sin calidad, aunque el personal de las demás áreas involucradas desempeñen sus labores con calidad.

Estos son algunos ejemplos que sirven para reconocer la importancia de la calidad del factor humano en el desarrollo de software de calidad y que puede experimentar cualquier persona que labore en el área de sistemas.

**2) Calidad del proceso de software.**

Antes que nada, entiéndase por proceso de software al "marco de trabajo de todas las tareas que se requieren para construir software de calidad" [PRES2002], desde la gestión del proyecto hasta las tareas de aseguramiento de calidad del mismo.

Como se mencionó anteriormente, el proceso del software es básicamente intelectual, de ahí la importancia de que se realice con calidad, ya que depende de las habilidades personales y la experiencia de los involucrados, por ejemplo: si durante la gestión del proyecto la selección del personal para el desarrollo del software es incorrecta, debido a que no se definieron adecuadamente los requerimientos de los puestos, puede darse el caso de que el personal elegido nunca pueda trabajar en equipo y no se alcancen los objetivos, aunque éstos hayan sido bien definidos.

A veces se cree que si la utilización de un proceso dio como resultado un software de calidad, podemos generalizar el uso de este proceso en el desarrollo de varios

proyectos, se recomienda tener cuidado con esta creencia ya que por experiencia se ha aprendido que si el desarrollo es muy grande, el proceso tiene un impacto determinante en la calidad del software porque depende en gran parte de la planificación y la administración del proyecto, pero si el desarrollo es pequeño las habilidades del personal involucrado son más importantes que el proceso empleado.

**3) Tecnología de desarrollo.**

Tecnología de desarrollo son las herramientas tecnológicas que se usan durante las fases del ciclo de vida del desarrollo del software que ayudan a lograr la calidad del mismo.

La tecnología es importante en un área en la que se desarrolla software, pero hay que tener cuidado para no considerarla primordial.

Contar con herramientas tecnológicas adecuadas en el área de desarrollo de sistemas es valioso, de otra forma se puede dar el caso de que la herramienta sea tan obsoleta que ya no pueda competir en tiempos de respuesta, confiabilidad y seguridad con otras herramientas que se encuentran en el mercado.

Pero, por otro lado, si una empresa compra una herramienta sofisticada, pensando que ésta le acortará los tiempos de desarrollo, puede acarrear graves problemas sin pensarlo, por ejemplo: tal vez tenga que cambiar todas sus computadoras personales porque las existentes no tienen la capacidad para soportar la nueva herramienta y además tenga que capacitar a todo su personal para usarla. Todos estos tiempos y costos, si no están planeados, provocarán que la calidad del desarrollo de software empiece a peligrar.

**4) Costo y tiempo.**

Este factor depende totalmente de la fase de planeación, ya que si un proyecto esta mal presupuestado en costo y tiempo ésta será la causa de que se apresure la entrega del mismo, sacrificando la calidad.

Hoy en día, cuando a una empresa se le presenta el proyecto para el desarrollo de un software generalmente se está compitiendo con otra alternativa de solución al problema, ya sea manual u otro proyecto, así que para ganar se dan tiempos de entrega, costos y ventajas operativas muchas veces irreales, esto lo único que genera es que cuando se van llegando a los límites presupuestados lo más afectado es la calidad del software.

La calidad de estos cuatro factores debe ser contemplada cuidadosamente en el momento de la planeación completa del desarrollo del software para garantizar la calidad del producto final.

Hasta este momento se han expuesto los conceptos básicos para entender el término "Calidad de Software", así como los factores que tienen un impacto determinante en la calidad del software durante su desarrollo.

# CAPÍTULO 2

# MEDICIÓN DEL SOFTWARE

"Tu no puedes controlar lo que no puedes medir."
Tom DeMarco

En Marzo de 1994 Norman Fenton [FENT1994] comentaba que a pesar de que hacía ya varios años se había empezado a hablar de la teoría de la medición del software, ésta había sido ignorada hasta ese momento; en Septiembre del 2002 Gopal, Krishnan, Mukhopadhyay y Goldenson [GOPA2002] mencionan que los programas de medición en las empresas de software son un recurso importante para controlar la calidad y el costo en el desarrollo de software y que están siendo considerados como una parte cada vez más importante de la ingeniería de software.

De los comentarios anteriores se concluye que, en los últimos años, el interés y la necesidad de controlar la calidad del software han propiciado la práctica de la medición del software para conocer, controlar y mejorar su calidad.

En este capítulo se mostrará que al aplicar la medición del software en algunos aspectos de su desarrollo nos ayuda a controlar la calidad y el costo del mismo.

## 2.1. Medida/Métrica/Indicadores.

En el lenguaje cotidiano, a veces, los términos medida, medición y métrica son usados indistintamente para expresar lo mismo, así que se dará una breve explicación de cada término para que se entienda la diferencia entre cada uno cuando estos conceptos se aplican al software.

### a) Medida del Software.

Fenton [FENT1994] define ***medida*** como **"el proceso mediante el cual se asignan números o símbolos a atributos de entidades del mundo real"**.

Una *entidad* puede ser un objeto, tal como una persona o una especificación de un software, o un evento, tal como un viaje o la fase de prueba de un proyecto de software.

Un *atributo* es una característica o propiedad de la entidad, tal como la presión sanguínea (de una persona), el costo (de un viaje), o la duración (de la fase de pruebas) de un proyecto de software.

Hay dos grandes tipos de medidas: directa e indirecta.

> La *medida directa* de un atributo es la medida que no depende de la medida de ningún otro atributo, ejemplo, el número de líneas de código.
>
> La *medida indirecta* de un atributo es la medida que involucra la medida de uno o más atributos, ejemplo, calidad y facilidad de mantenimiento.

Los autores que han tenido experiencia en trabajar con medidas del software describen las características que éstas deben de cumplir, las cuales se detallan a continuación:

> Partiendo de la definición de medida de Fenton es claro ver la necesidad de especificar la entidad y el atributo a medir antes de realizar cualquier medición de software, por lo tanto, [FENT1994] propone que las medidas deben **tener objetivos claros**. Se necesita primero conocer exactamente qué entidades del software son sujetas de interés, y después se necesita decidir cuáles atributos de las entidades seleccionadas tienen importancia.
>
> Kitchenham, Hughes y Linkman [KITC2001] enfatizan que además de que la medida debe ser clara debe también ser **confiable**, entendiéndose por esto, que pueda o no ser repetible, es decir, si la medida de un atributo es realizada por otra persona se producirá el mismo resultado; para esto es necesario, además de definir la entidad y el atributo a medir, definir la unidad de medida a ser usada, ejemplo, personas, líneas de código, escala de severidad, etc. También proponen que la medida debe poder ser **comparable**, para esto, es necesario especificar las condiciones bajo las cuales la medida será tomada, nombrándole a esta tarea el protocolo de medida o reglas de conteo, ejemplo, si se desea contar el número de defectos por módulo de un software, es necesario especificar cuando empezar a contar, ya que el conteo de errores[1] en la fase de pruebas no es comparable con el conteo de defectos[2] después de la liberación al usuario.

En la medición de software hay tres clases de entidades de interés:

1) *Proceso*: son todas las actividades de planificación y de garantía de calidad que permiten establecer un plan detallado para el desarrollo del software.

2) *Productos*: es cualquier entregable o documento que surge del proceso.

3) *Recursos*: es todo con lo que se cuenta para realizar el proceso, ejemplo, número de personas, equipo computacional o presupuesto.

Estas entidades tienen diferentes atributos, los internos y los externos.

> Los *atributos internos* de un producto, proceso o recurso son aquellos que pueden ser medidos en términos del producto, proceso o recurso en sí mismo. Por ejemplo, el tamaño en bites es un atributo interno de cualquier documento, mientras que el tiempo empleado en el análisis de un software es un atributo interno de cualquier proceso de software.

---

[1] El término de error representa un problema de calidad que es descubierto antes de entregar el software al usuario final.

[2] Dentro del contexto del proceso de software, el término de defecto implica un problema de calidad que es descubierto después de entregar el software a los usuarios finales.

Los *atributos externos* de un producto, proceso o recurso son aquellos que pueden ser medidos únicamente con respecto a cómo el producto, el proceso o el recurso se relaciona a otras entidades de su ambiente. Por ejemplo, la fiabilidad de un programa (un atributo del producto) depende no sólo del programa, sino también del compilador, de la máquina y del usuario. La productividad es un atributo externo de un recurso, nombrado gente (ya sea un individuo o un grupo de personas); es claramente dependiente de muchos aspectos del proceso y de la calidad de productos entregados.

Generalmente, cuando se desarrolla un software, lo que se desea es medir los atributos externos. Desafortunadamente, sólo pueden ser medidos a través de los atributos internos. Por ejemplo, la productividad del personal es comúnmente medida como una razón entre el tamaño de código entregado (un atributo interno del producto) y el esfuerzo (un atributo interno del proceso).

La figura 2.1 muestra algunos atributos de calidad externos de interés y los atributos internos que pueden medirse y estar relacionados con los atributos externos.

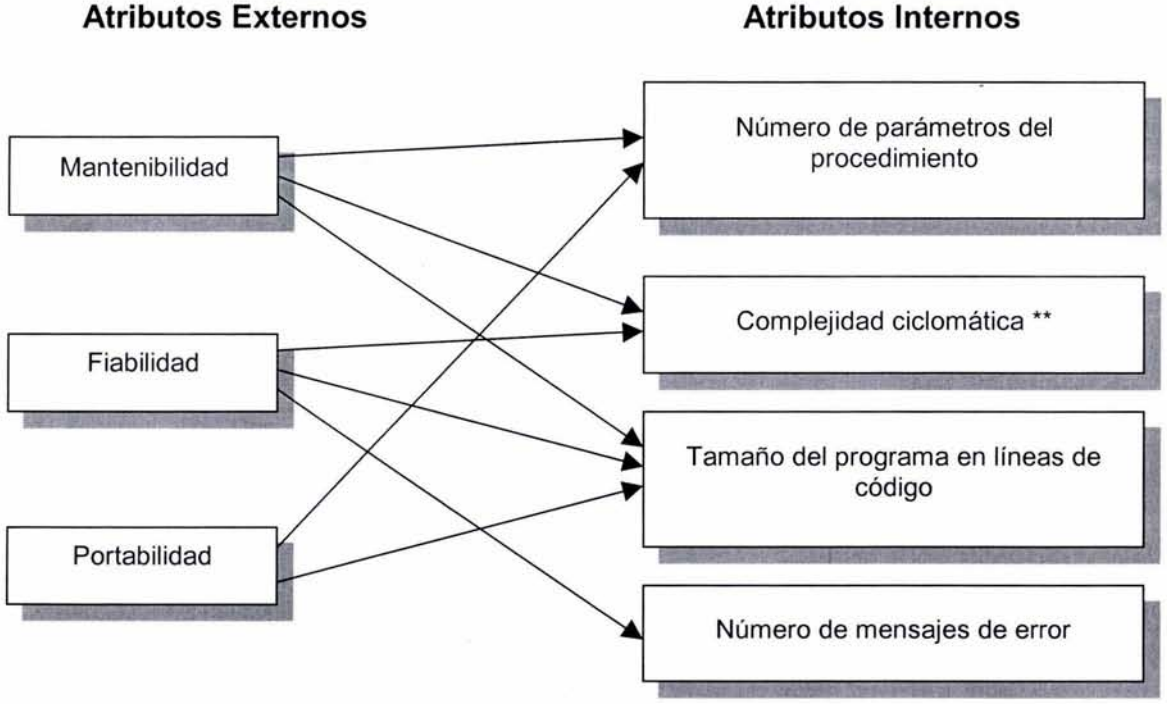



**Figura 2.1. Relaciones entre los atributos de software internos y externos.**

** Complejidad ciclomática es una medida de la complejidad del control de un programa y está relacionada con la comprensión del programa.

**b) Medición del software.**

La ***medición del software*** vamos a entenderla en este trabajo como **el acto de determinar una medida**.

**c) Métricas del Software.**

Según el diccionario de la computación en línea[3] una *métrica de software* es "una medida de la calidad del software, la cual indica la complejidad, la comprensión, la facilidad de prueba, la descripción y complejidad de código".

Esta definición esta muy enfocada a la calidad del código de un software. Buscando otra definición más general del concepto métrica se encontró la siguiente que es la que se usará en este trabajo:

En [GOPA2002] se define ***métrica de software*** como **"la medida cuantitativa del grado en que un software, producto o proyecto posee un atributo dado"**.

Las métricas pueden realizarse sobre el producto y/o sobre el proceso [SOMM2002].

Las **métricas del producto** se dividen en dos clases: dinámicas y estáticas.

> Las *métricas dinámicas* son las recolectadas por las mediciones hechas en un programa en ejecución. Éstas métricas ayudan a valorar la eficiencia y fiabilidad de un programa. Ejemplos relacionados a la eficiencia: tiempo de ejecución requerido por funciones particulares y tiempo estimado requerido para iniciar un software. Ejemplo relacionado con la fiabilidad: número y tipo de caídas del sistema.
>
> Las *métricas estáticas* son las recolectadas por las mediciones hechas en las representaciones del sistema como el diseño, el programa o la documentación. Éstas métricas ayudan a valorar la complejidad, la comprensión y la mantenibilidad de un software. Ejemplo: la medida del tamaño del programa (longitud de código). Generalmente, entre más grande sea el tamaño del código de un componente de un programa, más complejo y susceptible a errores será el componente.

Las **métricas del proceso** se utilizan para evaluar si la eficiencia de un proceso ha mejorado. Por ejemplo: se puede medir el esfuerzo y tiempo dedicados a las pruebas. Las mejoras efectivas para los procesos de prueba reducen el esfuerzo, el tiempo de prueba o ambos.

---

[3] Free on-line dictionary of computing (09-02-2002) IEEE. http://www.hiperdictionary.com/dictionary/IEEE. IEEE: siglas empleadas para designar al "Institute of Electrical and Electronics Engineers".

Se distinguen tres clases de métricas del proceso:

1. El tiempo requerido para completar un proceso en particular. Éste es el tiempo total dedicado al proceso, el tiempo de calendario, etc.

2. Los recursos requeridos para un proceso en particular. Los recursos pueden ser el esfuerzo total en personas-días, los costos de viaje, los recursos de cómputo, etc.

3. El número de ocurrencias de un evento en particular. Ejemplos: número de defectos descubiertos durante la inspección del código, el número de peticiones de cambios en los requerimientos, etc.

Los primeros dos tipos de métricas se utilizan para ayudar a descubrir si los cambios en el proceso mejoran su eficiencia. La tercera métrica tiene una influencia directa en la calidad del software.

**d) Indicadores de Software.**

Un ***indicador*** es **"una métrica o una combinación de métricas que proporcionan una visión profunda del proceso, del proyecto o del producto de software"**.

## 2.2. La importancia de medir.

¿Por qué es importante medir? La respuesta es porque medir es la forma de determinar si se está mejorando o no. Si en el momento en el que se desarrolla software no se conoce ciertamente donde se está, entonces en el futuro no se sabrá tampoco y ese desconocimiento puede llevar al fracaso sin que uno lo sepa ni lo pueda controlar.

Los argumentos para aplicar la medición al desarrollo del software son los siguientes:

- Establecimiento de una línea base del proceso desde donde se evaluarán las mejoras. En una empresa en la que apenas se está iniciando un programa de medición, la primera medición que se realice servirá de base para la estimación de ciertos aspectos del desarrollo, tales como, los módulos del sistema más propensos al error o el número de errores que se deben esperar cuando se inicien las pruebas.

- Evaluar el estado actual del desarrollo con respecto a la planeación original, de tal manera que podamos controlar las posibles desviaciones del proyecto o del proceso.

- Mejorar la calidad del proceso o del producto debido a la recolección de información cuantitativa que nos ayuda a identificar áreas con ineficiencias.

- Permite un mejor entendimiento y manejo del proceso del software y esto hace posible hacer los cambios necesarios para mejorar la productividad y la calidad. Estas mejoras deben resultar en un incremento de la productividad y de la calidad y una reducción de costos.

Algunos de los aspectos del desarrollo del software para los cuales la medición es útil son:

- ***Programa***. La medición en este aspecto ayuda a saber si el desarrollo del software esta en tiempo. Ejemplo: medir el estado de requerimientos del usuario y el estado de los cambios requeridos sirve para evaluar si se está dentro de los tiempos planeados.

- ***Costos***. La medición ayuda a saber si se tienen recursos financieros para finalizar el software. Ejemplo: medir la experiencia del personal sirve para saber si necesitará tomar cursos de capacitación.

- ***Calidad***. Medir ayuda a saber si el software fue bien hecho. Ejemplo: medir los reportes de los problemas sirve para conocer las áreas con deficiencias que necesitan un plan de corrección.

- ***Crecimiento***. La medición ayuda a saber si será fácil escalar el software a otras plataformas. Ejemplo: medir las líneas de código sirve para conocer el tamaño del software y evaluar su estabilidad en el momento del escalamiento.

- ***Habilidades***. Medir sirve para saber qué tan hábiles somos para desarrollar software. Ejemplo: medir las líneas de código y el esfuerzo aplicado en un software sirve para evaluar la productividad.

Estos aspectos interactúan entre sí, por ejemplo, las habilidades personales influyen en el costo y la planeación de horarios influye en la calidad.

Después de conocer las ventajas de aplicar la medición del software surge la pregunta ¿por qué aún no se tiene la cultura de aplicar la medición del software?. En [BRIA2002] se describen algunas de las razones por las que se dificulta la medición cuantitativa durante el desarrollo de software:

- La tarea de la medición es dura ya que involucra labores adicionales a las cargas de trabajo normales de un desarrollador y las organizaciones aún no tienen establecido un programa de métricas de software.

- El objetivo de la medición del software no está siempre derivado de un objetivo empresarial que se trate de alcanzar por lo que con el tiempo la medición pierde importancia para la empresa.

- Aún si el objetivo de la medición del software está bien definido y nació como parte de un objetivo de la empresa, a veces los resultados esperados de la medición no están bien definidos y se pierde interés en obtenerlos.

- A veces las medidas que se obtienen tratan de ser aplicadas en ambientes diferentes de las que fueron obtenidas, ejemplo, tratar de aplicar una medida que fue definida para un software no orientado a objetos en un software orientado a objetos.

Estas características no denotan que la medición del software no sea útil sino que para aplicarla se debe cuidar aspectos muy importantes. Además, se debe tomar en cuenta que en una empresa en donde no se tiene ni idea de por qué los proyectos de software están siempre atrasados ni de por qué se emplea mucho esfuerzo en tareas repetitivas y en corrección de defectos, siempre será útil el empezar por conocer el estado actual a través de la medición para saber cuáles actividades ocasionan esos atrasos y poderlas mejorar.

## 2.3. Modelos de evaluación de la calidad del software.

Como se vio anteriormente, hay varios aspectos del desarrollo del software que son factibles de medir, este apartado se enfocará a uno de ellos: *la calidad del software*.

Cuando se trata de evaluar la calidad de un producto de software hay que tener en cuenta que la calidad es un concepto muy complejo[4] y que, además, depende mucho del punto de vista que se adopte. Esto representa una dificultad para la evaluación formal de la calidad del software.

Para poder evaluar la calidad del software fácilmente es necesario descomponer el concepto de calidad en propiedades más sencillas de medir y evaluar. Este tipo de descomposición recibe el nombre de *modelo de calidad.* A continuación se describirán algunos modelos con los que se evalúa la calidad del software.

### a) Modelo de calidad de McCall.

Hace 26 años, McCall definió un modelo de calidad, el cual da inicio a los primeros pasos para el desarrollo de métricas de la calidad del software. El modelo de McCall se presenta en la figura 2.2.

[4] La definición de la calidad se dio en el capítulo 1

El modelo de McCall se basa en descomponer el concepto de calidad en tres aspectos importantes del producto de software, desde el punto de **vista del usuario**: su capacidad de operación, su capacidad para ser modificado y su capacidad de adaptación a otros entornos.

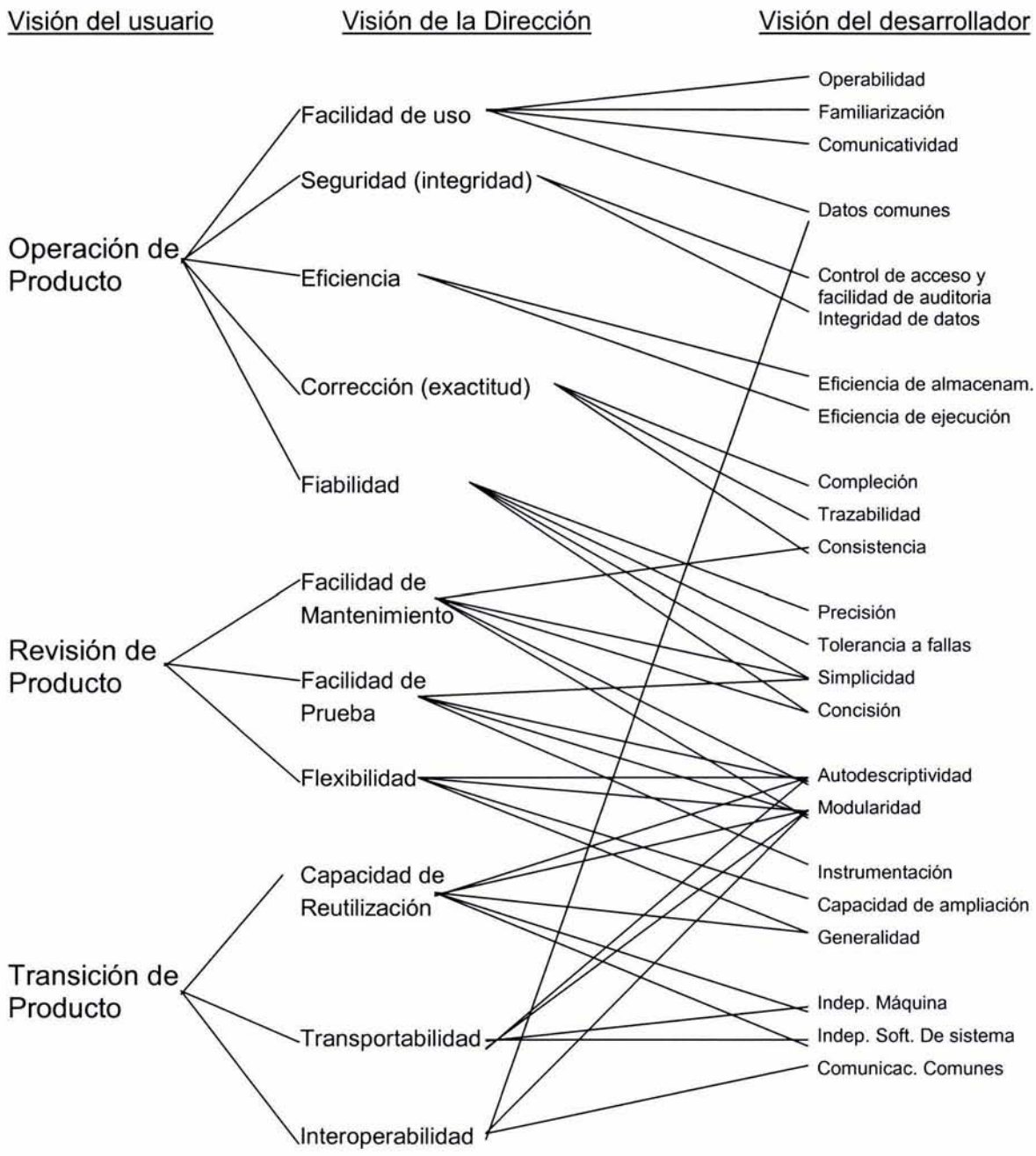



**Figura 2.2. El modelo de McCall.**

Cada aspecto se descompone en una serie de factores que determinan la calidad de cada aspecto y estos factores se evalúan más fácilmente que los tres aspectos antes mencionados y permiten tener una visión más apropiada de la calidad (desde el punto de **vista de la Dirección**). A continuación se describen las definiciones de los distintos factores [PIAT2000]:

- ***Facilidad de uso***: es el grado de esfuerzo necesario para aprender a utilizar el software, preparar la entrada de datos e interpretar la salida del mismo.

- ***Integridad***: es el grado en el que se puede controlar el acceso del personal al software o a los datos que utiliza.

- ***Eficiencia***: son las necesidades de recursos hardware y software requeridos por el software evaluado para realizar sus funciones.

- ***Corrección***: es el grado en el que el software cumple sus especificaciones.

- ***Fiabilidad***: es el grado o probabilidad de que el software no falle al realizar sus funciones.

- ***Facilidad de mantenimiento***: es la facilidad o grado de esfuerzo para mantener operativo el software mediante la corrección de los problemas.

- ***Facilidad de prueba***: es el esfuerzo necesario para probar el software de modo que se tenga un cierto grado de confianza en que realiza adecuadamente sus funciones.

- ***Flexibilidad***: es la facilidad o grado de esfuerzo necesario para modificar el software en funcionamiento.

- ***Capacidad de reutilización***: es la capacidad o grado de esfuerzo para que el software o alguna de sus partes puedan ser utilizadas en otros desarrollos de software.

- ***Transportabilidad***: es la facilidad o grado de esfuerzo necesario para migrar el software de un entorno de operación a otro.

- ***Interoperabilidad***: es la capacidad o grado de esfuerzo necesario para que el software pueda operar conjuntamente con otras aplicaciones de software.

Medir estos factores es difícil, y en algunos casos imposible, así que estos factores se descomponen en una serie de **métricas** que determinan su calidad. Estas métricas son, en muchos casos, propiedades internas del software que no dependen en su apreciación de quién esté observándolas y que **los desarrolladores** de software consideran que influyen en la calidad. Usando estas

métricas se desarrollan expresiones para los factores, de acuerdo con la siguiente relación:

$$F_q = c_1 X m_1 + c_2 X m_2 + .... + c_n X m_n$$

Donde $F_q$ es un factor de calidad de software, $c_n$ son coeficientes de regresión y $m_n$ son las métricas que afectan al factor de la calidad. Desgraciadamente, muchas de las métricas definidas por McCall pueden medirse solamente de manera subjetiva. Las métricas pueden ir en forma de lista de comprobación que se emplea para puntuar atributos específicos del software. El esquema de puntuación propuesto por McCall es una escala del 0 (bajo) al 10 (alto). A continuación se describen las definiciones de las diferentes métricas [PIAT2000]:

- ***Operabilidad***: Propiedades del software que determinan la facilidad de las operaciones y los procedimientos relativos a la explotación del software.

- ***Familiarización***: Propiedades del software que proporcionan al usuario información de operaciones reales o que facilitan la familiarización inicial con el producto.

- ***Comunicatividad***: Propiedades del software que proporcionan eficacia y facilidad en las comunicaciones.

- ***Normalización de datos (Datos comunes)***: Propiedades del software que determinan la posibilidad de utilización común de datos con otros sistemas.

- ***Control de Acceso (seguridad)***: Propiedades del software que proporcionan facilidades para el control de accesos al software y a los datos que maneja.

- ***Facilidad de auditoría***: Propiedades del software que proporcionan facilidades para realizar auditoría del software, de los datos empleados o de los resultados obtenidos.

- ***Integridad de datos***: El empleo de estructuras y tipos de datos estándares a lo largo del programa.

- ***Eficiencia de almacenamiento***: Propiedades del Software que proporcionan unas necesidades mínimas de memoria para su operación.

- ***Eficiencia de ejecución***: Propiedades del software que proporcionan un consumo mínimo de recursos de procesamiento al realizar sus operaciones.

- ***Compleción***: Propiedades del software que proporcionan la implantación total de todas las funciones requeridas.

- ***Trazabilidad***: Propiedades del software que proporcionan una traza o pista reconocible desde los requisitos hasta su implantación en relación a un desarrollo específico y a un determinado entorno de operaciones.

- ***Consistencia***: Propiedades del software que proporcionan técnicas y documentación uniforme y coherente a las distintas etapas del software.

- ***Precisión***: Propiedades del software que proporcionan el grado de precisión requerido para los resultados que hay que obtener.

- ***Tolerancia a fallas***: Propiedades del software que proporcionan la continuidad del funcionamiento bajo condiciones no habituales.

- ***Simplicidad***: Propiedades del software que proporcionan la implantación de funciones de la manera más comprensible posible (evitando que aumente la complejidad).

- ***Concisión***: Lo compacto que es el programa en términos de líneas de código.

- ***Autodescriptividad***: Propiedades del software que proporcionan explicaciones sobre el desarrollo de cada función.

- ***Modularidad***: Propiedades del software que proporcionan una estructura de módulos adecuadamente independientes.

- ***Instrumentación***: Propiedades del software que proporcionan la posibilidad de observar el comportamiento del software durante su ejecución (por ejemplo, durante el proceso de ejecución de las pruebas).

- ***Capacidad de Ampliación***: Propiedades del software que proporcionan facilidad de añadir nuevas capacidades funcionales o datos al sistema.

- ***Generalidad***: Propiedades del software que proporcionan amplitud a las funciones realizadas.

- ***Independencia de la máquina (del hardware)***: Propiedades del software que determinan su independencia de su entorno físico de trabajo.

- ***Independencia entre sistema y software***: Propiedades del software que determinan su independencia de su entorno lógico de trabajo.

- ***Normalización de comunicaciones (Comunicaciones comunes)***: Propiedades del software que favorecen una fácil intercomunicación del sistema con otros.

Hay que mencionar que algunas métricas influyen positivamente en otras (por ejemplo, la facilidad de prueba sobre la facilidad de mantenimiento), mientras que otros interactúan en contraposición (por ejemplo, la eficiencia frente a la portabilidad de la aplicación). Lo anterior quiere decir que no siempre se puede conseguir el máximo nivel en todas las métricas, en general se logra llegar a un equilibro más apropiado para la aplicación.

En [PRES2002] se comenta que los factores de calidad de McCall son, hoy en día, tan válidos como cuando se propusieron los primeros en los años 70.

**b) Otros modelos de evaluación de la calidad.**

En los años ochenta se impulsó la creación de modelos de calidad particulares para cada empresa o para cada proyecto, en vez de tratar de aplicar el mismo en todos los casos como se venía haciendo. A continuación se describirán algunos modelos que nacieron con este enfoque.

**b1) Modelo de calidad de GILB.**

Uno de los modelos resultantes de esta generación es el de GILB (1988). Este modelo propone la creación de una especificación de requisitos de calidad para cada proyecto que deben ser descritos conjuntamente por el usuario (cliente) y el analista. Aunque estos requisitos pueden ser tan originales como el proyecto lo requiera (por ejemplo, la facilidad de comercialización) lo más normal es que se tomen de los modelos tradicionales (ejemplo, el de McCall). Una vez que se determina la lista de requisitos, para cada uno de ellos se especifican los siguientes conceptos:

- *Nombre y definición del requisito*, por ejemplo, sòftware de fácil aprendizaje.
- *Escala o unidades de medición,* por ejemplo, tiempo en horas para que un usuario pueda operar solo la aplicación.
- *Recolección de datos o prueba*, por ejemplo, tomar el tiempo de aprendizaje de una muestra de 20 usuarios.
- *El peor valor aceptable,* ejemplo, tres horas.
- *El valor previsto*, ejemplo, 1 hora y media,
- *El valor óptimo*, ejemplo, cercano a 50 minutos.

### b2) El modelo GQM (Goal-Question-Metric).

Este modelo fue propuesto por Basili y Rombach (1988). La idea de este modelo esta descrita en [PIAT2000] y consiste en que toda actividad de medición debe estar precedida por la identificación de un objetivo a lograr (ejemplo, evaluar la calidad del software), lo que lleva a plantearse una serie de preguntas (ejemplo, ¿cómo influye la fiabilidad en la calidad del software?). Para responder las preguntas planteadas es necesario disponer de una serie de métricas (ejemplo, tiempo promedio de funcionamiento adecuado entre cada falla de la aplicación). El uso del modelo de GQM, aunque inicialmente parece poco práctico, es importante ya que todo el análisis que implica permite definir el concepto de calidad para el proyecto, cuando no se tiene este concepto es difícil saber por dónde empezar a mejorar. El modelo de GQM se presenta ejemplificado en la figura 2.3.

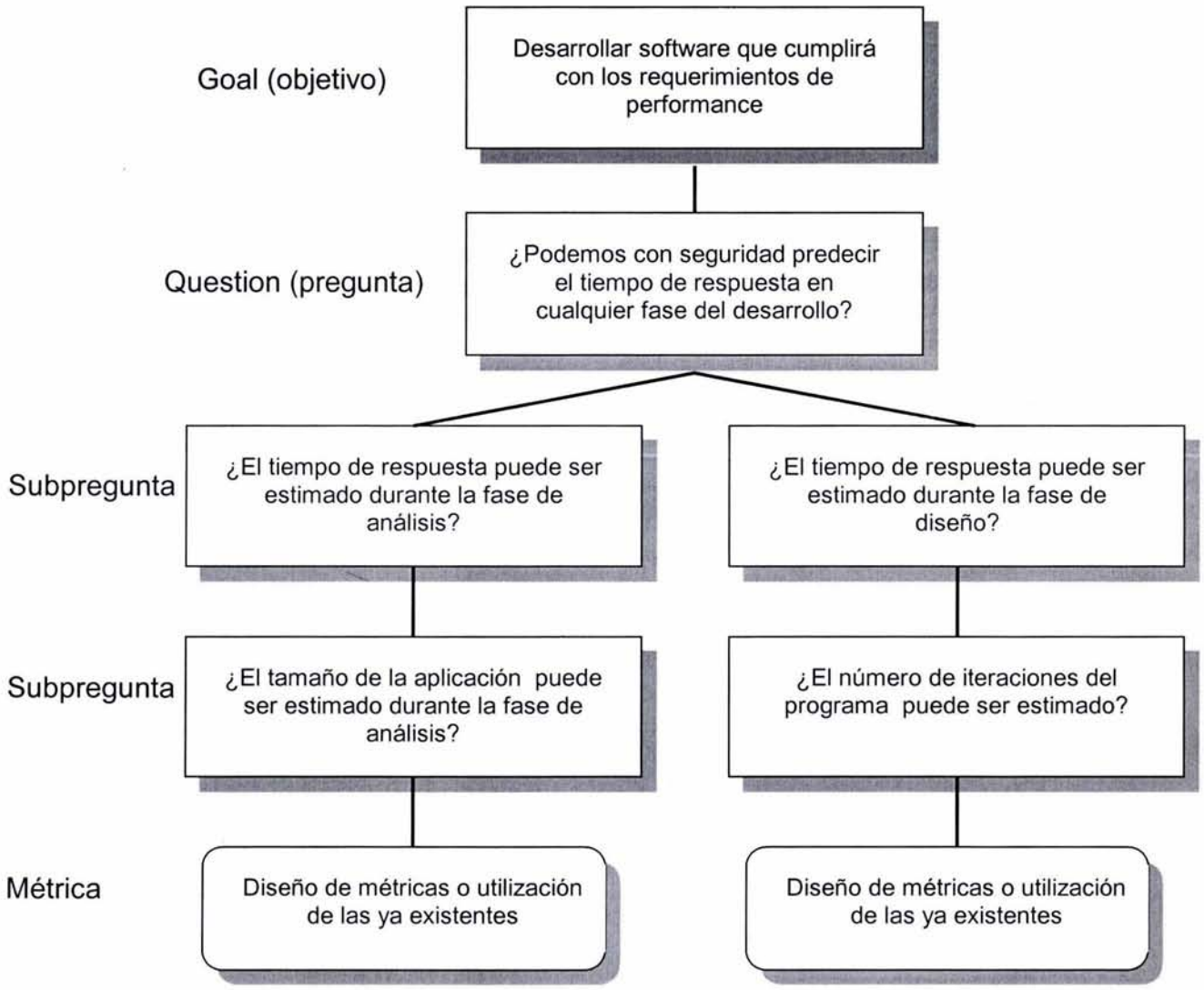



**Figura 2.3. El modelo de GQM.**

**b3) Modelos relacionados con la Eliminación de Defectos.**

Estos modelos de calidad tienen como base considerar que la calidad de un software es equivalente a la ausencia de defectos o errores[5], es decir, mientras menos defectos tenga un software será de mayor calidad. Este enfoque tiene la ventaja de que en una empresa en la que se desea empezar un programa de control de calidad de software es relativamente sencillo la recolección de defectos y errores, ya que estos se pueden obtener de la generación de registros de los errores detectados durante el desarrollo y las pruebas o de las peticiones de correcciones de los defectos detectados por parte de los usuarios una vez que se ha entregado el software.

Este tipo de modelo se emplea para filtrar los errores antes de que se conviertan en defectos, este filtro puede ser realizado de una manera global midiendo el total de errores y el total de defectos de un software o puede irse realizando en cada etapa del desarrollo del software, es decir, mientras más errores sean detectados durante la etapa de análisis de requerimientos, menos errores llegarán a la siguiente fase de diseño (defectos que se pueden detectar o no), y así sucesivamente durante todas las fases del desarrollo, de tal manera que a la hora de ser entregado al usuario el número de errores no detectados sea el mínimo.

El uso de este tipo de modelos nos permite conocer en qué fases del desarrollo el equipo de trabajo esta fuerte y en qué fases está débil. El tener conocimiento del estado actual del proyecto nos permitirá conocer el estado al que deseamos llegar en un futuro.

## 2.4. El proceso de medición.

En los puntos anteriores de este capítulo se dieron a conocer los conceptos y las características de las medidas, de las métricas y de los indicadores de software, se describieron también las ventajas de realizar una medición de ciertos aspectos del software, ahora es importante responder la siguiente pregunta ¿cómo se realiza la medición del software?. En [PRES2002] se describen las cinco actividades del proceso de medición:

- ***Formulación***: en esta actividad se debe definir el objetivo, es decir, se necesita tener claro si se desea medir para evaluar o para predecir, después se necesita definir exactamente qué entidades son sujetas de interés para de esta manera saber qué atributos de éstas entidades son importantes. En esta

[5] El término de error representa un problema de calidad que es descubierto antes de entregar el software al usuario final, mientras que defecto implica un problema de calidad que es descubierto después de entregar el software a los usuarios finales.

actividad se deben formular las preguntas que la medición intenta responder y definir las medidas requeridas para resolver estas preguntas.

- ***Recolección***: esta actividad se refiere al mecanismo empleado para acumular datos necesarios para obtener las medidas. El mecanismo empleado puede ser creado de forma especial o puede estar incorporado en herramientas CASE.

- ***Análisis***: esta actividad se refiere al cálculo de las métricas con base a los datos recolectados. Esta fase debe realizarse cuidadosamente para comprender lo que realmente significan las medidas obtenidas ya que pueden ser mal interpretadas fácilmente y debido a esto se pueden realizar inferencias incorrectas.

- ***Interpretación***: las métricas obtenidas se evalúan para tratar de identificar valores anómalos y decidir si los valores de la métrica indican que la calidad de la entidad elegida está en peligro.

- ***Retroalimentación***: recomendaciones obtenidas de las interpretaciones de las métricas transmitidas al equipo que forma parte del desarrollo del software. De nada serviría conocer el estado de la calidad del software si este conocimiento no se da a conocer a todo el personal involucrado en el equipo.

## 2.5. Recolección de datos. El inicio de la mejora.

Como se mencionó en la sección 2.2., uno de los argumentos para aplicar la medición al desarrollo del software es el establecimiento de una línea base de métricas, de la que se partirá para evaluar si el software mejora o no. Las líneas base de métricas constan de datos recolectados de proyectos de software y pueden ser tan simples como una tabla que contenga los datos por proyecto de líneas de código desarrolladas, horas-hombre empleadas y el costo del proyecto; o pueden ser tan complejas como una base de datos con una gran cantidad de medidas almacenadas de varios proyectos. Las métricas de esta línea base pueden ser usadas varias veces en diferentes proyectos. En [PRES2002] se describe el proceso para la recopilación de métricas de software, el cual se resume en la figura 2.4.

De la figura 2.4 notamos que es precisamente la recolección de datos lo que hace posible el cálculo de métricas, este paso de recolección, el cual es indudablemente el más difícil, es el que determina la gama de métricas que se podrán obtener, ya que dependiendo de qué tan amplia sea la recolección de datos ésta permitirá determinar métricas de calidad, métricas de las fases del proceso de desarrollo, etc.

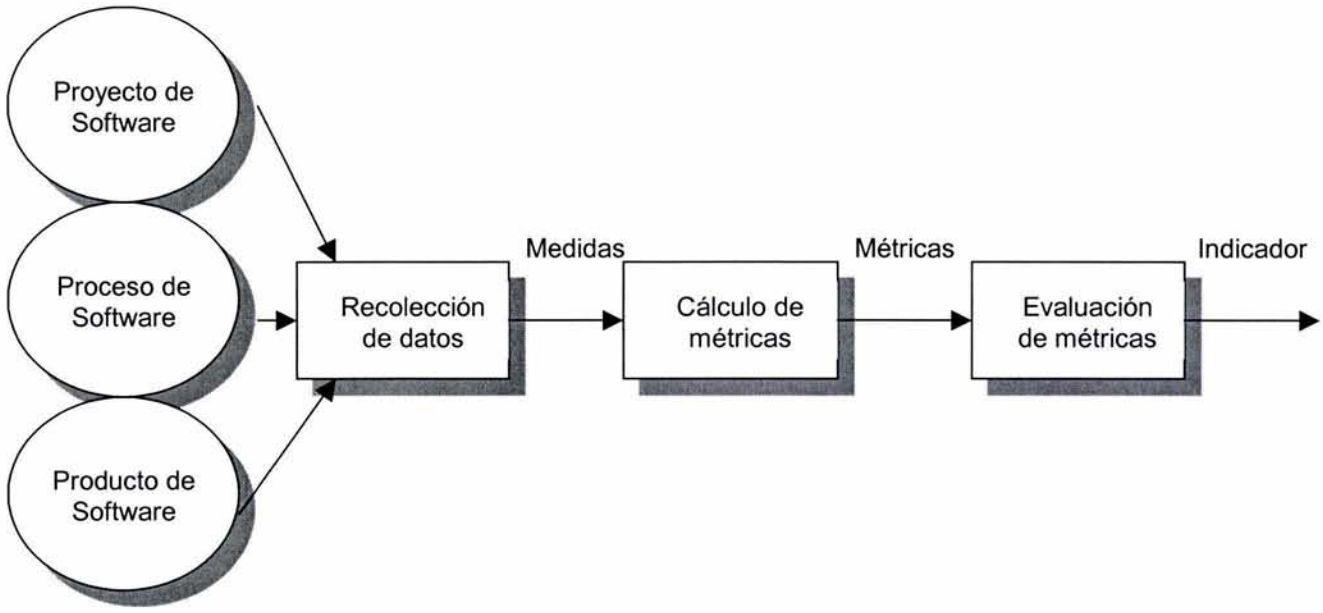



**Figura 2.4. Proceso de recopilación de métricas de software.**

Como se ha visto a través de este capítulo, así como en las empresas se miden las ventas realizadas, los gastos, las ganancias, etc., también, hoy en día, ya empieza a crecer una cultura de medición del software, en la cual ya se tienen metodologías, modelos y herramientas que permiten obtener medidas objetivas del estado de calidad del software. También es importante tener presentes las ventajas que se obtienen al realizar una medición durante el desarrollo del software.

# CAPÍTULO 3

# EL ASEGURAMIENTO DE CALIDAD DE SOFTWARE (SQA) COMO HERRAMIENTA DE MEDICIÓN

"La perseverancia siempre vencerá al talento
y a la apariencia."
Aaron Brown

En este capítulo se verá en qué momento del proceso del desarrollo de software deben realizarse las actividades para asegurar su calidad (Aseguramiento de Calidad del Software) y cuáles son éstas actividades.

No se puede hablar de Aseguramiento de Calidad de Software sin antes definir lo que es el control de calidad, la garantía de calidad y costo de calidad, los cuales serán definidos en este capítulo.

Finalmente, se explicará la Técnica de Aseguramiento de Calidad Estadística y los pasos para aplicarla.

## 3.1. Control de Calidad.

En el capítulo 1 se tomó como concepto de *Calidad* el que la define como "la totalidad de rasgos y características de una entidad que le confieren la capacidad de satisfacer necesidades implícitas o explícitas", también se dio la definición de *Calidad de Software* que la describe como el "grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario". En este subcapítulo toca el turno al concepto *Control de calidad* y su aplicación al desarrollo del software.

Para hablar del origen del vocablo Control de calidad se hará referencia a [GES2003], donde se explica que el control de calidad fue resultado de la evolución del concepto de calidad, que hasta los años 30, se concentraba en evitar que se produjeran fallos durante la fabricación y que evolucionó para convertirse en lo que se conoce hoy como *Control de Calidad* cuya principal actividad es centrarse en inspeccionar el producto y separar aquel que es aceptable, respecto a su especificación, del que no lo es, lo cual se obtiene realizando controles en las fases de producción que se desean controlar.

También en [GES2003] encontramos las características y desventajas del control de calidad, las cuales se resumirán a continuación.

**Principales características del control de calidad:**

- La responsabilidad de la calidad la tiene el departamento de calidad de las empresas.
- El control de calidad surge cuando se detecta un error en la producción y se actúa para corregirlo.
- El control de la calidad se aplica al producto no al proceso.
- Para dar seguimiento al control de calidad se elabora un plan de inspección.

**Principales desventajas del control de calidad:**

- Para las empresas representa costos en la evaluación y reparación de los productos defectuosos.

- Se incorpora a la empresa la actitud de la tolerancia al error ocasionando que los empleados puedan pensar que ahora es más importante que no exista un control de calidad a que se generen errores durante el proceso.

Toda esta evolución histórica se dio principalmente en el ámbito de la fabricación de bienes pero ¿cómo se aplica el concepto del control de calidad al software?.

En [SOMM2002] se propone al ***Control de Calidad*** como una actividad de la administración de la calidad del software y lo define como **"la definición y promulgación de los procesos que se aseguran que los procedimientos y estándares para la calidad del proyecto son seguidos por el equipo de desarrollo de software"**.

La definición anterior implica que una empresa que desarrolla software debe tener estándares de control definidos para el proyecto de software y el control de calidad debe comprobar que estos se sigan durante su desarrollo.

## 3.2. Garantía de Calidad.

En [GES2003] se explica que el origen histórico del vocablo *Garantía de calidad* surgió como evolución del concepto control de calidad debido a sus desventajas (subcapítulo 3.1.) y a la importancia que fue adquiriendo la calidad como factor competitivo. Mientras que el control de calidad se enfocaba en detectar y corregir defectos del producto, el aseguramiento de calidad sustenta que la calidad se construye en los procesos, es decir, ***si cada proceso se realiza correctamente no hay razón para que aparezcan defectos y, por lo tanto, no será necesario controlar la calidad del producto***. Es en este momento cuando se incorpora a la empresa la idea de hacer las cosas bien a la primera y de que es el operario, y no el experto, el que está en una mejor posición para controlar su trabajo.

En [GES2003] también se dan las características y ventajas del aseguramiento de calidad, las cuales se resumirán a continuación.

**Principales características del aseguramiento de calidad:**

- La responsabilidad de la calidad la tienen compartida los operarios y el departamento de control de calidad.

- El aseguramiento de calidad surge porque se desea evitar el error.

- La aplicación de calidad es al proceso productivo.
- Para dar seguimiento a la calidad se elabora el Manual de calidad.

**Principales ventajas del aseguramiento de calidad:**

- Los operarios se sienten más comprometidos con la calidad de sus resultados.
- El operario puede aportar ideas de mejora puesto que es el que conoce mejor que nadie su trabajo.
- Los defectos se detectan en el momento en que se producen, evitando que se propaguen a etapas posteriores.
- Al disminuir los defectos y los gastos de control de calidad disminuye el costo de calidad.

Toda esta evolución histórica se dio principalmente en el ámbito de la fabricación de bienes pero ¿cómo se aplica el concepto del aseguramiento de calidad al software?.

En [PRES2002] se define a la ***Garantía de Calidad*** en el software como **"la auditoría y las funciones de información de la planificación"**, y su objetivo es proporcionar la gestión para informar de los datos necesarios acerca de la calidad del producto, de tal manera que se adquiere una visión mas profunda acerca de la calidad.

Finalmente, en [GES2003] se encuentra la Figura 3.1. con las diferencias fundamentales entre el CONTROL DE CALIDAD, explicadas en el subcapítulo anterior, y la GARANTÍA DE CALIDAD.

## 3.3. Costo de Calidad.

**Los controles de la calidad de los que se hablaban en la sección 3.1 y 3.2 y todas las actividades relacionadas en la obtención de la calidad tienen un costo**, a este costo se le conoce como ***Costo de Calidad***.

El costo del ciclo de vida de un software esta compuesto principalmente de dos factores, el costo de desarrollo y el costo incurrido en el soporte del producto entregado. El costo del desarrollo es determinado por varias características del software, tales como, su complejidad, la capacidad del equipo de trabajo y la tecnología empleada para el desarrollo, entre otras. El costo incurrido en el

soporte del producto es influenciado directamente por la calidad del producto entregado y los factores relacionados al proceso de mantenimiento.

| | CONTROL DE CALIDAD | GARANTÍA DE CALIDAD |
|---|---|---|
| **CONCEPTO DE CALIDAD** | Conformidad con las especificaciones | Aptitud para el uso |
| **RESPONSABILIDAD DE LA CALIDAD** | Departamento de Calidad | Departamento de Calidad + operarios |
| **SE ACTÚA PORQUE...** | Se detecta un error | Se intenta evitar el error |
| **APLICACIÓN DE LA CALIDAD** | Al producto | Al proceso productivo |
| **ACTUACIÓN** | Corregir el error | Modificar el procedimiento |
| **PARTICIPACIÓN DEL PERSONAL** | Sólo Departamento de Calidad | Departamento de Calidad + operarios |
| **IMPORTANCIA DE LA PARTICIPACIÓN** | No se espera participación | Importante |
| **GENERACIÓN DE VALOR AÑADIDO** | No está claro | Si |
| **MATERIALIZACIÓN** | Plan de Inspección | Manual de Calidad |
| **FILOSOFÍA** | Arreglo | Prevención |

**Figura 3.1. Diferencias entre Control de calidad y Garantía de calidad.**

En [PRES2002] el costo de calidad lo dividen en tres categorías:

- ***Costos de prevención***, que como su nombre lo dice son los costos acarreados por prevenir, tales como, planificación de la calidad, revisiones técnicas formales, costos del equipo de pruebas, entre otros.

- ***Costos de evaluación***, definidos como las actividades realizadas para tener una visión más profunda de la condición del producto, tales como las inspecciones en el proceso y entre procesos, las pruebas, etc.

- ***Costos de fallos***, definidos como los que desaparecerían si no surgieran defectos. Estos costos se subdividen en costos de fallos antes de la entrega del software al cliente y costos de fallos después de la entrega, como ejemplos de costos de fallos antes de la entrega se tienen las revisiones, las correcciones, etc., como ejemplos de costos de fallos después de la entrega se tienen las respuestas a quejas, la devolución y sustitución de productos, soporte en línea, etc.

Cabe mencionar que los costos e implicaciones de defectos después de la entrega del producto al cliente son más elevados que los anteriores a la entrega.

## 3.4. Actividades del SQA[1].

Para dar a conocer al lector en que parte del proceso de desarrollo del software se debe planear su calidad, a continuación, se listan los aspectos que deben considerarse al inicio de un proyecto de desarrollo de software [PIAT2000]:

- La selección del modelo del ciclo de vida para el desarrollo del software.
- Los aspectos relacionados con la financiación, la disponibilidad y la viabilidad.
- La definición del entorno del proyecto: metodologías, técnicas y herramientas.
- Planificación de la gestión del proyecto.

A partir de estos puntos se elabora un plan de gestión del proyecto de software que abarca los siguientes puntos:

- La gestión día a día del proyecto.
- La planificación del **Aseguramiento de la Calidad del Software**.
- La planificación de la documentación que debe generarse a lo largo del ciclo de vida del software.

Se tiene entonces que es durante el desarrollo del plan de gestión del proyecto de un software donde debe considerarse el plan para asegurar su calidad. Con base en la experiencia laboral adquirida en el desarrollo de sistemas en México se puede comentar que la planificación de las actividades para asegurar la calidad del software es la tarea a la que se le da menor importancia o en algunos casos ni siquiera se planea.

**"El Aseguramiento de calidad del software es el establecimiento de un marco de trabajo de procedimientos y estándares organizacionales que conduce a un software de calidad".**

De la definición anterior se entiende que el Aseguramiento de Calidad del Software es la tarea en la que se establecen las actividades planificadas y sistemáticas, que cumplen con estándares de calidad organizacionales, requeridas para asegurar la calidad del software. Algunas de las personas que se encuentran relacionadas con la responsabilidad de la calidad del software son ingenieros de software, líderes de proyecto, clientes y un grupo de SQA.

[1] Software Quality Assurance: siglas empleadas para designar al Aseguramiento de Calidad de Software.

El personal del grupo SQA debe mirar al software desde el punto de vista del cliente y contestar las siguientes preguntas ¿se ha realizado el desarrollo del software de acuerdo con estándares preestablecidos? ¿han desempeñado apropiadamente sus papeles las disciplinas técnicas como parte de la actividad de SQA?, entre otras.

Una vez que se definió lo que es el Aseguramiento de Calidad de software se listarán las actividades que deben realizarse en ella.

Según [PRES2002] y [PIAT2000] las actividades a realizar por las personas involucradas en el Aseguramiento de la calidad del software se dividen en dos grupos:

1. Las actividades que realiza el personal de sistemas como son las revisiones técnicas formales y las pruebas formales; y

2. Las actividades realizadas por el grupo independiente de SQA, las cuales se listan a continuación:

   a) *Establecimiento de un plan de SQA para un proyecto*. Los planes de SQA son específicos para cada proyecto porque deben ser coherentes con el sistema de calidad de la empresa y deben indicar lo siguiente:

      - Objetivos de la calidad del proyecto.

      - Gestión del aseguramiento de la calidad, por ejemplo: organización personal, actividades, responsabilidades, etc.

      - Documentación mínima exigible a los desarrolladores de software, por ejemplo: especificaciones, diseño, documentación de usuario, etc.

      - Estándares, normas y prácticas que hay que cumplir, ejemplo: estilo de programación y de comentarios.

      - Actividades de revisión y auditorias.

      - Herramientas, por ejemplo: analizadores de código; técnicas y métodos, por ejemplo: revisiones e inspecciones.

      - Realimentación de información proporcionada al equipo de proyecto de software.

   b) *Participación en el desarrollo de la descripción del proceso de software del proyecto*. Una vez que el área de sistemas elige un proceso para el proyecto a desarrollar el grupo de SQA debe revisar que este proceso

se ajuste a la política de la empresa y los estándares internos de software.

c) *Revisión de las actividades de ingeniería del software para verificar su ajuste al proceso de software definido.* El grupo de SQA identifica, documenta y verifica que se hayan realizado las correcciones de las desviaciones desde el proceso.

d) *Auditoría de los productos de software designados para verificar que se ajusten a los definidos.* El grupo SQA revisa los productos seleccionados; identifica, documenta, da seguimiento a las desviaciones y revisa que se hayan hecho las correcciones.

e) *Asegura que las desviaciones encontradas en el trabajo y en el producto se documenten y se manejen de acuerdo con un procedimiento establecido.*

f) *El grupo SQA coordina el control y la gestión de cambios y ayuda a recopilar y analizar las métricas del software.*

## 3.5. Técnica del Aseguramiento de Calidad Estadística.

Como se mencionó en el punto anterior entre las actividades del Aseguramiento de Calidad de Software está la de recolectar métricas apropiadas para asegurar que el trabajo de desarrollo de software sea efectivo y de calidad, para la recolección de éstas métricas existen diferentes técnicas y en este apartado se explicará una de ellas.

Como se describió en el capítulo 2, existen muchos modelos para evaluar la calidad del software y generar métricas que se basan en medir factores tales como la funcionalidad, la eficiencia, la fiabilidad, la usabilidad, la facilidad de mantenimiento, la portabilidad, etc., por desgracia estos factores son difíciles de medir directamente; es por esa razón que los modelos relacionados con la eliminación de defectos[2] están siendo usados por las empresas que desean iniciar un programa de control de calidad que les permita establecer más cuantitativamente la calidad del software; además de que, como se vio en este capítulo, la reducción en el número de defectos de un software reduce el costo de calidad del software.

[2] Estos modelos se basan, como se explica en el capítulo 2, en la idea intuitiva de que la calidad de un software es equivalente a la ausencia de defectos y cuya principal ventaja es que para una empresa que desea iniciar un programa de control de calidad es relativamente sencillo la recolección de defectos para conocer el estado actual del proyecto y a partir de éste poder mejorarlo.

El desarrollo de este tipo de modelos de eliminación de defectos está fuera del alcance de este trabajo pero las personas interesadas pueden encontrar más información en [CARD2003].

Una de las técnicas de los modelos relacionados con la eliminación de defectos que está ganando auge en las industrias desarrolladoras de software es el ***Aseguramiento de la Calidad Estadística del software***.

La técnica del Aseguramiento de Calidad Estadística de software implica los siguientes pasos [PRES2002]:

1. Se agrupa y se clasifica la información sobre los defectos del software.

2. Se intenta encontrar la causa subyacente de cada defecto.

3. Mediante el principio de Pareto (el 80% de los defectos se pueden encontrar en el 20% de las posibles causas), se aísla el 20% (los vitales).

4. Una vez que se identifican las causas vitales, se actúa para corregir los problemas que han producido los defectos.

[PRESS2002] comenta la importancia de esta técnica cuando escribe "este concepto relativamente sencillo representa un paso importante hacia la creación de un proceso de ingeniería del software adaptativo en el cual se realizan cambios para mejorar aquellos elementos del proceso que introducen errores".

La recolección de los defectos para la aplicación de la técnica de Aseguramiento de Calidad Estadística puede hacerse antes o después de la entrega del producto al cliente.

Si la recolección se realiza antes de la entrega al cliente, es decir, durante el desarrollo del software, es conveniente que se realice en cada fase del desarrollo del software, de esta manera se tendrá el número de errores detectados durante las fases de: análisis, diseño, codificación y pruebas. Si la recolección se realiza después de la entrega del software al cliente, los datos recolectados serán los defectos detectados por el usuario.

Una vez que los errores/defectos son recolectados se construye una tabla desplegando todas las causas categorizadas de los errores/defectos, el número de errores/defectos por cada causa y si los errores/defectos fueron graves, moderados o leves. Una vez que las causas vitales son determinadas (éstas deben ser definidas con base a lo que se desea mejorar, por ejemplo: reducir el número de defectos graves) pueden tomarse acciones correctivas. Con la aplicación de está técnica el esfuerzo del personal se enfoca en lo importante.

La actividad de recolección de datos es indispensable para el proceso de recopilación de métricas de software[3], independientemente del aspecto del desarrollo del software que se desee medir y de la técnica a aplicar, por lo que para una empresa que desea iniciar un programa de medición de software la aplicación de la técnica de Aseguramiento de Calidad Estadística lo introduce "fácilmente" tanto en la recolección de datos (paso 1 de la técnica) como en la generación de métricas, ya que para otras técnicas, la recolección de datos es un proceso previo e independiente a la generación de métricas.

De esta forma se ha cumplido con el objetivo de conocer el origen del Aseguramiento de la calidad como la evolución del control de la calidad y conocer los costos que estas actividades involucran en el desarrollo del software. También se describieron de forma general las actividades que se siguen para asegurar la calidad del software, enfocándose en las tareas de apoyo a la recolección de datos y mecanismos de medición, entre los que se encuentra la técnica de Aseguramiento de Calidad Estadística.

En el siguiente capítulo se aplicará, para nuestro caso práctico, la técnica de Aseguramiento de Calidad Estadística durante la fase de mantenimiento del software de una empresa.

[3] Como se explicó en la sección 2.5. de este trabajo.

# CAPÍTULO 4

# APLICACIÓN DE LA TÉCNICA DEL ASEGURAMIENTO DE CALIDAD ESTADÍSTICA.

# CASO PRÁCTICO.

"Muchos de nuestros sueños parecen imposibles;
luego improbables y finalmente inevitables"
Christopher Reeve

Cuando se ha estado inmerso en las áreas de desarrollo y mantenimiento de software por casi 12 años se acumula una gran experiencia en los problemas que se presentan en estas áreas y será en este capítulo donde se reflejará.

Cuando se vive diariamente con la presión de que se está corrigiendo un defecto del software que ha tenido detenida la operación de una empresa por 4 o más horas y siguen llegando más defectos detectados por otros usuarios, defectos de igual o menor importancia que el que se está resolviendo y que seguramente irán formando parte de la lista de defectos pendientes de solución, esta situación se puede llegar a hacer una costumbre o también se puede suspirar y decir que es otro día común y corriente en el área de mantenimiento de un sistema, pero no se reconoce que algo anda mal y que es necesario analizarlo para poder mejorarlo.

En este capítulo se trata la importancia de la fase de mantenimiento de un software dentro del ciclo de vida de los sistemas, se realiza una propuesta respecto a una serie de actividades que deben realizarse en ésta fase y finalmente, se aplica la técnica de Aseguramiento de Calidad Estadística, explicada en el capítulo anterior, para analizar las actividades realizadas durante la fase de mantenimiento de software y distinguir en cual de ellas se necesita mejorar ya que es la que está ocasionando realizar grandes esfuerzos.

Cabe mencionar que realizar este caso práctico implicó aproximadamente ocho meses de dedicación diaria.

## 4.1. Fase de Mantenimiento del Software.

La fase de Mantenimiento del software es la fase del ciclo de vida de un software que comienza una vez que se terminan las pruebas y se realiza la entrega formal al cliente; aunque la fase de mantenimiento es la última fase del ciclo de vida del software es la más costosa y en la que se emplea el mayor esfuerzo del personal, según lo indica la figura 4.1. obtenida de [PIAT2000], en la que se puede observar que el mantenimiento del software representa las dos terceras partes de la vida del sistema.

Algunos de los factores que afectan directamente estos costos son:

- Si la participación del cliente es pobre durante las fases anteriores a la entrega del software, éste no cumplirá con las necesidades del usuario y el esfuerzo por alcanzar estas necesidades durante la fase de mantenimiento incrementará el costo de ésta última.

- La falta de conocimiento de la importancia, costo y esfuerzo de la fase de mantenimiento del software, en las actividades que hay que realizar en ella,

provoca que sea la fase en la que no existan métodos, ni técnicas, ni herramientas y además sea a la que menos personal se le asigne.

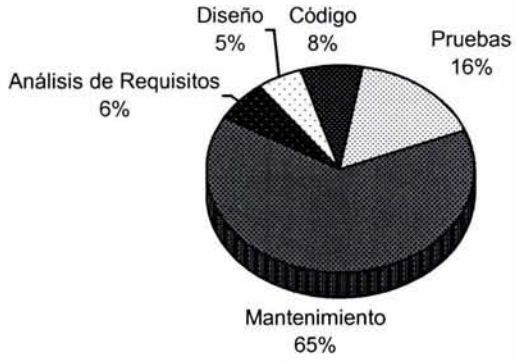



**Figura 4.1. Distribución del costo y el esfuerzo en grandes sistemas del ciclo de vida.**

- Las modificaciones continuas a las definiciones, a la estructura de la base de datos y al código, la mayoría de las veces realizadas por personal diferente al que lo desarrolló, provocan que el sistema se vuelva cada vez más complejo, ya que se pierde su documentación y estandarización.

- Debido a que durante la fase de mantenimiento el software ya se encuentra siendo usado por el cliente, las actividades de esta fase suelen realizarse bajo mucha presión y las correcciones a defectos que no se realizan analizando su origen dan lugar a que el mismo defecto se repita en un futuro o a que la corrección origine otros defectos.

- Debido a que el mantenimiento es la última fase del ciclo de vida del software, se le considera menos creativa que el resto del desarrollo del software y es muy común ver que al personal nuevo se le asignen actividades de mantenimiento de un software que apenas conoce.

**"La fase de mantenimiento es en la que al software, después de entregado, se le realizan todas las adecuaciones necesarias debidas a defectos, a cambios de entorno o a mejoras funcionales solicitadas por los usuarios"** [PIAT2000].

Según la definición anterior se distinguen tres diferentes tipos de mantenimiento:

- ***Mantenimiento perfectivo y adaptativo***: son las actividades que se realizan para mejorar el sistema desde el punto de vista del usuario y para adaptar el

sistema a los cambios en su entorno. Ejemplo: adecuaciones realizadas al software por cambios en el sistema operativo bajo el que se está trabajando.

- ***Mantenimiento correctivo***: son las actividades enfocadas a corregir defectos detectados por los usuarios una vez que se les entrega el sistema, ejemplo: lentitud en tiempos de respuesta, salidas incorrectas del sistema, pérdidas de información, etc.

- ***Mantenimiento preventivo***: son las actividades que se realizan para asegurar que el mantenimiento futuro del sistema sea más fácil.

En [PIAT2000] se muestra una gráfica que refleja que de todo el mantenimiento aplicado a un software el 17% es correctivo, el 5% es preventivo y el 78% es perfectivo y adaptativo pero tomando en cuenta la experiencia adquirida se puede concluir que en los sistemas comprados, mientras más grandes sean y mientras se adapten menos a las necesidades, el mantenimiento correctivo es tan grande que imposibilita al personal para dedicarse a los otros tipos de mantenimientos, los cuales son muy importantes para la vida futura del sistema.

Cada adecuación al sistema durante la fase de mantenimiento debe pasar por casi todas las fases del ciclo de vida de un sistema, ya que cuando se presenta un defecto a corregir (mantenimiento correctivo), cuando se necesita una adecuación del sistema por algún cambio en su entorno (mantenimiento adaptativo) o cuando se le realiza una mejora funcional (mantenimiento perfectivo), las actividades, que según la experiencia adquirida, debe realizar el personal de mantenimiento para implementar el cambio son:

1. ***Análisis del requerimiento***: cada adecuación que se necesita realizar al software, ya sea para resolver un defecto o para realizarle una mejora, es necesario que sea analizada a detalle desde la definición original hasta el posible impacto que puede tener en otros módulos. A veces, el análisis detallado de los defectos detectados da como resultado que el defecto era originado por la operación incorrecta del software y que el sistema no necesita ninguna adecuación; sin éste análisis, los desarrolladores se pueden dejar llevar por la presión del usuario y realizar modificaciones innecesarias, mismas que pueden generar nuevos defectos en el sistema. Esta es la actividad en la que el usuario que realiza el requerimiento debe estar completamente involucrado y es la actividad que no debe saltarse si existe una duda del requerimiento por parte del personal de mantenimiento.

2. ***Diseño del requerimiento***: si después de que se ha analizado el requerimiento se decide que es necesaria una adecuación a la estructura actual de la base de datos (ejemplo: modificar o agregar tablas, campos, índices, etc.) es importante que se conozca la relación actual entre las tablas para que la adecuación cumpla con la normalización y consistencia de la base de datos. Esta actividad no es forzosa que se realice, sólo cuando la corrección o adecuación así lo requiera.

3. ***Desarrollo del cambio***: en esta actividad se realizarán o se modificarán, según sea el caso, las pantallas, reportes, procedimientos, funciones, etc. que sean necesarias para la corrección o la mejora del sistema. Esta es la actividad a la que los desarrolladores tienden a ver como la primera y más importante en la fase de mantenimiento, pero es precisamente la que puede ocasionar más problemas si no va precedida de un análisis del requerimiento, ejemplo: el modificar erróneamente código, ya sea porque se desconoce la relación entre procedimientos o porque la presión en tiempo de la adecuación no permite analizar el código, puede provocar que el sistema deje de funcionar, sobre todo si el código al que se le hace la modificación es parte medular del sistema. Esta actividad también debe contener las pruebas de la modificación por parte del desarrollador, éstas pruebas no son tan exhaustivas como las de la siguiente actividad pero es responsabilidad del desarrollador que la adecuación al sistema no llegue con errores de compilación o de navegación a las pruebas del usuario.

4. ***Realización de pruebas***: una vez que se ha realizado la adecuación al sistema es necesario efectuar las pruebas, en éstas hay que probar que el sistema funcione como venía funcionando antes del cambio y que la adecuación realizada dé los resultados esperados. Si las pruebas de la adecuación no se realizan o son incompletas puede que la adecuación, una vez que sea liberada, genere defectos más graves de los que se supone iba a corregir. Esta actividad depende en gran parte de la definición de los requerimientos ya que las pruebas se efectúan en base a lo requerido, si la requisición fue mal hecha las pruebas pocas veces lo detectaran, es por esta razón que es importante que también en esta actividad el usuario este involucrado. De preferencia estas pruebas deben ser realizadas por una persona que no sea el desarrollador que realizó la adecuación.

5. ***Actualización de la documentación***: la importancia de esta actividad es relevante, sin embargo es la actividad que rara vez se hace debido a que es la actividad que siempre se deja al final de una modificación y nunca hay tiempo para ella. Tener la documentación al día (los manuales de funcionalidad o los manuales de la estructura de la base de datos) permite que otros desarrolladores puedan conocer y modificar el sistema con más seguridad.

Todas estas actividades deben realizarse con mucho cuidado ya que hay que tomar en cuenta que la adecuación a realizar se hará sobre una estructura de datos que se está usando y sobre código que se encuentra referenciado entre sí, el perder de vista o el desconocer este punto tan importante puede ocasionar que varias partes o todo el software quede sin funcionar. Siempre hay que tener en cuenta que corregir errores en un sistema aún no liberado no tiene el mismo impacto que corregir defectos en un sistema ya liberado.

## 4.2. Caso práctico.

Para el caso práctico se aplicará la "técnica de Aseguramiento de Calidad Estadística" para encontrar las actividades del mantenimiento de un software, descritas en la sección 4.1, en las que se originan los defectos más graves y aplicar medidas de corrección, reduciendo de esta manera costos y esfuerzos durante la fase de mantenimiento.

Como se comentó en la sección 3.5. el objetivo de la técnica de Aseguramiento de Calidad Estadística es que el esfuerzo del personal se enfoque en lo que realmente interesa (causa vital). Para el caso práctico de este trabajo la causa vital es identificar la actividad del mantenimiento que genera más defectos graves, ya que en la fase de mantenimiento del software se necesita que el personal corrija defectos y realice las adecuaciones al software para soportar los cambios del entorno, pero como son tantos los defectos que se presentan, el personal enfoca la mayoría de su tiempo y esfuerzo a corregirlos, asignándole muy poco tiempo al mantenimiento adaptativo del software. Se considera que si se logra conocer el origen y la gravedad de los defectos se podrá enfocar correctamente el esfuerzo del personal para corregir de forma definitiva los defectos que más esfuerzo le requieren (graves), permitiéndole dedicarse a otras actividades. Para verificar esta hipótesis se aplicará la técnica de Aseguramiento de Calidad Estadística (pasos descritos en la sección 3.5) al proceso de este departamento. A continuación se describen las actividades que se realizaron para cubrir los pasos 1 y 2 de la técnica.

### 4.2.1. Agrupación e identificación de las causas subyacentes de los defectos del software.

Para el caso práctico, los defectos se recolectaron durante 6 meses de la fase de mantenimiento de algunos módulos del software integral PT1[1] que fue comprado para una empresa, la cual hasta el día de hoy no cuenta con un programa de control de calidad del software, así que el personal de mantenimiento se ve continuamente en apuros para cumplir con los tiempos de entrega pactados, lo que ocasiona que el personal se desgaste tanto física como emocionalmente y no alcance a comprender ¿por qué no se siente satisfecho con su trabajo?.

Para este trabajo los defectos que se consideraron se agruparon en las siguientes categorías:

[1] Se ha modificado el nombre del software para proteger información confidencial de la empresa. Para que el lector se forme una idea de la complejidad del software citado podemos decir que éste consta de más de 500 tablas, más de 1000 procedimientos y funciones de base de datos y más de 200 pantallas.

1. Detectados por el usuario final
2. Detectados por el área de mantenimiento del sistema.

El paso 1 de la técnica es "agrupar y clasificar la información sobre los defectos del software" y el paso 2 es "encontrar la causa subyacente de cada defecto". Puesto que la empresa no tiene un programa de control de calidad de software, en el área de mantenimiento del sistema no se cuenta con un formato para la recolección de información de los defectos que se presentan, así que para cubrir los pasos 1 y 2 de la técnica se diseñó una matriz a la que se denominó "RECOLECCIÓN DE DEFECTOS" (figura 4.2.). La información recabada en la matriz permitirá conocer tanto la gravedad del defecto como la actividad del mantenimiento que lo originó (causa vital). Las columnas de esta matriz son los aspectos a evaluar de los defectos y los renglones son los defectos mismos. El llenado de esta matriz de defectos y su evaluación permitirá generar la tabla "RESUMEN DE RECOLECCIÓN", figura 4.3, a través de la cual se aplicará la técnica de Aseguramiento de Calidad Estadística.

Las columnas de la matriz de RECOLECCIÓN DE DEFECTOS se dividen en **evaluables** y *no evaluables*[2]. Las **evaluables** serán calificadas, por el personal del mantenimiento del software, con 7, 8 y 9 dependiendo del costo y esfuerzo empleado por la persona de mantenimiento que atendió el defecto; las *no evaluables* son más bien informativas y descriptivas, pero servirán para obtener estadísticas futuras.

El promedio de las columnas **evaluables** de cada defecto indicará la gravedad final del defecto detectado, la gravedad de un defecto puede ser GRAVE, MODERADO o LEVE.

> Defecto grave. Un defecto se considerará grave cuando el promedio de los aspectos **evaluables** dé como resultado un valor entre 8.5 y 9. En la columna ORIGEN DEL DEFECTO se identificará con 9.
>
> Defecto moderado. Un defecto se considerará moderado cuando el promedio de los aspectos **evaluables** dé como resultado un valor entre 7.5 y 8.4. En la columna ORIGEN DEL DEFECTO se identificará con 8.
>
> Defecto leve. Un defecto se considerara leve cuando el promedio de los aspectos **evaluables** dé como resultado un valor entre 7 y 7.4. En la columna ORIGEN DEL DEFECTO se identificará con 7.

A continuación se describe a detalle cada una de las columnas que conforman la matriz de RECOLECCIÓN DE DEFECTOS.

---

[2] Para distinguir fácilmente las columnas **evaluables** de las *no evaluables* se usaron los tipos de letras negrita y cursiva, estos tipos de letra se emplean tanto en el texto como en la "matriz de recolección de defectos".

Las columnas *no evaluables* en la matriz son:

- *Fecha inicial y fecha final*. Estas fechas comprenden desde que el defecto empieza a ser analizado por el personal de mantenimiento hasta que su solución es liberada a operación.

- *Código del defecto*. A cada defecto recolectado se le colocó un código de identificación único, éste código servirá para ubicar si los defectos se presentan más de una vez.

- *Descripción del defecto*. Es la descripción corta del defecto recolectado (la descripción no puede ser detallada para mantener la confidencialidad de la empresa).

- *Módulo del defecto*. Los sistemas computacionales se dividen en módulos, por ejemplo: módulo de altas, módulo de bajas, módulo de cambios, módulo de consultas, etc. para el caso práctico, en cuestión, se identificará el módulo en el que se presenta el defecto para posteriormente identificar los módulos con más defectos (el nombre real del módulo no puede ser presentado para mantener la confidencialidad de la empresa así que a cada módulo se le identificará por la letra "Z" y un número consecutivo).

- *Prioridad jerárquica*. Este aspecto es la prioridad asignada por la Dirección de Sistemas de la empresa para la corrección del defecto, es decir, hay defectos detectados que son tan importantes que la Dirección de Sistemas de alguna manera le asigna su gravedad.

Las columnas **evaluables** en la matriz de RECOLECCIÓN DE DEFECTOS son:

- **Productos**. Debido a que la empresa ofrece al cliente varios productos es de interés conocer, a través de la evaluación de este aspecto, si el defecto detectado afecta a uno o varios productos y el nivel en que los afecta. (El nombre de los productos no puede ser detallado para mantener la confidencialidad de la empresa por eso los llamaremos A, B y C).

- **Afectación a la operación**. Este aspecto evalúa el grado en el que el defecto afectó la operación del sistema.

- **Reproceso**. En este aspecto se evalúa el grado del reproceso que se necesitó para corregir el defecto, si este fue necesario.

- **Costo $**. En este aspecto se evalúa el grado en que el defecto causó un costo monetario en la operación.

- **Costo HH**. En este aspecto se evalúa el grado de esfuerzo del personal del área de mantenimiento que se necesitó para corregir el defecto.

- **Repercute otras áreas**. Este aspecto evalúa el grado en que el defecto detectado afectó a otras áreas operativas además del área en que se presentó.

- **Programación**. En este aspecto se mide el grado de codificación, realización de pantallas, etc. que el personal de mantenimiento tuvo que realizar para la corrección del defecto.

- **Recapacitación**. En este aspecto se evalúa el grado de recapacitación al usuario que se tuvo que realizar para corregir el defecto.

- **Evaluación Total**. Esta columna de la matriz es la suma de las evaluaciones de los ocho aspectos descritos anteriormente.

- **Cuenta Evaluaciones**. Ésta columna de la matriz contiene el número de aspectos evaluados para el defecto detectado.

- **Evaluación Promedio**. Ésta columna es el resultado de dividir la columna "Evaluación Total" entre la columna "Cuenta Evaluaciones" y es la que nos indicará la gravedad del defecto (GRAVE, LEVE o MODERADO).

- **Horas Empleadas en la solución**. En esta columna de la matriz se registra las horas que emplea el personal de mantenimiento en la corrección del defecto, este tiempo incluye el análisis, diseño (si fue necesario para la corrección), desarrollo y pruebas; este registro nos servirá para conocer el tiempo empleado por el personal en el mantenimiento correctivo del sistema.

- **Horas de afectación a la producción**. En esta columna se registrarán las horas que el defecto afectó la operación de los usuarios del sistema en el modulo en el que se presentó.

- **Actividad origen del defecto**. El personal que corrige el defecto debe categorizarlo en la actividad del mantenimiento que lo originó y una vez que tenga identificada la actividad origen le asignará a ésta un 7,8 ó 9 si el defecto se evaluó como LEVE, MODERADO o GRAVE en la columna "Evaluación Promedio".

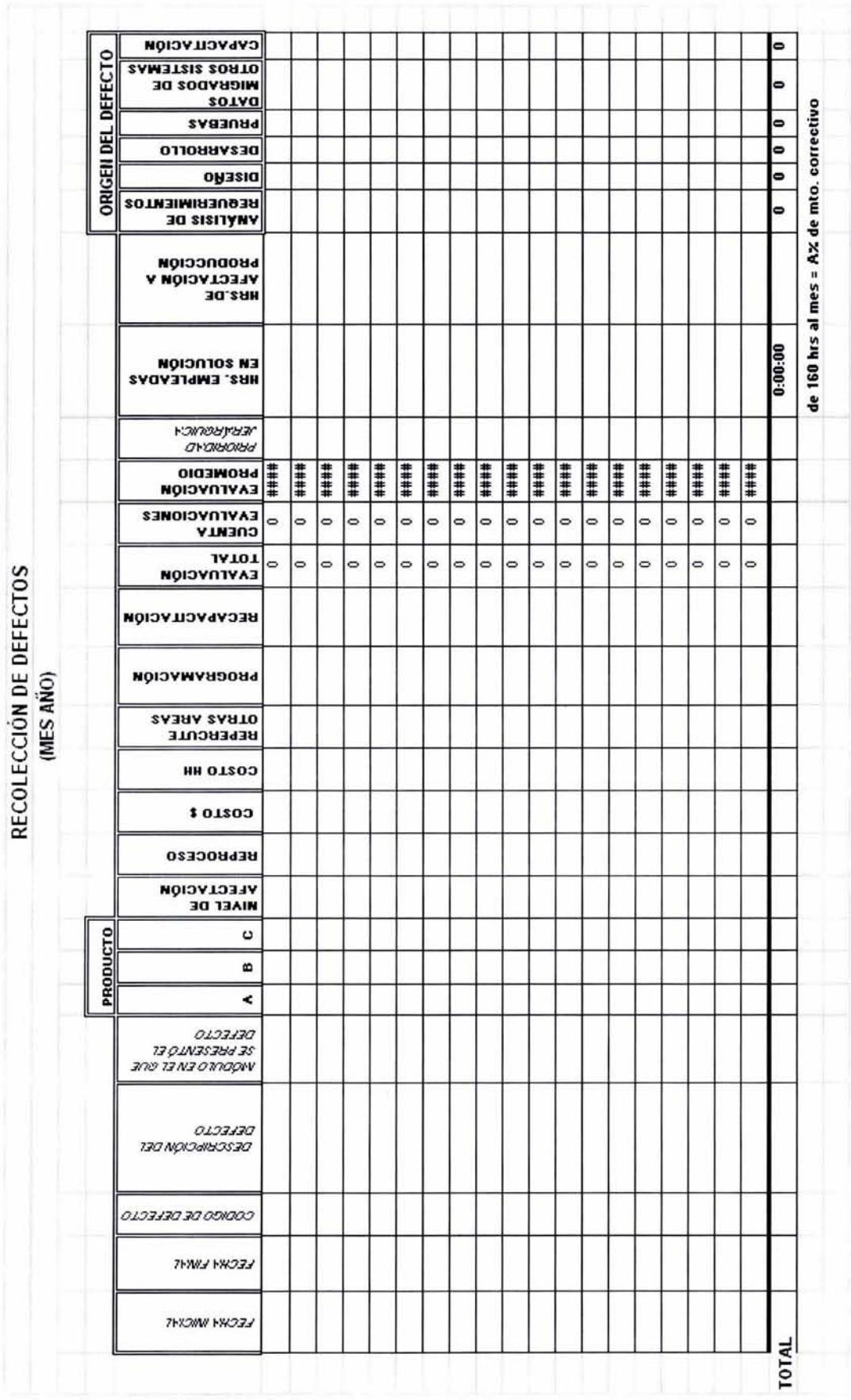



Figura 4.2. Matriz de recolección de defectos.

**Método para el llenado de la matriz de RECOLECCIÓN DE DEFECTOS.**

Cada defecto que llega al área de mantenimiento del sistema es asignado a una persona de ésta, la persona analiza con el usuario el defecto, si es necesario, realiza el diseño de la corrección, desarrolla la corrección y realiza sus pruebas. De esta manera el personal de mantenimiento conoce a fondo el defecto y debe llenar y evaluar los datos de la matriz de RECOLECCIÓN DE DEFECTOS.

Por cada mes de recolección se realizará una matriz de RECOLECCIÓN DE DEFECTOS que agrupe los defectos que se detectaron durante ese mes.

Al final de la matriz se totalizan las horas empleadas en la solución de los defectos del mes para obtener el porcentaje mensual del personal dedicado al *mantenimiento correctivo*, también se obtiene el número total de ocurrencias de los defectos categorizados en cada actividad del mantenimiento que dio origen al defecto.

Ejemplo de llenado: suponiendo que un defecto tuvo ocho aspectos evaluados, seis de los cuales se evaluaron con 8 y los dos restantes con 7; la suma de éstas evaluaciones da una *evaluación total* de 62, al dividirla entre las 8 evaluaciones da la *evaluación promedio* de 7.75, la cual cae en la categoría de *defecto moderado* por lo que se asignará un peso de 8 a la *actividad origen* del defecto una vez que sea identificada por el personal.

Para el llenado de la matriz de RECOLECCIÓN DE DEFECTOS se tuvieron en cuenta las siguientes consideraciones:

- A veces el período de fechas desde que se empieza a analizar el defecto por el personal de mantenimiento (columna *fecha inicial*) hasta que su solución es liberada al cliente (columna *fecha final*) contempla más de un día pero no implica que para su solución el personal haya empleado los días completos sino que pueden ser horas de un día y horas de otro día.

- Todos los aspectos *no evaluables* deben ser llenados en la matriz pero algunos de los aspectos **evaluables** puede que no se presenten para la solución del defecto y por lo tanto no serán evaluados.

- Dado que el personal que recibe y da solución al defecto lo conoce a fondo puede identificar la actividad del mantenimiento que lo originó y categorizarlo en ella.

- Algunas veces un defecto se inicia en un mes y se libera al cliente hasta otro mes, en estos casos el defecto se categorizará una sola vez en la matriz del mes en que se detectó y en el otro mes la descripción del defecto se sombreará para marcarlo como *defecto no detectado* y no se categorizará para no volver a contarlo como defecto detectado, pero sí se tomarán en cuenta las

horas empleadas de ese mes en su solución ya que es parte del tiempo empleado del personal en mantenimiento correctivo.

- Algunas veces, los defectos que fueron detectados por el área de mantenimiento del sistema son identificados antes de que el defecto afecte a la operación de la empresa, en estos casos la columna HRS. DE AFECTACIÓN A PRODUCCIÓN quedará en blanco.

- Se definirán como *falsas alarmas* a los defectos que el usuario reporta y que después de analizarlos no es necesario dar solución ya que lo reportado no es un defecto, para estos casos la columna HRS. DE AFECTACIÓN A PRODUCCIÓN también quedará en blanco.

**Tabla "RESUMEN DE RECOLECCIÓN" de defectos para la técnica de Aseguramiento de Calidad Estadística.**

Una vez que se recolectó la información en la matriz de RECOLECCIÓN DE DEFECTOS de un mes completo, se genera la tabla RESUMEN DE RECOLECCIÓN de ese mismo mes, con el total de las ocurrencias de los defectos categorizados en cada actividad del mantenimiento que le dio origen y que se obtuvieron en la matriz de RECOLECCIÓN DE DEFECTOS.

La tabla RESUMEN DE RECOLECCIÓN esta compuesta en sus renglones por las actividades origen del defecto y en sus columnas por el total de los defectos categorizados en cada actividad resultantes de la matriz de RECOLECCIÓN DE DEFECTOS y el total de defectos evaluados como graves (número de ocurrencias de 9's) en la matriz de recolección, el total de defectos evaluados como moderados (número de ocurrencias de 8's) en la matriz de recolección y el total de defectos evaluados como leves (número de ocurrencias de 7's) en la matriz de recolección. Con estos totales podemos obtener la (las) actividad(es) del mantenimiento del sistema que dieron origen a los defectos graves del sistema durante ese mes.

**RESUMEN DE RECOLECCIÓN**

(MES AÑO)

| ORIGEN DEL DEFECTO | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | | | | | | | | |
| Diseño | | | | | | | | |
| Desarrollo | | | | | | | | |
| Pruebas | | | | | | | | |
| Datos Migrados | | | | | | | | |
| Capacitación | | | | | | | | |
| **Totales** | **0** | **0.00%** | **0** | **0.00%** | **0** | **0.00%** | **0** | **0.00%** |

**Figura 4.3. Tabla Resumen de recolección.**

### 4.2.1.1. Matrices y tablas resultantes de la recolección de defectos del primer trimestre.

A continuación se tienen las matrices de RECOLECCIÓN DE DEFECTOS para los meses de Marzo a Mayo del 2003 en las figuras 4.4, 4.5 y 4.6, respectivamente.

## RECOLECCIÓN DE DEFECTOS
### (MARZO 2003)

| | | | | | PRODUCTO | | | | | | | | | | | | | | | ORIGEN DEL DEFECTO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA INICIAL | FECHA FINAL | CODIGO DE DEFECTO | DESCRIPCIÓN DEL DEFECTO | MÓDULO EN EL QUE SE PRESENTÓ EL DEFECTO | A | B | C | NIVEL DE AFECTACIÓN | REPROCESO | COSTO $ | COSTO HH | REPERCUTE OTRAS AREAS | PROGRAMACIÓN | RECAPACITACIÓN | EVALUACIÓN TOTAL | CUENTA EVALUACIONES | EVALUACIÓN PROMEDIO | PROBABILIDAD RESOLUCIÓN | HRS. EMPLEADAS EN SOLUCIÓN | HRS. DE AFECTACIÓN A PRODUCCIÓN | ANÁLISIS DE REQUERIMIENTOS | DISEÑO | DESARROLLO | PRUEBAS | DATOS MIGRADOS DE OTROS SISTEMAS | CAPACITACIÓN |
| 3-Mar | 11-Mar | A1 | Se cubrió periodo de más en registros "a" por un proceso másivo X en el producto C (300 casos) | Z1 | | | 9 | 9 | 9 | 7 | 9 | 8 | 9 | | 60 | 7 | 8.57 | 9 | 25:00:00 | | 9 | | | | | |
| 05-Mar | 05-Mar | A2 | Reproceso doctos. X del proceso másivo Y | Z2 | | 9 | | 9 | 9 | 8 | 9 | 9 | 8 | | 61 | 7 | 8.71 | 9 | 4:30:00 | 5:00:00 | 9 | | | | | |
| 06-Mar | 06-Mar | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 7 | 8 | | 7 | 7 | | | 50 | 7 | 7.14 | | 0:30:00 | 0:30:00 | | 7 | | | | |
| 10-Mar | 10-Mar | A4 | Grabación incorrecta de un dato en el proceso masivo X | Z1 | | | 7 | 7 | | | 7 | 7 | 7 | | 35 | 5 | 7 | | 2:00:00 | | | | | 7 | | |
| 10-Mar | 11-Mar | A5 | Grabación incorrecta de datos | Z3 | 9 | 9 | 9 | 9 | 8 | | 8 | 9 | 9 | 8 | 78 | 9 | 8.67 | 9 | 11:00:00 | 5:00:00 | 9 | | | | | |
| 11-Mar | 11-Mar | A6 | Inconsistencia de datos en interfaz automática X | Z1 | 7 | 7 | 7 | 7 | 9 | | 8 | 8 | 7 | | 60 | 8 | 7.5 | | 4:00:00 | | | | | | 8 | |
| 12-Mar | 12-Mar | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 7 | 8 | | 7 | 7 | | | 50 | 7 | 7.14 | | 3:30:00 | 0:30:00 | | 7 | | | | |
| 12-Mar | 12-Mar | A7 | Direccionamiento incorrecto de documentos Z | Z4 | | 9 | | 7 | 9 | 7 | 9 | 7 | 7 | | 52 | 7 | 7.43 | | 3:00:00 | | 7 | | | | | |
| 17-Mar | 17-Mar | A8 | Recepción de datos incorrectos de interfaz externa J | Z5 | 9 | 9 | | 9 | 8 | 8 | 8 | 9 | 8 | | 68 | 8 | 8.5 | 9 | 11:30:00 | 5:00:00 | 9 | | | | | |
| 18-Mar | 25-Mar | A9 | Cargo de penalización incorrecto a personal | Z3 | 8 | 8 | 8 | 7 | | 7 | 8 | 7 | 8 | | 61 | 8 | 7.63 | 8 | 19:00:00 | | 8 | | | | | |
| 20-Mar | 20-Mar | A10 | Pérdida de paquete de impresión a recuperar | Z4 | 8 | 8 | 8 | 8 | | 7 | 8 | | | 8 | 55 | 7 | 7.86 | 8 | 3:00:00 | 3:00:00 | | | | | | 8 |
| 24-Mar | 24-Mar | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 7 | 8 | | 7 | 7 | | | 50 | 7 | 7.14 | | 1:00:00 | 1:00:00 | | 7 | | | | |
| 24-Mar | 24-Mar | A10 | Pérdida de paquete de impresión a recuperar | Z4 | 9 | 9 | | 9 | | 7 | 7 | | | 8 | 46 | 6 | 7.67 | 9 | 1:30:00 | 1:30:00 | | | | | | 9 |
| 25-Mar | 28-Mar | A11 | Reproceso de registros de una entidad federativa | Z3 | 7 | 7 | 7 | 8 | 9 | 8 | 9 | | 8 | | 63 | 8 | 7.88 | 8 | 14:00:00 | 1:00:00 | 8 | | | | | |
| 27-Mar | 27-Mar | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 8 | 8 | | 8 | 8 | | | 53 | 7 | 7.57 | | 6:00:00 | | | 8 | | | | |
| 27-Mar | 27-Mar | A10 | Pérdida de paquete de impresión a recuperar | Z4 | 7 | 7 | | 8 | | 7 | 8 | | | 8 | 45 | 6 | 7.5 | 8 | 1:30:00 | 1:30:00 | | | | | | 8 |
| 28-Mar | 28-Mar | A7 | Direccionamiento incorrecto de documentos Z | Z4 | | 7 | | 7 | 7 | 8 | 8 | 9 | 7 | | 52 | 7 | 7.43 | 8 | 2:00:00 | 1:00:00 | 7 | | | | | |
| 31-Mar | 31-Mar | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 7 | 9 | | 8 | 7 | | | 51 | 7 | 7.29 | 9 | 2:00:00 | | | 7 | | | | |
| 31-Mar | 31-Mar | A12 | Cambio de estatus de un dato en el proceso masivo X | Z1 | | | 8 | 8 | | | 8 | | 8 | | 32 | 4 | 8 | 8 | 4:00:00 | 1:00:00 | 8 | | | | | |
| 31-Mar | 31-Mar | A13 | Reporte de auditoría con información de más | Z3 | 8 | 8 | 8 | 8 | 8 | | 8 | | 8 | 7 | 63 | 8 | 7.88 | 8 | 4:30:00 | 1:00:00 | | | | 8 | | |
| TOTAL | | | | | | | | | | | | | | | | | | | 123:30:00 | | 9 | 5 | 0 | 2 | 1 | 3 |

de 160 hrs al mes = 76.8% de mto. correctivo

Figura 4.4. Matriz de recolección de defectos de Marzo 2003.

## RECOLECCIÓN DE DEFECTOS
### (ABRIL 2003)

| | | | | | PRODUCTO | | | | | | | | | | | | | | | | ORIGEN DEL DEFECTO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA INICIAL | FECHA FINAL | CODIGO DE DEFECTO | DESCRIPCIÓN DEL DEFECTO | MÓDULO EN EL QUE SE PRESENTÓ EL DEFECTO | X | Y | Z | NIVEL DE AFECTACIÓN | REPROCESO | COSTO $ | COSTO HH | REPERCUTE OTRAS AREAS | PROGRAMACIÓN | RECAPACITACIÓN | EVALUACIÓN TOTAL | CUENTA EVALUACIONES | EVALUACIÓN PROMEDIO | PRIORIDAD ASIGNADA | HRS. EMPLEADAS EN SOLUCIÓN | HRS. DE AFECTACIÓN A PRODUCCIÓN | ANÁLISIS DE REQUERIMIENTOS | DISEÑO | DESARROLLO | PRUEBAS | DATOS MIGRADOS DE OTROS SISTEMAS | CAPACITACIÓN |
| 1-Abr | 1-Abr | A13 | Reporte de auditoría con información de más | Z3 | 8 | 8 | 8 | 8 | 8 | | 8 | | 8 | 7 | 63 | 8 | 7.88 | 8 | 3:00:00 | | | | | | | |
| 01-Abr | 04-Abr | A9 | Cargo de penalización incorrecto a personal | Z3 | 8 | 8 | 8 | 8 | | 7 | 8 | 8 | 8 | | 63 | 8 | 7.88 | 8 | 5:00:00 | | | | | | | |
| 03-Abr | 8-Abr | A14 | Regional específica con problemas en módulo Z4 | Z4 | 9 | 9 | 9 | 9 | | | 9 | 9 | 9 | 9 | 72 | 8 | 9 | 9 | 20:30:00 | 8:30:00 | | | | | | 9 |
| 7-Abr | 7-Abr | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 7 | 8 | | 7 | 7 | | | 50 | 7 | 7.14 | | 1:00:00 | | | 7 | | | | |
| 7-Abr | 10-Abr | A15 | Bloqueo (causa K) de impresión a nivel divisional | Z4 | 8 | 8 | 8 | 8 | | | 8 | 8 | 8 | 7 | 63 | 8 | 7.88 | | 6:00:00 | 3:00:00 | | 8 | | | | |
| 8-Abr | 8-Abr | A16 | 13000 Documentos a depurar generados incorrectamente | Z4 | | 9 | | 9 | | 9 | 7 | 9 | 7 | | 50 | 6 | 8.33 | | 3:30:00 | | | | | 8 | | |
| 08-Abr | 08-Abr | A17 | Falta representación de datos en un reporte | Z3 | 7 | 7 | 7 | 7 | | | 7 | | 7 | | 42 | 6 | 7 | | 1:00:00 | | | | | 7 | | |
| 8-Abr | 9-Abr | A18 | Dato cero en un campo | Z2 | 7 | 7 | 7 | 7 | | | 8 | 7 | 8 | | 51 | 7 | 7.29 | | 4:30:00 | | | | | 7 | | |
| 9-Abr | 10-Abr | A19 | Estado incorrecto de registros | Z2 | 8 | | | 8 | | | 8 | 8 | 8 | | 40 | 5 | 8 | | 7:30:00 | | | | | | 8 | |
| 10-Abr | 10-Abr | A10 | Pérdida de paquete de impresión a recuperar | Z4 | 7 | 7 | | 8 | | 8 | 8 | | | 8 | 46 | 6 | 7.67 | 8 | 1:00:00 | 1:00:00 | | | | | | 8 |
| 15-Abr | 15-Abr | A10 | Pérdida de paquete de impresión a recuperar | Z4 | 8 | 8 | | 7 | | 8 | 8 | | | 8 | 47 | 6 | 7.83 | 8 | 2:00:00 | 2:00:00 | | | | | | 8 |
| 15-Abr | 16-Abr | A20 | Funcionalidad incorrecta en pantalla lanzadora de reporte | Z4 | | 9 | | 9 | | | 8 | 9 | 8 | 9 | 52 | 6 | 8.67 | 9 | 7:00:00 | 5:00:00 | | | 9 | | | |
| 21-Abr | 22-Abr | A21 | Falta de generación de documentos de proceso Y | Z4 | | 9 | | 9 | | 9 | | 9 | | | 36 | 4 | 9 | 9 | 4:30:00 | 16:00:00 | 9 | | | | | |
| 21-Abr | 22-Abr | A22 | Falta de direccionamiento de documentos por funcionalidad incorrecta de pantalla | Z4 | | 8 | | 8 | 8 | 8 | 8 | | 8 | | 48 | 6 | 8 | | 7:30:00 | | 8 | | | | | |
| 23-Abr | 23-Abr | A23 | Error de impresión debido a corrección de defecto A15 | Z4 | | 8 | | 8 | | | 7 | 8 | 7 | | 38 | 5 | 7.6 | 8 | 1:00:00 | 3:00:00 | 8 | | | | | |
| 24-Abr | 24-Abr | A24 | Direccionamiento incorrecto de documentos Z para oficinas específicas | Z4 | | 8 | | 8 | 7 | 8 | 8 | | 8 | | 47 | 6 | 7.83 | 8 | 9:40:00 | | 8 | | | | | |
| 25-Abr | 25-Abr | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | 8 | | 8 | | 8 | 8 | | | 8 | 40 | 5 | 8 | 8 | 1:00:00 | 1:00:00 | | | | | | 8 |
| 25-Abr | 25-Abr | A25 | Falta de generación de documentos S | Z4 | | | 8 | 8 | | 7 | 7 | 8 | 7 | 8 | 53 | 7 | 7.57 | 8 | 7:30:00 | 1:00:00 | 8 | | | | | |
| 28-Abr | 28-Abr | A26 | Lentitud en tiempo de respuesta de generación de reporte | Z4 | | 8 | | 8 | | | 7 | 8 | 7 | | 38 | 5 | 7.6 | 8 | 2:30:00 | | | | 8 | | | |
| 28-Abr | 28-Abr | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | 7 | 8 | | 7 | 7 | | | 50 | 7 | 7.14 | | 1:00:00 | | | 7 | | | | |
| TOTAL | | | | | | | | | | | | | | | | | | | 96:40:00 | | 5 | 3 | 2 | 3 | 1 | 4 |

de 160 hrs al mes = 60.25% de mto. correctivo

Figura 4.5. Matriz de recolección de defectos de Abril 2003.

## RECOLECCIÓN DE DEFECTOS
### (MAYO 2003)

| | | | | | PRODUCTO | | | | | | | | | | | | | | | | | ORIGEN DEL DEFECTO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA INICIAL | FECHA FINAL | CODIGO DE DEFECTO | DESCRIPCIÓN DEL DEFECTO | MÓDULO EN EL QUE SE PRESENTÓ EL DEFECTO | A | B | C | NIVEL DE AFECTACIÓN | REPROCESO | COSTO $ | COSTO HH | REPERCUTE OTRAS AREAS | PROGRAMACIÓN | RECAPACITACIÓN | EVALUACIÓN TOTAL | CUENTA EVALUACIONES | EVALUACIÓN PROMEDIO | PROGRAMADO ACUMULADO | HRS. EMPLEADAS EN SOLUCIÓN | HRS.DE AFECTACIÓN A PRODUCCIÓN | ANÁLISIS DE REQUERIMIENTOS | DISEÑO | DESARROLLO | PRUEBAS | DATOS MIGRADOS DE OTROS SISTEMAS | CAPACITACIÓN |
| 5-May | 27-May | A27 | Generación incorrecta de 14000 documentos | Z6 | | | 9 | 9 | | | 9 | | 9 | | 36 | 4 | 9 | 9 | 11:30:00 | 2:00:00 | | 9 | | | | |
| 06-May | 14-May | A12 | Falta de cambio de estatus de un dato en el proceso masivo X | Z1 | | | 8 | 9 | 8 | | 9 | 8 | 9 | | 51 | 6 | 8.5 | | 24:30:00 | 1:00:00 | 9 | | | | | |
| 06-May | 6-May | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 1:30:00 | 1:30:00 | | | | | | 8 |
| 6-May | 6-May | A3 | Interfaz automática X que no se ejecuta | Z1 | 7 | 7 | 7 | | 7 | | 7 | 7 | | | 42 | 6 | 7 | | 0:30:00 | | | 7 | | | | |
| 6-May | 28-May | A15 | Bloqueo (causa K) de impresión a nivel divisional | Z4 | 8 | 8 | 8 | 8 | | | 8 | 8 | 8 | 7 | 63 | 8 | 7.88 | | 11:15:00 | | | | | | | |
| 13-May | 13-May | A28 | Desplegado 0 en documento | Z4 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | | 7 | 7 | 63 | 9 | 7 | 7 | 1:15:00 | | | | | 7 | | |
| 14-May | 16-May | A24 | Direccionamiento incorrecto de documentos Z para oficinas específicas | Z4 | | 8 | | 8 | 7 | 8 | 8 | | 8 | | 47 | 6 | 7.83 | 8 | 5:20:00 | | | | | | | |
| 14-May | 15-May | A29 | Exclusión de un plan para el cargo de penalización | Z3 | 7 | 7 | 7 | 8 | | | 7 | | 7 | | 43 | 6 | 7.17 | | 4:00:00 | | 7 | | | | | |
| 15-May | 26-May | A22 | Falta de direccionamiento de documentos por funcionalidad incorrecta de pantalla | Z4 | | 8 | | 8 | 8 | 8 | 8 | | 8 | | 48 | 6 | 8 | | 14:00:00 | | 8 | | | | | |
| 15-May | 15-May | A30 | Generación incorrecta de documentos G en proceso g | Z4 | | 7 | | 7 | 7 | | 7 | | 7 | | 35 | 5 | 7 | | 1:30:00 | | 7 | | | | | |
| 16-May | 22-May | A31 | Orden incorrecto de documentos tipo E | Z4 | | 8 | | 8 | 8 | | 8 | | 8 | | 40 | 5 | 8 | | 18:30:00 | 3:00:00 | 8 | | | | | |
| 19-May | 20-May | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 4:30:00 | 2:30:00 | | | | | | 8 |
| 23-May | 26-May | A32 | Exclusión de ciertas oficinas para direccionamiento de documentos Z | Z4 | | 7 | | 8 | | 7 | 7 | | 7 | | 36 | 5 | 7.2 | | 5:00:00 | | 7 | | | | | |
| 26-May | 26-May | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 1:30:00 | 1:30:00 | | | | | | 8 |
| 26-May | 26-May | A33 | Fallo de funcionalidad por defecto en migración | Z2 | | | 7 | 7 | | 7 | 7 | 7 | 7 | | 42 | 6 | 7 | 7 | 1:30:00 | 1:30:00 | | | | | 7 | |
| 27-May | 28-May | A10 | Pérdida de paquete de impresión a recuperar para una entidad federativa | Z4 | | | | 8 | 8 | 8 | 8 | 8 | 8 | | 48 | 6 | 8 | 8 | 3:30:00 | 3:30:00 | | | | | | 8 |
| 29-May | 30-May | A34 | Recibos faltantes en tablas de auditoría | Z3 | | | | 7 | 7 | | 8 | 7 | 7 | | 36 | 5 | 7.2 | | 6:40:00 | 0:30:00 | | | | | 7 | |
| 29-May | 30-May | A7 | Direccionamiento incorrecto de documentos Z | Z4 | | 8 | | 7 | 8 | 7 | 8 | 7 | 7 | | 52 | 7 | 7.43 | | 8:15:00 | | 7 | | | | | |
| TOTAL | | | | | | | | | | | | | | | | | | | 124:45:00 | | 7 | 2 | 0 | 1 | 2 | 4 |

de 160 hrs al mes = 77.5% de mto. correctivo

Figura 4.6. Matriz de recolección de defectos de Mayo 2003.

A continuación, en las figuras 4.7, 4.8 y 4.9, se tienen las Tablas de RESUMEN DE RECOLECCIÓN para los meses de Marzo a Mayo del 2003 resultantes de las matrices de recolección anteriores.

**RESUMEN DE RECOLECCIÓN**

(MARZO 2003)

| | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| **ORIGEN DEL DEFECTO** | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | 9 | 45.00% | 4 | 100.00% | 3 | 33.33% | 2 | 28.57% |
| Diseño | 5 | 25.00% | 0 | 0.00% | 1 | 11.11% | 4 | 57.14% |
| Desarrollo | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Pruebas | 2 | 10.00% | 0 | 0.00% | 1 | 11.11% | 1 | 14.29% |
| Datos Migrados | 1 | 5.00% | 0 | 0.00% | 1 | 11.11% | 0 | 0.00% |
| Capacitación | 3 | 15.00% | 0 | 0.00% | 3 | 33.33% | 0 | 0.00% |
| **Totales** | **20** | **100.00%** | **4** | **100.00%** | **9** | **100.00%** | **7** | **100.00%** |

**Figura 4.7. Tabla Resumen de recolección de Marzo 2003.**

**RESUMEN DE RECOLECCIÓN**

(ABRIL 2003)

| | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| **ORIGEN DEL DEFECTO** | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | 5 | 27.78% | 1 | 33.33% | 4 | 36.36% | 0 | 0.00% |
| Diseño | 3 | 16.67% | 0 | 0.00% | 1 | 9.09% | 2 | 50.00% |
| Desarrollo | 2 | 11.11% | 1 | 33.33% | 1 | 9.09% | 0 | 0.00% |
| Pruebas | 3 | 16.67% | 0 | 0.00% | 1 | 9.09% | 2 | 50.00% |
| Datos Migrados | 1 | 5.56% | 0 | 0.00% | 1 | 9.09% | 0 | 0.00% |
| Capacitación | 4 | 22.22% | 1 | 33.33% | 3 | 27.27% | 0 | 0.00% |
| **Totales** | **18** | **100.00%** | **3** | **100.00%** | **11** | **100.00%** | **4** | **100.00%** |

**Figura 4.8. Tabla resumen de recolección de Abril 2003.**

**RESUMEN DE RECOLECCIÓN**

(MAYO 2003)

| | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| ORIGEN DEL DEFECTO | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | 7 | 43.75% | 1 | 50.00% | 2 | 33.33% | 4 | 50.00% |
| Diseño | 2 | 12.50% | 1 | 50.00% | 0 | 0.00% | 1 | 12.50% |
| Desarrollo | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Pruebas | 1 | 6.25% | 0 | 0.00% | 0 | 0.00% | 1 | 12.50% |
| Datos Migrados | 2 | 12.50% | 0 | 0.00% | 0 | 0.00% | 2 | 25.00% |
| Capacitación | 4 | 25.00% | 0 | 0.00% | 4 | 66.67% | 0 | 0.00% |
| **Totales** | **16** | **100.00%** | **2** | **100.00%** | **6** | **100.00%** | **8** | **100.00%** |

**Figura 4.9. Tabla resumen de recolección de Mayo 2003.**

Hasta este momento ya se tienen cubiertos los pasos 1 y 2 de la técnica de Aseguramiento de Calidad Estadística, así que se continuará con el paso 3 "mediante el principio de Pareto se aíslan las causas vitales".

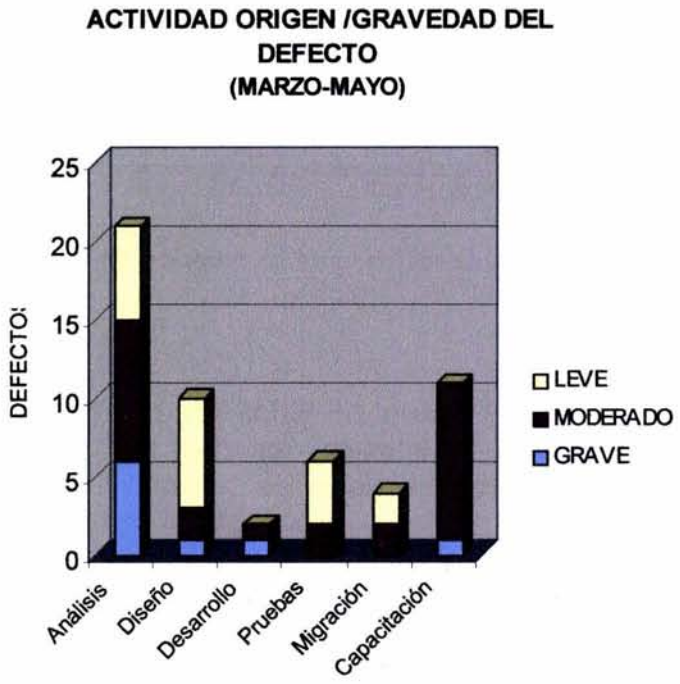
## 4.2.2. Identificación de las causas vitales.

Debido a que nuestra causa vital es "identificar la actividad del mantenimiento que genera más defectos graves" los datos de las tablas RESUMEN DE RECOLECCIÓN de los meses de Marzo a Mayo, se agruparon de dos maneras:

1. El total de defectos por GRAVEDAD para conocer el porcentaje de defectos GRAVES, MODERADOS Y LEVES que se habían generado en ese período de tiempo, lo cual dio como resultado la gráfica 4.10.


**Gráfica 4.10. Defectos agrupados por gravedad (Marzo-Mayo).**

2. Los defectos por GRAVEDAD y por ACTIVIDAD EN LA QUE SE ORIGINARON generando como resultado la gráfica 4.11, en la que se puede observar claramente que la actividad del mantenimiento que generó más defectos GRAVES fue ANÁLISIS DE REQUERIMIENTOS.




**Gráfica 4.11. Defectos graves, moderados y leves originados en las diferentes actividades del mantenimiento (Marzo-Mayo).**

Una vez que se tiene identificada la **causa vital**, se continuará con el paso 4 de la técnica de Aseguramiento de Calidad Estadística, "se actúa para corregir los problemas que han producido los defectos".

## 4.2.3. Medidas correctivas.

Debido a que en la gráfica 4.11 se detectó que la actividad que generó más defectos graves durante el período de marzo a mayo fue la actividad de ANÁLISIS DE REQUERIMIENTOS y que de todo el tiempo laboral de ese mismo período de

tiempo se empleó un 71.7 % en mantenimiento correctivo (corrección de defectos) se decidió aplicar las siguientes medidas durante la actividad del ANÁLISIS:

1. ***Antes de realizar una corrección al sistema se debe analizar la definición original***, ya que muchas veces la presión del usuario, por obtener la corrección, provoca modificar algo que tiene su razón de ser y esta modificación innecesaria genera otros defectos. Ejemplo: a veces el usuario final del sistema es una persona diferente al usuario que definió los requerimientos, así que el usuario final detecta como "defecto" una funcionalidad del sistema que sí cumple con los requerimientos definidos.

2. Una vez que se identifica que el sistema sí necesita una corrección, es importante ***analizar todas las posibles soluciones*** que se le pueden dar para tomar la que impacte menos a los demás módulos del sistema que están relacionados con el defecto. Ejemplo: A veces, el cambio de formato de un campo impacta menos que agregar un parámetro a un procedimiento que es utilizado por otros 30 procedimientos y pantallas.

3. ***Analizar todo lo que es necesario corregir para que al defecto se le dé la solución definitiva***, es decir, realizar corrección a pantallas, al código de base de datos y a todos los registros dañados por el defecto, ya que a través de la recolección de defectos se identificó que muchas veces la solución que se le daba al defecto era parcial, lo que provocaba que en futuro se volviera a presentar. Ejemplo: el usuario reporta "un calculo incorrecto en el registro X", si el personal de mantenimiento se enfoca sólo en corregir el registro X, seguramente en un futuro el usuario reportará el mismo cálculo incorrecto en otro registro, ocasionando malestar tanto en el usuario como en el personal.

Los 3 puntos anteriores deben hacerse en la medida que la prioridad para la corrección del defecto lo permita, y en los casos que no sea posible seguir todas estas medidas, se debe tomar nota de lo que faltó por corregirse para hacerla en un momento más APROPIADO, pero que no se olvide.

### 4.2.3.1. Matrices y tablas resultantes de la recolección de defectos del segundo trimestre.

Después de plantear las medidas correctivas a aplicar se siguieron recolectando los defectos de Junio a Agosto, en las figuras 4.12, 4.13 y 4.14 se presentan las matrices de RECOLECCIÓN DE DEFECTOS resultantes para ese período de tiempo y en las figuras 4.15, 4.16 y 4.17 se presentan las tablas RESUMEN DE RECOLECCIÓN del mismo período.

## RECOLECCIÓN DE DEFECTOS
### (JUNIO 2003)

| | | | | | PRODUCTO | | | | | | | | | | | | | | | | ORIGEN DEL DEFECTO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA INICIAL | FECHA FINAL | CÓDIGO DE DEFECTO | DESCRIPCIÓN DEL DEFECTO | MÓDULO EN EL QUE SE PRESENTÓ EL DEFECTO | > | B | C | NIVEL DE AFECTACIÓN | REPROCESO | COSTO $ | COSTO HH | REPERCUTE OTRAS AREAS | PROGRAMACIÓN | RECAPACITACIÓN | EVALUACIÓN TOTAL | CUENTA EVALUACIONES | EVALUACIÓN PROMEDIO | PORCENTAJE EFICIENCIA | HRS. EMPLEADAS EN SOLUCIÓN | HRS. DE AFECTACIÓN A PRODUCCIÓN | ANÁLISIS DE REQUERIMIENTOS | DISEÑO | DESARROLLO | PRUEBAS | DATOS MIGRADOS DE OTROS SISTEMAS | CAPACITACIÓN |
| 2-Jun | 2-Jun | A7 | Direccionamiento incorrecto de documentos Z | Z4 | | 8 | | 7 | | 7 | 8 | 7 | 7 | | 52 | 7 | 7.43 | | 1:00:00 | | | | | | | |
| 02-Jun | 02-Jun | A34 | Registros "a" faltantes en tablas de auditoría | Z3 | | | | 7 | 7 | | 8 | 7 | 7 | | 36 | 5 | 7.2 | | 4:30:00 | | | | | | | |
| 02-Jun | 3-Jun | A35 | Falta de generación de documentos a ser impresos | Z4 | 7 | 7 | 7 | 8 | 7 | | 7 | 7 | 7 | 7 | 64 | 9 | 7.11 | | 5:00:00 | | 7 | | | | | |
| 3-Jun | 3-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 1:30:00 | 1:00:00 | | | | | | 8 |
| 3-Jun | 9-Jun | A36 | Falta de cambio de estatus de registros de proc. AA en proceso masivo J | Z3 | 8 | 8 | 8 | 8 | 8 | 7 | 8 | 8 | 7 | | 70 | 9 | 7.78 | | 11:00:00 | | 8 | | | | | |
| 3-Jun | 3-Jun | A12 | Cambio de estatus de un dato en el proceso masivo X | Z1 | | | 8 | 9 | 8 | | 9 | 9 | 9 | | 51 | 6 | 8.5 | | 13:00:00 | | 9 | | | | | |
| 05-Jun | 09-Jun | A15 | Bloqueo (causa K) de impresión a nivel divisional | Z4 | 8 | 8 | 8 | 8 | | | 8 | 8 | 8 | 7 | 63 | 8 | 7.88 | | 5:00:00 | | | | | | | |
| 9-Jun | 9-Jun | A7 | Direccionamiento incorrecto de documentos Z | Z4 | | 8 | | 7 | | 7 | 8 | 7 | 7 | | 44 | 6 | 7.33 | | 3:00:00 | | 7 | | | | | |
| 9-Jun | 10-Jun | A37 | Documentos que incorrectamente se direccionaron mal de impresión | Z4 | | | | 8 | | 7 | 7 | 7 | 7 | | 36 | 5 | 7.2 | | 9:30:00 | | | | | 7 | | |
| 10-Jun | 10-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 2:30:00 | 2:00:00 | | | | | | 8 |
| 11-Jun | 11-Jun | A7 | Direccionamiento incorrecto de documentos Z | Z4 | | 8 | | 7 | | 7 | 8 | 7 | 7 | | 44 | 6 | 7.33 | | 2:00:00 | 2:00:00 | 7 | | | | | |
| 16-Jun | 16-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 2:00:00 | 2:00:00 | | | | | | 8 |
| 17-Jun | 17-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 3:30:00 | 3:30:00 | | | | | | 8 |
| 17-Jun | 17-Jun | A38 | Falta de información en reporte mensual | Z4 | | 7 | | 7 | | | 7 | | 7 | | 28 | 4 | 7 | | 1:30:00 | | | | | 7 | | |
| 19-Jun | 19-Jun | A12 | Cambio de estatus de un dato en el proceso masivo X | Z1 | | | 8 | 8 | 8 | | 8 | 8 | 8 | | 48 | 6 | 8 | | 6:00:00 | | 8 | | | | | |
| 19-Jun | 19-Jun | A39 | Documentos que quedan pendientes de impresión hasta después de un mes | Z4 | | 8 | | 8 | | | 8 | 8 | 8 | | 40 | 5 | 8 | 8 | 2:30:00 | | | | | 8 | | |
| 23-Jun | 23-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 3:30:00 | 3:30:00 | | | | | | 8 |
| 23-Jun | 23-Jun | A40 | Problemas de módulo Z3 debido a liberación incorrecta al cliente | Z3 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | | 63 | 9 | 7 | | 1:00:00 | 1:00:00 | 7 | | | | | |
| 24-Jun | 24-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 3:30:00 | 2:00:00 | | | | | | 8 |
| 25-Jun | 25-Jun | A41 | Falta de cuadre de información entre reportes | Z4 | | 7 | | 7 | | | 7 | | 7 | 7 | 35 | 5 | 7 | | 2:30:00 | 2:30:00 | | | | | | 7 |
| 26-Jun | 27-Jun | A42 | Datos incorrectos de fin de período vs inicio de período | Z3 | 8 | | | 8 | 8 | | 8 | | 8 | | 40 | 5 | 8 | | 7:00:00 | 0:30:00 | | | | | 8 | |
| 27-Jun | 27-Jun | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 5:00:00 | 3:00:00 | | | | | | 8 |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TOTAL | | | | | | | | | | | | | | | | | | | 96:00:00 | | 7 | 0 | 0 | 3 | 1 | 8 |

de 160 hrs al mes = 60% de mto. correctivo

Figura 4.12. Matriz de recolección de defectos de Junio 2003.

## RECOLECCIÓN DE DEFECTOS
(JULIO 2003)

| | | | | | PRODUCTO | | | | | | | | | | | | | | | ORIGEN DEL DEFECTO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA INICIAL | FECHA FINAL | CODIGO DE DEFECTO | DESCRIPCIÓN DEL DEFECTO | MÓDULO EN EL QUE SE PRESENTÓ EL DEFECTO | A | B | C | NIVEL DE AFECTACIÓN | REPROCESO | COSTO $ | COSTO HH | REPERCUTE OTRAS AREAS | PROGRAMACIÓN | RECAPACITACIÓN | EVALUACIÓN TOTAL | CUENTA EVALUACIONES | EVALUACIÓN PROMEDIO | PRIORIDAD JERÁRQUICA | HRS. EMPLEADAS EN SOLUCIÓN | HRS. DE AFECTACIÓN A PRODUCCIÓN | ANÁLISIS DE REQUERIMIENTOS | DISEÑO | DESARROLLO | PRUEBAS | DATOS MIGRADOS DE OTROS SISTEMAS | CAPACITACIÓN |
| 1-Jul | 1-Jul | A43 | Problemas de módulo Z4 por comunicación | Z4 | 8 | 8 | 8 | 8 | | | 8 | 8 | | | 48 | 6 | 8 | 8 | 1:30:00 | 1:30:00 | | ∞ | | | | |
| 01-Jul | 01-Jul | A44 | Documento incorrecto de un producto en otro producto | Z4 | | 7 | | 7 | | 7 | 7 | | 7 | | 35 | 5 | 7 | | 3:00:00 | 1:00:00 | | | | 7 | | |
| 02-Jul | 03-Jul | A45 | Reproceso de proceso M | Z4 | 7 | 7 | | 7 | 7 | | 8 | | 8 | 7 | 51 | 7 | 7.29 | 7 | 6:00:00 | | 7 | | | | | |
| 02-Jul | 09-Jul | A46 | Registro resultante de proceso incorrecto x | Z1 | | | 9 | 9 | | | 9 | | 9 | | 36 | 4 | 9 | | 18:30:00 | 4:00:00 | | | | 9 | | |
| 3-Jul | 4-Jul | A24 | Direccionamiento incorrecto de documentos Z para oficinas específicas | Z4 | | 8 | | 8 | 7 | 8 | 8 | | 8 | | 47 | 6 | 7.83 | 8 | 3:45:00 | | 8 | | | | | |
| 4-Jul | 7-Jul | A47 | Generación 2 primeros doctos. | Z4 | | 7 | | 7 | | 7 | 7 | 7 | 7 | | 42 | 6 | 7 | | 5:30:00 | | 7 | | | | | |
| 14-Jul | 14-Jul | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 3:00:00 | 1:00:00 | | | | | | 8 |
| 18-Jul | 28-Jul | A48 | Problemas de módulo Z7 | Z7 | 7 | 7 | 7 | 7 | 7 | | 7 | | | | 42 | 6 | 7 | | 1:30:00 | | | | | | 7 | |
| 18-Jul | 28-Jul | A49 | Problemas de módulo Z3 | Z3 | 8 | | | 9 | 9 | | 9 | 9 | | 9 | 53 | 6 | 8.83 | | 25:45:00 | 4:00:00 | | | | 9 | | |
| 21-Jul | 28-Jul | A50 | Operación incompleta de operación de una pantalla | Z8 | 7 | 7 | | 8 | 7 | 7 | 8 | 7 | | 8 | 59 | 8 | 7.38 | 7 | 6:40:00 | 1:00:00 | | | | | | 7 |
| 22-Jul | 22-Jul | A51 | Error de resolución del mismo documento al mismo tiempo | Z4 | 8 | 8 | 8 | 8 | | | 8 | | 7 | | 47 | 6 | 7.83 | | 5:30:00 | | | | | 8 | | |
| 24-Jul | 24-Jul | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 3:30:00 | 1:00:00 | | | | | | 8 |
| 29-Jul | 31-Jul | A52 | Los cargos a realizar en módulo Z3 deben tomarse de un catalogo | Z3 | 7 | | | 7 | | 7 | 8 | 7 | 7 | | 43 | 6 | 7.17 | 7 | 10:15:00 | | 7 | | | | | |
| 31-Jul | 31-Jul | A53 | Documentos BB no deben imprimirse en linea | Z4 | | 7 | | 7 | | 8 | 7 | | 7 | | 36 | 5 | 7.2 | | 2:00:00 | | | | | 7 | | |
| 31-Jul | 31-Jul | A54 | En proceso Z4 faltan documentos de pasarlos de un lugar a otro | Z4 | | 8 | | 8 | 7 | 7 | 7 | 7 | 7 | | 51 | 7 | 7.29 | | 1:30:00 | | | | | 7 | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TOTAL | | | | | | | | | | | | | | | | | | | 97:55:00 | | 4 | 1 | 0 | 6 | 1 | 3 |

de 160 hrs al mes = 61% de mto. correctivo

Figura 4.13. Matriz de recolección de defectos de Julio 2003.

## RECOLECCIÓN DE DEFECTOS
### (AGOSTO 2003)

| | | | | | PRODUCTO | | | | | | | | | | | | | | | ORIGEN DEL DEFECTO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA INICIAL | FECHA FINAL | CODIGO DE DEFECTO | DESCRIPCIÓN DEL DEFECTO | MÓDULO EN EL QUE SE PRESENTÓ EL DEFECTO | A | B | C | NIVEL DE AFECTACIÓN | REPROCESO | COSTO $ | COSTO HH | REPERCUTE OTRAS AREAS | PROGRAMACIÓN | RECAPACITACIÓN | EVALUACIÓN TOTAL | CUENTA EVALUACIONES | EVALUACIÓN PROMEDIO | PRIORIDAD JERÁRQUICA | HRS. EMPLEADAS EN SOLUCIÓN | HRS DE AFECTACIÓN A PRODUCCIÓN | ANÁLISIS DE REQUERIMIENTOS | DISEÑO | DESARROLLO | PRUEBAS | DATOS MIGRADOS DE OTROS SISTEMAS | CAPACITACIÓN |
| 1-Ago | 1-Ago | A55 | Documentos que se imprimen en oficina correcta pero se ordenan incorrectamente | Z4 | | 7 | | 7 | | | 7 | | | 7 | 28 | 4 | 7 | | 1:30:00 | | | | | | | 7 |
| 01-Ago | 22-Ago | A54 | Documentos que el CC deja incorrectamente | Z4 | 8 | 8 | | 8 | 8 | 8 | 7 | 8 | 7 | | 62 | 8 | 7.75 | 8 | 2:20:00 | 0:30:00 | | | | | | |
| 04-Ago | 04-Ago | A57 | Problemas de adecuación para desplegado de impresoras en XP en v6i | Z4 | 9 | 9 | 9 | 9 | | | 7 | 9 | 7 | | 59 | 7 | 8.43 | 9 | 4:15:00 | 4:15:00 | | | | 9 | | |
| 05-Ago | 05-Ago | A58 | Registros XX que pasaron de listas impresión a impresas sin sello | Z4 | | 7 | | 7 | | | 7 | | | 7 | 28 | 4 | 7 | | 0:40:00 | | | | | | | 7 |
| 7-Ago | 7-Ago | A59 | Registros AA pendientes en cero de módulo Z2 de registros XX en estatus Y | Z2 | 7 | 7 | | 7 | 7 | | 7 | 7 | 7 | | 49 | 7 | 7 | | 1:30:00 | 0:30:00 | | | 7 | | | |
| 7-Ago | 7-Ago | A60 | Falta de cancelación de registros XX | Z3 | 7 | | | 7 | | | 7 | | 7 | 7 | 35 | 5 | 7 | | 2:00:00 | | | | | | | 7 |
| 07-Ago | 22-Ago | A61 | Falta de cancelación de registros XX | Z3 | | 7 | | 7 | 8 | | 8 | 8 | 8 | | 46 | 6 | 7.67 | | 20:00:00 | 1:30:00 | | | | | 8 | |
| 14-Ago | 14-Ago | A52 | Los cargos a realizar en módulo Z3 deben tomarse de un catalogo | Z3 | 7 | | | 7 | | 7 | 8 | 7 | 7 | | 43 | 6 | 7.17 | | 3:00:00 | | | | | | | |
| 14-Ago | 14-Ago | A62 | Realización de cambio de categoría de proc. AA en proceso x | Z1 | | | 7 | 7 | 7 | | 8 | 7 | 8 | | 44 | 6 | 7.33 | | 6:00:00 | 0:30:00 | | | | 7 | | |
| 18-Ago | 18-Ago | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 6:45:00 | 6:45:00 | | | | | | 8 |
| 19-Ago | 21-Ago | A63 | Registros XX no pasaron por módulo Z2 | Z2 | | 8 | | 8 | 7 | 8 | 8 | 8 | 8 | | 55 | 7 | 7.86 | | 4:00:00 | | | | | | 8 | |
| 19-Ago | 28-Ago | A64 | Problema de impresión en XP | Z4 | 9 | 9 | 9 | 9 | | 8 | 9 | | 8 | | 61 | 7 | 8.71 | 9 | 34:30:00 | 25:00:00 | | | | 9 | | |
| 20-Ago | 20-Ago | A10 | Pérdida de paquete de impresión a recuperar | Z4 | | | | 8 | | 8 | 8 | 8 | | 8 | 40 | 5 | 8 | 8 | 1:15:00 | 1:15:00 | | | | | | 8 |
| 21-Ago | 21-Ago | A65 | Registros AA de más en reporte de producto B | Z4 | | 7 | | 7 | | 7 | 7 | | 7 | | 35 | 5 | 7 | | | 1:45:00 | 7 | | | | | |
| 21-Ago | 21-Ago | A66 | doctos. VV que se iban a oficina int. incorrectamente | Z4 | | 7 | | 7 | 7 | | 7 | 7 | 7 | | 42 | 6 | 7 | | 1:00:00 | 0:30:00 | | | | 7 | | |
| 27-Ago | 27-Ago | A67 | Plant. con acento que generó problemas en Int. | Z4 | 7 | | | 7 | 7 | | 7 | 7 | 7 | | 42 | 6 | 7 | | 1:00:00 | 1:00:00 | | | | 7 | | |
| 28-Ago | 28-Ago | A68 | Prob. generación de archivos en int. | Z4 | 8 | 8 | 8 | 8 | | 7 | 8 | 8 | | 8 | 63 | 8 | 7.88 | 8 | 4:45:00 | 4:45:00 | | | | | | 8 |
| 29-Ago | 29-Ago | A69 | Prob. por adecuación de cargo de catalogo | Z3 | 7 | 7 | | 8 | | | 7 | 8 | 7 | | 44 | 6 | 7.33 | 8 | 2:00:00 | 2:00:00 | | | | 7 | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TOTAL | | | | | | | | | | | | | | | | | | | 96:30:00 | | 1 | 0 | 1 | 6 | 2 | 6 |

de 160 hrs al mes = 60% de mto. correctivo

Figura 4.14. Matriz de recolección de defectos de Agosto 2003.

**RESUMEN DE RECOLECCIÓN**

(JUNIO 2003)

| ORIGEN DEL DEFECTO | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | 7 | 36.84% | 1 | 100.00% | 2 | 18.18% | 4 | 57.14% |
| Diseño | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Desarrollo | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Pruebas | 3 | 15.79% | 0 | 0.00% | 1 | 9.09% | 2 | 28.57% |
| Datos Migrados | 1 | 5.26% | 0 | 0.00% | 1 | 9.09% | 0 | 0.00% |
| Capacitación | 8 | 42.11% | 0 | 0.00% | 7 | 63.64% | 1 | 14.29% |
| **Totales** | **19** | **100.00%** | **1** | **100.00%** | **11** | **100.00%** | **7** | **100.00%** |

**Figura 4.15. Tabla resumen de recolección de Junio de 2003.**

**RESUMEN DE RECOLECCIÓN**

(JULIO 2003)

| ORIGEN DEL DEFECTO | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | 4 | 26.67% | 0 | 0.00% | 1 | 20.00% | 3 | 37.50% |
| Diseño | 1 | 6.67% | 0 | 0.00% | 1 | 20.00% | 0 | 0.00% |
| Desarrollo | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Pruebas | 6 | 40.00% | 2 | 100.00% | 1 | 20.00% | 3 | 37.50% |
| Datos Migrados | 1 | 6.67% | 0 | 0.00% | 0 | 0.00% | 1 | 12.50% |
| Capacitación | 3 | 20.00% | 0 | 0.00% | 2 | 40.00% | 1 | 12.50% |
| **Totales** | **15** | **100.00%** | **2** | **100.00%** | **5** | **100.00%** | **8** | **100.00%** |

**Figura 4.16. Tabla resumen de recolección de Julio de 2003.**

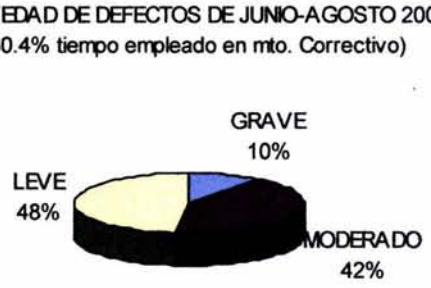
**RESUMEN DE RECOLECCIÓN**

(AGOSTO 2003)

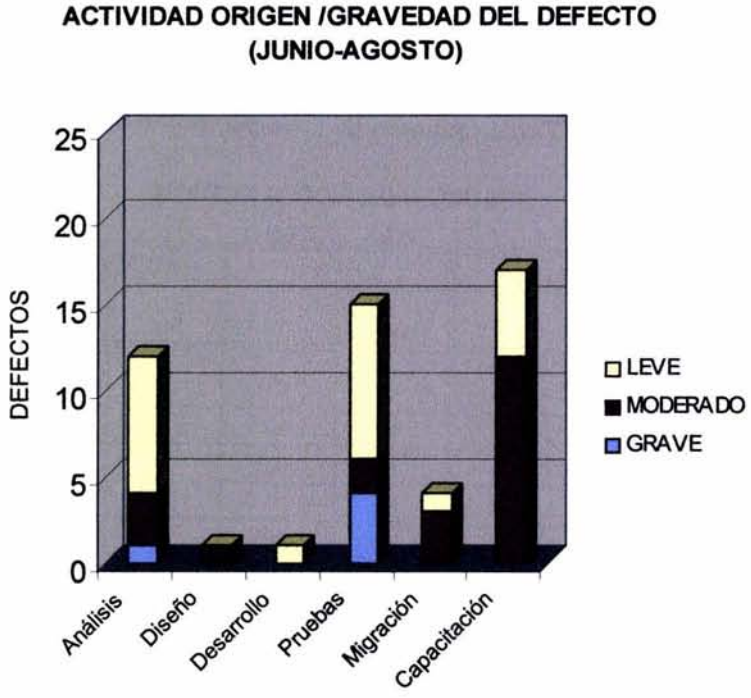
| ORIGEN DEL DEFECTO | TOTAL | | GRAVE | | MODERADO | | LEVE | |
|---|---|---|---|---|---|---|---|---|
| | No. | % | No. | % | No. | % | No. | % |
| Análisis de Requerimientos | 1 | 6.25% | 0 | 0.00% | 0 | 0.00% | 1 | 11.11% |
| Diseño | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% | 0 | 0.00% |
| Desarrollo | 1 | 6.25% | 0 | 0.00% | 0 | 0.00% | 1 | 11.11% |
| Pruebas | 6 | 37.50% | 2 | 100.00% | 0 | 0.00% | 4 | 44.44% |
| Datos Migrados | 2 | 12.50% | 0 | 0.00% | 2 | 40.00% | 0 | 0.00% |
| Capacitación | 6 | 37.50% | 0 | 0.00% | 3 | 60.00% | 3 | 33.33% |
| **Totales** | **16** | **100.00%** | **2** | **100.00%** | **5** | **100.00%** | **9** | **100.00%** |

**Figura 4.17. Tabla resumen de recolección de Agosto de 2003.**

Los datos de las tablas RESUMEN DE RECOLECCIÓN de los segundos tres meses de recolección de defectos se agruparon de las dos maneras antes descritas, la primera representa los defectos por gravedad, lo cual dio como resultado la gráfica 4.18 en la que se ve una disminución en el porcentaje de defectos GRAVES y en el porcentaje de tiempo laboral empleado en mantenimiento correctivo, lo anterior con respecto al período de Marzo a Mayo; y la segunda que representa los defectos por GRAVEDAD y por ACTIVIDAD EN LA QUE SE ORIGINARON dando como resultado la gráfica 4.19, en la que se puede ver claramente que, comparándola con la gráfica 4.11, disminuyó el número de defectos GRAVES generados en la actividad del mantenimiento de ANÁLISIS DE REQUERIMIENTOS.




**Gráfica 4.18. Defectos agrupados por gravedad (Junio-Agosto).**




**Gráfica 4.19. Defectos graves, moderados y leves originados en las diferentes actividades del mantenimiento (Junio-Agosto).**

## 4.2.4. Interpretación de los resultados.

A continuación se presenta la interpretación de los resultados obtenidos con el uso de la Técnica de Aseguramiento de Calidad Estadística en la fase de mantenimiento del software PT1.

- Realmente es sorprendente el tiempo que se estaba empleando en el mantenimiento correctivo del software, hasta antes de aplicar la técnica se tenía idea de que era mucho pero no se tenía una medida de ello, con la aplicación de la técnica se pudo conocer el porcentaje del tiempo invertido en mantenimiento correctivo y se logró disminuir de un 71.7% a un 60.4%.

- Con la aplicación de la técnica y la asignación del CÓDIGO DE DEFECTO en la matriz de RECOLECCIÓN DE DEFECTOS, se concluye que las soluciones que se le estaban dando a algunos defectos eran incompletas o incorrectas ya que éstos defectos se presentaban varias veces en el mismo mes o en meses posteriores.

- Con la aplicación de la técnica durante el período de Marzo a Mayo la actividad del mantenimiento que hasta ese momento estaba generando la mayoría de los defectos GRAVES era el ANÁLISIS DE REQUERIMIENTOS. Después de la aplicación de las medidas para mejorar el análisis previo a la corrección del defecto, el número de defectos GRAVES originados por ANÁLISIS DE REQUERIMIENTOS disminuyó.

- El porcentaje de defectos GRAVES en general, el cual durante el período de Marzo a Mayo fue de un 17% (gráfica 4.10), disminuyó a un 10% durante el período de Junio a Agosto (gráfica 4.18).

- Con la aplicación de las medidas correctivas para mejorar la actividad de ANÁLISIS DE REQUERIMIENTOS, específicamente la tercera (Analizar todo lo que es necesario corregir para que al defecto se le dé la solución definitiva), algunos defectos que se venían repitiendo mes con mes finalmente quedaron solucionados, tales como: los defectos A3 y A22 que dejaron de presentarse en el mes de Junio y los defectos A7 y A12 que dejaron de presentarse en el mes de Julio.

- El módulo con más defectos fue el Z4, la mayoría de los defectos de este modulo fueron MODERADOS y generados por una CAPACITACIÓN incorrecta; desafortunadamente durante el tiempo que duró este trabajo no se pudo planear una solución viable para este problema pero lo importante es que se tiene identificado y en cuanto se pueda se abordará.

En este capítulo se puede concluir que el Mantenimiento es la fase del ciclo de vida de un software en la que se detectan todos los defectos si éste no fue analizado, diseñado, codificado y probado con calidad.

De acuerdo a la interpretación de los resultados obtenidos, se puede concluir que la aplicación de la técnica de Aseguramiento de Calidad Estadística durante la fase de mantenimiento mejoró la calidad del software PT1 debido a que:

a) Algunos defectos que se presentaban continuamente se corrigieron definitivamente,

b) El porcentaje de los defectos que tienen más impacto en el software, que son los GRAVES, disminuye, y

c) El tiempo que el personal de mantenimiento le dedica a la corrección de defectos disminuye permitiéndole dedicarse a otras actividades.

Los resultados de la técnica hacen reflexionar acerca de todos los errores que se podrían depurar antes de convertirse en defectos si la técnica se aplicara durante las fases de análisis, diseño, codificación y pruebas del desarrollo de un software.

# CONCLUSIONES Y RECOMENDACIONES

- Cuando empecé el trabajo de investigación me di cuenta que en las librerías de la Ciudad de México encontramos cientos de libros que nos proporcionan, a los que nos dedicamos a construir software, diferentes instrumentos: manejadores de bases de datos, lenguajes de programación, plataformas de red, etc., pero es muy difícil encontrar un libro que nos indique cómo emplear todas esas herramientas para construir software de calidad, así que depende totalmente de nuestro interés en garantizar la calidad del software a nuestros clientes para buscar indicadores que nos permitan garantizarlo cuantitativamente.

- Estando en la posición del *desarrollador del software* no le hemos dado la importancia a asegurar a nuestros clientes la calidad de un software de forma cuantitativa, esto puede ser debido a que el desempeño diario de actividades, con el tiempo, se convierte en una labor rutinaria y nos volvemos indiferentes para asegurar la calidad, pero pueden estar seguros que si nos toca estar en la posición del *usuario de un software* de mala calidad por ejemplo: el tener que abandonar la fila de un banco debido a la caída del sistema después de haber estado formado en ella por una hora; o que en el estado de cuenta de la tarjeta de crédito aparezcan cargos que no hayamos realizado, entonces sí le daremos importancia al término "Software de Calidad".

- Es recomendable que los datos a recolectar para realizar una medición estén relacionados con lo que se desea medir. En nuestro caso, debido a que el objetivo de la aplicación de la técnica era conocer la gravedad de los defectos para poder enfocar los esfuerzos del personal en la corrección de los defectos GRAVES, los datos evaluables y no evaluables de la matriz diseñada fueron totalmente enfocados para medir el impacto de los defectos en el esfuerzo, costo y afectación en la operación.

- Para la aplicación de la técnica de Aseguramiento de Calidad Estadística fue necesario realizar tareas laborales adicionales a las cargas de trabajo diarias (diseño de formatos, establecimiento del método para el llenado y la recolección de datos) pero créanme que valió la pena.

- En el área de sistemas nos gusta ser reconocidos como héroes cuando resolvemos problemas graves, lo que nos da como resultado creer que prevenir esos problemas a tiempo nos hará pasar inadvertidos para el resto del personal; con la realización de este trabajo me he dado cuenta que hay que cambiar esta forma de pensar porque sino nos colocamos en una posición en la que no podemos crecer profesionalmente cuando haya la oportunidad ya que siempre seremos indispensables para "salvar el sistema".

- Con la aplicación de la técnica de Aseguramiento de Calidad Estadística se demostró que no es necesario contar con personal altamente capacitado en las empresas para practicar la medición del software.

- Se confirmó la importancia de la medición del software planteada en la sección 2.2 de este trabajo, ya que la aplicación de la técnica de Aseguramiento de Calidad Estadística nos ayudó a conocer cuantitativamente la situación de la calidad actual del software lo que nos permitirá planear qué es lo que se desea mejorar en el futuro.

- La técnica de Aseguramiento de Calidad Estadística permite establecer programas de medición de software dado que los datos recolectados en un proyecto servirán como base histórica para proyectos futuros y permitirán saber en qué fases del ciclo de vida del software es necesario poner más atención.

- Espero que este trabajo haya servido para que el lector se dé cuenta que hace varios años ha habido el interés de medir la calidad del software y que en México, a veces, ni siquiera tenemos conocimiento de los métodos y las herramientas que han sido desarrolladas con este objetivo, de tal manera que nazca en el desarrollador de software un interés para investigar más acerca sobre este tema y la aplicación en su trabajo diario.

# BIBLIOGRAFÍA.

[BRIA2002] Briand Lionel, Morasca Sandro, Basili Victor, "*An Operational Process for Goal-Driven Definition of Measures*", IEEE Transactions on Software Engineering, vol. 28, No. 12, pp. 1106-1125, December 2002.

[CARD2003] *Managing Software Quality With Defects* [en línea]. David N. Card, Software Productivity Consortium, Mar 2003, [citado 30 de Agosto 2003 09:41]. Disponible en world wide web: http://www.stsc.hill.af.mil/CrossTalk/2003/03/card.html

[CUEV1993] Cuevas Agustín Gonzalo, "*Ingeniería del Software: práctica de la programación*", Addison –Wesley Iberoamericana, 1993.

[DEL] "*Importancia de asimilar el concepto de calidad y beneficios de implementar un sistema de gestión de la calidad en la empresa" [en línea]*, De la Cruz Bovea César A.,[citado 21 Marzo 2003], disponible en world wide web: http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/calidadcesaraugusto.htm.

[FENT1994] Fenton Norman, "*Software Measurement: A Necessary Scientific Basis"*, IEEE Transactions on Software Engineering, vol. 20, No. 3, pp. 199-206, March 1994.

[GES2003] *Evolución Histórica del concepto de calidad* [en línea]. José Luis Arroyo [citado 21 Marzo 2003]. Disponible en world wide web : http://www.getiopolis.com/

[GOPA2002] Gopal Anandasivam, Krishnan M., Mukhopadhyay Tridas, Goldenson Dennis, *"Measurement Programs in Software Development: Determinants of Success"*, IEEE Transactions on Software Engineering, vol. 28, No. 9, pp. 863-875, September 2002.

[KITC2001] Kitchenham Barbara, Hughes Robert, Linkman Stephen, *"Modeling Software Measurement Data"*, IEEE Transactions on Software Engineering, vol. 27, No. 9, pp. 788-804, September 2001.

[PIAT2000] Piattini Mario G., Calvo-Manzano José A., Cervera Joaquín, Fernández Luis, *"Análisis y diseño detallado de Aplicaciones de Gestión"*, Alfaomega Grupo Editor, México, 2000.

[PRES2002] Pressman Roger S., "*Ingeniería del Software, un enfoque práctico"*, Editorial McGraw-Hill, México, 2002.

[SOMM2002] Sommerville Ian, *"Ingeniería de Software"*, Editorial Pearson Educación, México, 2002