

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/273380176>

Crterios de Calidad en el desarrollo de Software de tipo Web

Book · January 2011

CITATIONS

0

READS

3,358

4 authors, including:



[Rene Edmundo Cuevas Valencia](#)

Universidad Aut3noma de Guerrero

85 PUBLICATIONS 22 CITATIONS

[SEE PROFILE](#)



[Feliciano Morales Severino](#)

Universidad Aut3noma de Guerrero

13 PUBLICATIONS 74 CITATIONS

[SEE PROFILE](#)



[Felix Molina Angel](#)

Universidad Aut3noma de Guerrero

3 PUBLICATIONS 0 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:




Proyecto de Titulaci3n de tesis [View project](#)



Resultados de CA TIC [View project](#)

Criterios de Calidad en el desarrollo de Software de tipo Web



FELICIANO MORALES SEVERINO
MEDINA MARTÍNEZ JUAN CARLOS
CUEVAS VALENCIA RENÉ EDMUNDO
MOLINA ÁNGEL FÉLIX



Criterios de Calidad en el desarrollo de Software de tipo Web

***Severino Feliciano Morales
Juan Carlos Medina Martínez
René Edmundo Cuevas Valencia
Félix Molina Ángel***

ISBN: 978-607-00-3903-4



CONTENIDO:

PRÓLOGO	1
CAPÍTULO I. INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE.	4
1.1 Aspectos Generales de Software	5
1.1.1 Definición de Software	5
1.1.2 Clasificación del software	5
1.1.3 Proceso de creación de software	6
1.1.4 Página web	7
1.1.5 Aplicación Web	11
1.1.6 ¿En que consiste la Evaluación de la Calidad en los Productos de Software?	14
1.1.7 Antecedentes de la Evaluación de la Calidad en Aplicaciones Web	19
CAPÍTULO II. FUNDAMENTOS DE LA ADMINISTRACIÓN DE LA CALIDAD DE SOFTWARE.	21
2.1 Evaluación de la Calidad en el Desarrollo de Productos de Software	22
2.2 Administración de la Calidad	23
2.3 Evolución del Concepto de Administración de Calidad	24
2.3.1 El control estadístico de la calidad	24
2.4. Administración de Calidad Total (ACT)	25
2.5. Concepto de Calidad de Software	26
2.6. El Modelo de McCall	27
2.7. El Modelo de Boehm	27
CAPÍTULO III. ASPECTOS DE CALIDAD EN APLICACIONES WEB Y PROCESOS DE MEJORA CONTÍNUA	30
3.1. Criterios de Calidad	31
3.2. Aspectos de Calidad en Aplicaciones Web	32
3.2.1. Confiabilidad	34
3.2.2. Usabilidad	34
3.2.3. Seguridad	35
3.2.4. Disponibilidad	35
3.2.5. Escalabilidad	36
3.2.6 Mantenibilidad	36
3.4. Métricas de Software	37
3.5. Procesos de Mejora Continua	40
CAPÍTULO IV. METODOLOGÍA PARA LA EVALUACIÓN DE LOS ASPECTOS DE CALIDAD EN APLICACIONES WEB.	51
4.1. Introducción.	52
4.2. Aspectos importantes de la usabilidad.	52
4.3. Estándares internacionales	61
4.4. Laboratorio de usabilidad	63

4.5. Establecimiento de la Metodología	65
CAPÍTULO V. ANÁLISIS Y DISEÑO DE LA APLICACIÓN ECA-WEB.	
5.1 Planteamiento del sistema	69
5.2 Análisis de contenido	69
5.3 Análisis de interacción	70
5.4 Análisis funcional	70
5.5 Arquitectura del Sistema	71
5.6 Diseño de la aplicación Web	72
RESUMEN	78
REFERENCIAS	79

PRÓLOGO

Debido a su creciente importancia, las **aplicaciones web** demandan una alta calidad en su desarrollo y operación. Sin embargo, para lograr que un producto de software opere con calidad es muy complejo, ya que existen factores que son parte del proceso de desarrollo y otros que son parte del producto de software mismo y que afectan de manera directa a su operación.

Para el aseguramiento de la calidad de un producto de software, implica que se le deben realizar varias evaluaciones. Sin embargo, no existen estudios que hablen en forma específica de cómo se debe realizar el proceso de evaluación para un cierto atributo de calidad.

Cada uno de los atributos de calidad de un producto de software es un tema muy amplio, por lo que es difícil que se traten todos los aspectos en esta publicación.

Específicamente, trataremos el aspecto de la usabilidad, este atributo de calidad de software, se enfoca a mejorar la simplicidad, entendibilidad y facilidad de uso de un sistema de software para un cliente o usuario final.

De acuerdo a este enfoque es obvio que la calidad del producto se ve afectada por el proceso de desarrollo. Existe una gran dependencia de la calidad del producto de software con el cumplimiento de los requerimientos. En este aspecto, los procesos de mejora continua se restringen a frecuentes validaciones durante cada una de las fases.

A pesar del gran número de artículos de investigación y normas existentes sobre el tema de validación de calidad del producto de software, en la actualidad existen muy pocas empresas de Software que utilicen procesos de evaluación y análisis para este efecto. Actualmente, la gran mayoría de estudios están enfocados a las actividades de administración de los proyectos de desarrollo de software. En diversos entornos industriales y académicos, la calidad del software ha sido evaluada mediante distintos estudios analíticos.

Sin embargo, la evaluación de la calidad de software ha evolucionado hacia modelos formales estadísticos que se basan en métricas como fundamento para el aseguramiento, control y evaluación de la calidad de un producto o proceso de software. Grandes compañías como IBM, Hewlett Packard, Motorola y Siemens, entre otras, fundamentan su marco de producción de software con este enfoque estadístico, lo cual las ha convertido en pioneras de este campo.

Es evidente la importancia de la evaluación de un producto de software en un contexto real. Algunas industrias que desarrollan software, en la actualidad se limitan sólo a realizar pruebas del producto de software y a corregir los defectos que puedan ser localizados en dichas pruebas.

La falta de un análisis formal de los resultados, fomenta que en muchos de los casos este proceso resulte poco fiable y carente de aportaciones significativas. Además, no se aplican métricas de software durante el proceso de evaluación, lo cual produce una gran incertidumbre acerca de los atributos que se desean asegurar.

Este proceso resulta ser tan inmaduro que en la mayoría de las ocasiones que lejos de beneficiar al producto de software, este resulta afectado.

De acuerdo a esto, es importante contar con una metodología y una herramienta que permita realizar la evaluación y el análisis de los atributos de calidad en aplicaciones WEB. Esta metodología y esta herramienta, puede servir de modelo para las organizaciones que requieran realizar la validación de los atributos de calidad en sus aplicaciones de tipo WEB.

Se pretende presentar un estudio teórico práctico que permita describir una metodología y presentar el análisis y diseño de una herramienta, que sirvan como herramienta para mejorar la calidad de las aplicaciones WEB. Se puntualizarán los procesos para la validación y el control de los atributos de calidad, para mejorar el proceso de desarrollo de los productos de software, mediante el uso de procesos de mejora continua y el análisis y diseño de una herramienta que la denominamos ECA-WEB. Cabe mencionar que esta herramienta ha mejorado de acuerdo a la versión inicial, en virtud de que se hizo necesario agregarle algunos aspectos.

Al validar un producto de software de manera eficiente, se logra que este producto sea solvente y con más razón si se va ejecutar en red debe validarse correctamente para que no sea necesario un mantenimiento costoso.

Puesto que este libro, va enfocado específicamente a los productos de tipo WEB, quiere decir que esta metodología se puede aplicar a cualquiera de este tipo de aplicaciones, para asegurar que estas, sean confiables y por lo tanto aceptables ante la sociedad.

El objetivo principal es desarrollar una metodología y el análisis y diseño de una herramienta que permita realizar la evaluación y el análisis del atributo de usabilidad en aplicaciones WEB.

Esta metodología y esta herramienta, puede servir de modelo para las realizar la validación del atributo de usabilidad en aplicaciones de tipo WEB.

También se pretende, lograr que esta metodología sirva para llevar a cabo la validación de los atributos de calidad en los productos de software. En esta metodología se integrarán métricas de software para su análisis.

Se pretende además, que este libro sea un buen material de consulta, para que las personas que necesiten información de esta área pueda servirles de manera eficiente y para los que deseen hacer un trabajos de tesis pueda servirles de cimiento.

Agradecimientos

Expreso mi más sincero agradecimiento a **Juan Carlos Medina Martínez, René Edmundo Cuevas Valencia y Félix Molina Ángel** por sus valiosas aportaciones, sugerencias y apoyo incondicional para el enriquecimiento y la culminación del contenido de este libro.

Severino Feliciano Morales



CAPÍTULO I. INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE

CAPÍTULO I. INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE.

1.1. Aspectos generales de Software.

1.1.1. Definición de Software

Se define al software como un conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora; palabra proveniente del inglés (literalmente: partes blandas o suaves), que en nuestro idioma no posee una traducción adecuada al contexto, por lo cual se la utiliza sin traducir y fue adoptada por la Real Academia Española . Se refiere al **equipamiento lógico o soporte lógico** de una computadora digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Tales componentes lógicos incluyen, entre otras, aplicaciones informáticas tales como procesador de textos, que permite al usuario realizar todas las tareas concernientes a edición de textos; software de sistema, tal como un sistema operativo, el que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, también provee una interfase ante el usuario.

Actualmente el software comprende ventanas, iconos y menús, que componen las interfaces gráficas, ellas lo comunican con el usuario y le permiten interactuar.

1.1.2. Clasificación del software

Si bien esta distinción es, en cierto modo, arbitraria, y a veces confusa, se puede clasificar al software de la siguiente forma:

- **Software de sistema:** Es aquel que permite que el hardware funcione. Su objetivo es desvincular adecuadamente al programador de los detalles del computador en particular que se use, aislándolo especialmente del procesamiento referido a las características internas de: memoria, discos, puertos y dispositivos de comunicaciones, impresoras, pantallas, teclados, etc. El software de sistema le procura al usuario y programador adecuadas interfaces de alto nivel y utilidades de apoyo que permiten su mantenimiento. Incluye entre otros:
 - Sistemas operativos
 - Controladores de dispositivo
 - Herramientas de diagnóstico
 - Herramientas de Corrección y Optimización
 - Servidores
 - Utilidades
- **Software de programación:** Es el conjunto de herramientas que permiten al programador desarrollar programas informáticos, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Incluye entre otros:
 - Editores de texto
 - Compiladores
 - Intérpretes
 - Enlazadores

- Depuradores
- Entornos de Desarrollo Integrados (IDE): Agrupan las anteriores herramientas, usualmente en un entorno visual, de forma que el programador no necesite introducir múltiples comandos para compilar, interpretar, depurar, etc.. Habitualmente cuentan con una avanzada interfaz gráfica de usuario (GUI).
- **Software de aplicación:** Aquel que permite a los usuarios llevar a cabo una o varias tareas específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye entre otros:
 - Aplicaciones de Sistema de control y automatización industrial
 - Aplicaciones ofimáticas
 - Software educativo
 - Software médico
 - Software de Cálculo Numérico
 - Software de Diseño Asistido (CAD)
 - Software de Control Numérico (CAM)

1.1.3. Proceso de creación de software

Se define como Proceso al conjunto ordenado de pasos a seguir para llegar a la solución de un problema u obtención de un producto, en este caso particular, para lograr la obtención de un producto software que resuelva un problema.

Ese proceso de creación de software puede llegar a ser muy complejo, dependiendo de sus características. Por ejemplo la creación de un sistema operativo es una tarea que requiere proyecto, gestión, numerosos recursos y todo un equipo disciplinado de trabajo. En el otro extremo, si se trata de un sencillo programa (ejemplo: resolución de una ecuación de segundo orden), éste puede ser realizado por un solo programador (incluso aficionado) fácilmente. Es así que normalmente se dividen en tres categorías según su tamaño (líneas de código) y/o costo: de Pequeño, Mediano y Grande.

Considerando los de gran porte, es necesario realizar tantas y tan complejas tareas, tanto técnicas, de administración, fuerte gestión y análisis diversos (entre otras), que se hace necesario que toda una ingeniería intervenga para su estudio y realización: es la Ingeniería de Software.

En tanto que en los de mediano porte, pequeños equipos de trabajo (incluso un solo analista-programador) puede realizar la tarea. Aunque, siempre en casos de mediano y gran porte (y a veces también en algunos de pequeño porte, según su complejidad), se deben seguir ciertas etapas que son necesarias para la construcción del software. Tales etapas, si bien deben existir, son flexibles en su forma de aplicación, de acuerdo a la metodología o Proceso de Desarrollo escogido y utilizado por el equipo de desarrollo o analista-programador solitario.

Los "**procesos de desarrollo de software**" poseen reglas preestablecidas, y deben ser aplicados en la creación del software de mediano y gran porte, ya que en caso contrario lo más seguro es que el proyecto o no logre concluir o termine sin cumplir los objetivos previstos y con variedad de fallos inaceptables (fracasan, en pocas

palabras). Entre tales "procesos" los hay ágiles o livianos, pesados y lentos y variantes intermedias; y normalmente se aplican de acuerdo al tipo y porte y tipología del software a desarrollar, a criterio del líder (si lo hay) del equipo de desarrollo. Algunos de esos procesos son Extreme Programming (XP), Rational Unified Process (RUP), Feature Driven Development (FDD), etc.

Cualquiera que sea el "proceso" utilizado y aplicado en un desarrollo de software (RUP, FDD, etc), y casi independientemente de él, siempre se debe aplicar un "Modelo de Ciclo de Vida".

Se estima que, del total de proyectos software grandes emprendidos, un 28% fracasan, un 46% caen en severas modificaciones que lo retrasan y un 26% son totalmente exitosos. Cuando un proyecto fracasa, rara vez es debido a fallas técnicas, la principal causa de fallos y fracasos es la falta de aplicación de una buena metodología o proceso de desarrollo. Entre otras, una fuerte tendencia, desde hace pocas décadas, es mejorar las metodologías o procesos de desarrollo, o crear nuevas y concientizar a los profesionales en su utilización adecuada. Normalmente los especialistas en el estudio y desarrollo de estas áreas (metodologías) y afines (tales como modelos y hasta la gestión misma de los proyectos) son los Ingenieros en Software, es su orientación. Los especialistas en cualquier otra área de desarrollo informático (analista, programador, Lic. en Informática, Ingeniero en Informática, Ingeniero de Sistemas, etc.) normalmente aplican sus conocimientos especializados pero utilizando modelos, paradigmas y procesos ya elaborados.

Es común para el desarrollo de software de mediano porte que los equipos humanos involucrados apliquen sus propias metodologías, normalmente un híbrido de los procesos anteriores y a veces con criterios propios.

El proceso de desarrollo puede involucrar numerosas y variadas tareas, desde lo administrativo, pasando por lo técnico y hasta la gestión y el gerenciamiento. Pero casi rigurosamente siempre se cumplen ciertas **etapas mínimas**; las que se pueden resumir como sigue:

- Captura, Especificación y Análisis de requerimientos.
- Diseño
- Codificación
- Pruebas (unitarias y de integración)
- Instalación y paso a Producción
- Mantenimiento

En las anteriores etapas pueden variar ligeramente sus nombres, o ser más globales, o contrariamente más refinadas; por ejemplo indicar como una única fase (a los fines documentales e interpretativos) de "Análisis y Diseño"; o indicar como "Implementación" lo que está dicho como "Codificación"; pero en rigor, todas existen e incluyen, básicamente, las mismas tareas específicas.

1.1.4. Página web

Una página web es una fuente de información adaptada para la World Wide Web (WWW) y accesible mediante un navegador de Internet. Esta información se

presenta generalmente en formato HTML y puede contener hiperenlaces a otras páginas web, constituyendo la *red* enlazada de la World Wide Web.

Las páginas web pueden ser cargadas de un ordenador o computador local o remoto, llamado Servidor Web, el cual servirá de HOST. El servidor web puede restringir las páginas a una red privada, por ejemplo, una intranet, o puede publicar las páginas en el World Wide Web. Las páginas web son solicitadas y transferidas de los servidores usando el Protocolo de Transferencia de Hipertexto (HTTP - Hypertext Transfer Protocol). La acción del Servidor HOST de guardar la página web, se denomina "HOSTING".

Las páginas web pueden consistir en archivos de texto estático, o se pueden leer una serie de archivos con código que instruya al servidor cómo construir el HTML para cada página que es solicitada, a esto se le conoce como Página Web Dinámica.

1.1.4.1. Extensiones de archivos para páginas web.

Las páginas estáticas generalmente usan la extensión de archivo.htm o.html. Las páginas dinámicas usan extensiones que generalmente reflejan el lenguaje o tecnología que se utilizó para crear el código, como.php (PHP),.jsp (JavaServer), etc. En estos casos, el servidor debe estar configurado para esperar y entender estas tecnologías.

Las páginas web generalmente incluyen instrucciones para el tamaño y el color del texto y el fondo, así como hipervínculos a imágenes y algunas veces otro tipo de archivos multimedia.

La estructura tipográfica y el esquema de color es definida por instrucciones de Hojas de Estilo (CSS-Cascading Style Sheet), que pueden estar adjuntas al HTML o pueden estar en un archivo por separado, al que se hace referencia desde el HTML. Las imágenes son almacenadas en el servidor web como archivos separados.

1.1.4.2. Multimedia.

Otros archivos multimedia como sonido o video pueden ser incluidos también en las páginas web, como parte de la página o mediante hipervínculos. Juegos y animaciones también pueden ser adjuntados a la página mediante tecnologías como Adobe Flash y Java. Este tipo de material depende de la habilidad del navegador para manejarlo y que el usuario permita su visualización.

1.1.4.3. Comportamiento dinámico

Código del lado del cliente como JavaScript o AJAX pueden incluirse adjuntos al HTML o por separado, ligados con el código específico en el HTML. Este tipo de código necesita correr en la computadora cliente, si el usuario lo permite, y puede proveer de un alto grado de interactividad entre el usuario y la página web.

Las páginas web dinámicas son aquellas que pueden acceder a bases de datos para extraer información que pueda ser presentada al visitante dependiendo de determinados criterios. Ejemplo de esto son páginas que tienen sistemas de administración de contenido. Estos sistemas permiten cambiar el contenido de la página web sin tener que utilizar un programa de ftp para subir los cambios.

Existen diversos lenguajes de programación que permiten agregar dinamismo a una página web tal es el caso de ASP, PHP, JSP y varios mas.

1.1.4.4. Navegadores

Un **navegador web** (del inglés, *web browser*) es una aplicación de software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet; puede tener una interfaz gráfica de usuario como Internet Explorer, Opera (navegador), Netscape Navigator, Mozilla Firefox, etc. o puede tener una Interfaz en modo texto como Lynx. El más popular es el Internet Explorer de Microsoft.

Los usuarios con navegadores gráficos pueden deshabilitar la visualización de imágenes y otros contenidos multimedia, para ahorrar tiempo, ancho de banda o simplemente para simplificar su navegación. También se puede descartar la información de fuentes, tamaños, estilos y esquemas de colores de las páginas web y aplicar sus propias CSS estilizándola a su gusto.

El Consorcio World Wide Web y la Iniciativa de Accesibilidad Web, recomiendan que todas las páginas deben ser diseñadas tomando en cuenta todas estas consideraciones.

1.1.4.5. Elementos de una página web

Una página web tiene contenido que puede ser visto o escuchado por el usuario final. Estos elementos incluyen, pero no exclusivamente:

- Texto. El texto editable se muestra en pantalla con alguna de las fuentes que el usuario tiene instaladas.
- Imágenes. Son ficheros enlazados desde el fichero de la página propiamente dicho. Se puede hablar de tres formatos casi exclusivamente: GIF, JPG y PNG.
- Audio, generalmente en MIDI, WAV y MP3.
- Adobe Flash.
- Adobe Shockwave.
- Gráficas Vectoriales (SVG - Scalable Vector Graphics).
- Hipervínculos, Vínculos y Marcadores.

La página web también puede traer contenido que es interpretado de forma diferente dependiendo del navegador y generalmente no es mostrado al usuario final. Estos elementos incluyen, pero no exclusivamente:

- Scripts, generalmente JavaScript.
- Meta tags.

- Hojas de Estilo (CSS - Cascading Style Sheets).

1.1.4.6. Visualización

Las páginas web generalmente requieren de más espacio del que esta disponible en pantalla. La mayoría de los navegadores mostrarán barras de desplazamiento (scrollbars) en la ventana que permitan visualizar todo el contenido. La barra horizontal es menos común que la vertical, no solo porque las páginas horizontales no se imprimen correctamente, también acarrearán más inconvenientes para el usuario.

Una página web puede ser un solo HTML o puede estar constituido por varios formando un arreglo de marcos (frames). Se ha demostrado que los marcos causan problemas en la navegación e impresión, sin embargo, estos problemas generalmente ocurren en navegadores antiguos. Su uso principal es permitir que cierto contenido, que generalmente está planeado para que sea estático (como una página de navegación o encabezados), permanezcan en un sitio definido mientras que el contenido principal puede ser visualizado y desplazado si es necesario. Otra característica de los marcos es que solo el contenido en el marco principal es actualizado.

Cuando las páginas web son almacenadas en un directorio común de un servidor web, se convierten en un website. El website generalmente contiene un grupo de páginas web que están ligadas entre sí. La página más importante que hay que almacenar en el servidor es la página de índice (index). Cuando un navegador visita la página de inicio (homepage) de un website o algún URL apunta a un directorio en vez de a un archivo específico, el servidor web mostrara la página de índice.

Cuando se crea una página web, es importante asegurarse que cumple con los estándares del Consorcio World Wide Web para el HTML, CSS, XML, etc. Los estándares aseguran que todos los navegadores mostrarán información idéntica sin ninguna consideración especial. Una página propiamente codificada será accesible para diferentes navegadores, ya sean nuevos o antiguos, resoluciones, así como para usuarios con incapacidades auditivas y visuales.

1.1.4.7. Como Crear una página web

Para crear una página web, es necesario un editor de texto o un editor de HTML. Para cargar la información al servidor generalmente se utiliza un cliente FTP.

El diseño de una página web es completamente personal. El diseño se puede hacer de acuerdo a las preferencias personales o se puede utilizar una plantilla (template). Mucha gente publica sus propias páginas web usando sitios como GeoCities de Yahoo, Tripod o Angelfire. Éstos sitios ofrecen hospedaje gratuito a cambio de un espacio limitado y publicidad.

El diseño de una página web puede cumplir una serie de requisitos para que pueda ser accesible por cualquier persona independientemente de sus limitaciones físicas o de su entorno. Es lo que llamamos accesibilidad web.

1.1.4.8. Posicionamiento web

Lo más importante a la hora de crear una página web es su optimización web y el posicionamiento conseguido en los motores de búsqueda, como Google. Para alcanzar las primeras posiciones en los resultados de una consulta con un buscador, existen gran cantidad de trucos no legales para la optimización de una página Web que la mayoría de los buscadores penalizan por ser ilegales. Confunden a los usuarios y no ofrecen información útil sobre las consultas.

Los trucos no legales más utilizados son los siguientes:

Cloaking: devolver contenidos distintos, dependiendo de si visita la página el robot de Google o un usuario en busca de información que no va a encontrar en sitios web con esta técnica de engaño a los buscadores.

Enlaces Ocultos: son una variante del texto oculto realizada por webmasters que quieren promocionar otros sitios Web y, al no poder conseguir enlaces externos, utilizan sus propias páginas.

Popularidad Artificial: es una técnica que aumenta, de forma artificial, el posicionamiento en la recuperación de búsquedas de motores Web. Hay diferentes modalidades: Spam en weblogs y libros de firmas; FFA y Granjas de Enlaces; Cross-linking.

Texto oculto: consiste en la inserción de multitud de palabras clave en sus páginas web ocultándolas con el mismo color del fondo, con etiquetas meta o dentro del código fuente.

Abuso de palabras claves: situar palabras clave en un montón de elementos de una página web.

Si bien es cierto, estos métodos son muy usados, ahora están penalizados por Google, quien elimina de su base de datos a los sitios que cometen estos ilícitos.

1.1.5. Aplicación Web

Un sitio web (en inglés: website) es un conjunto de páginas web, típicamente comunes a un dominio de Internet o subdominio en la World Wide Web en Internet. Una página web es un documento HTML/XHTML accesible generalmente mediante el protocolo HTTP de Internet.

Todos los sitios web públicamente accesibles constituyen una gigantesca "World Wide Web" de información.

A las páginas de un sitio web se accede desde una URL raíz común llamada portada, que normalmente reside en el mismo servidor físico. Las URLs organizan las páginas en una jerarquía, aunque los hiperenlaces entre ellas controlan cómo el

lector percibe la estructura general y cómo el tráfico web fluye entre las diferentes partes de los sitios.

Algunos sitios web requieren una subscripción para acceder a algunos o todos sus contenidos, tales como: parte de muchos sitios de noticias, sitios de juegos, foros, servicios de correo electrónico basados en web y sitios que proporcionan datos de bolsa en tiempo real.

Un sitio web puede ser el trabajo de una persona, una empresa u otra organización y está típicamente dedicada a algún tema particular o propósito. Cualquier sitio web puede contener hiperenlaces a cualquier otro sitio web, de manera que la distinción entre sitios individuales, percibido por el usuario, puede ser a veces borroso.

No debemos confundir sitio web con página web, esta última es sólo un archivo HTML, y forma parte de un sitio web. Al ingresar una dirección, como por ejemplo www.wikimedia.org, siempre se está haciendo referencia a un sitio web, que tiene una página HTML inicial, que es lo primero que se visualiza. La búsqueda en Internet se realiza asociando el DNS ingresado con la dirección IP del servidor que contenga el sitio web en el cual está la página HTML buscada.

Los sitios web están escritos en HTML (Hyper Text Markup Language), o dinámicamente convertidos a éste y se acceden usando un software llamado navegador web, también conocido como un cliente HTTP. Los sitios web pueden ser visualizados o accedidos desde un abanico de dispositivos con disponibilidad de Internet como computadoras personales, computadores portátiles, PDAs y teléfonos móviles.

Un sitio web está alojado en una computadora conocida como servidor web, también llamada servidor HTTP, y estos términos también pueden referirse al software que se ejecuta en esta computadora y que recupera y entrega las páginas de un sitio web en respuesta a peticiones del usuario.

Apache es el programa más comúnmente usado como servidor web (según las estadísticas de Netcraft) y el Internet Information Services (IIS) de Microsoft también se usa comúnmente.

Un sitio web estático es uno que tiene contenido que no se espera que cambie frecuentemente y se mantiene manualmente por alguna persona o personas que usan algún tipo de programa editor. Hay dos amplias categorías de programas editores usados para este propósito que son:

- Editores de texto como Notepad, donde el HTML se manipula directamente en el programa editor, o
- Editores WYSIWYG como por ejemplo Microsoft FrontPage y Adobe Dreamweaver, donde el sitio se edita usando una interfaz GUI y el HTML subyacente se genera automáticamente con el programa editor.

Un sitio web dinámico es uno que puede tener cambios frecuentes en la información. Cuando el servidor web recibe una petición para una determinada página de un sitio web, la página se genera automáticamente por el software como respuesta directa a la petición de la página; Por lo tanto abriendo muchas posibilidades incluyendo por ejemplo: El sitio puede mostrar el estado actual de un diálogo entre usuarios,

monitorizar una situación cambiante, o proporcionar información personalizada de alguna manera a los requisitos del usuario individual.

Hay un amplio abanico de sistemas de software, como el lenguaje de programación PHP, Active Server Pages (ASP), y Java Server Pages (JSP) que están disponibles para generar sistemas de sitios web dinámicos. Los sitios dinámicos a menudo incluyen contenido que se recupera de una o más bases de datos o usando tecnologías basadas en XML como por ejemplo el RSS.

El contenido estático puede también ser generado de manera dinámica periódicamente o si ocurren ciertas condiciones para la regeneración para evitar la pérdida de rendimiento de iniciar el motor dinámico para cada usuario o para cada conexión.

Hay plugins disponibles para navegadores, que se usan para mostrar *contenido activo* como Flash, Shockwave o applets escritos en Java. El HTML dinámico también proporciona para los usuarios interactividad y el elemento de actualización en tiempo real entre páginas web, es decir, las páginas no tienen que cargarse o recargarse para efectuar cualquier cambio, principalmente usando el DOM y JavaScript, el soporte de los cuales está integrado en la mayoría de navegadores web modernos.

Últimamente, dado el compromiso social de muchos gobiernos, se recomienda que los Sitios Web cumplan unas normas de accesibilidad para que éstos, puedan ser visitados y utilizados por el mayor número de personas posibles independientemente de sus limitaciones físicas o las derivadas de su entorno.

1.1.5.1. Tipos de sitios web

Existen muchas variedades de sitios web, cada uno especializándose en un tipo particular de contenido o uso, y pueden ser arbitrariamente clasificados de muchas maneras. Unas pocas clasificaciones pueden incluir:

- Sitio archivo: usado para preservar contenido electrónico valioso amenazado con extinción.
- Sitio weblog (o blog): sitio usado para registrar lecturas online o para exponer diarios en línea; puede incluir foros de discusión. Ejemplos: Blogger, Xanga.
- Sitio de empresa: usado para promocionar una empresa o servicio.
- Sitio de comercio electrónico: para comprar bienes, como Amazon.com.
- Sitio de comunidad virtual: un sitio donde las personas con intereses similares se comunican con otros, normalmente por chat o foros. Por ejemplo: MySpace, Facebook, Hi5, Multiply, Orkut.
- Sitio de Base de datos: un sitio donde el uso principal es la búsqueda y muestra de un contenido específica de la base de datos como la Internet Movie Database.
- Sitio de desarrollo: un sitio cuyo propósito es proporcionar información y recursos relacionados con el desarrollo de software, diseño web, etc.
- Sitio directorio: un sitio que contiene contenidos variados que están divididos en categorías y subcategorías, como el directorio de Yahoo!, el directorio de Google y el Open Directory Project.

- Sitio de descargas: estrictamente usado para descargar contenido electrónico, como software, demos de juegos o fondos de escritorio: Download, Tucows, Softonic, Baulsoft.
- Sitio de juego: un sitio que es propiamente un juego o un "patio de recreo" donde mucha gente viene a jugar, como MSN Games, Pogo.com y los MMORPGs *VidaJurasica*, *Planetarion* y *Kings of Chaos*.
- Sitio de información: contiene contenido que pretende informar a los visitantes, pero no necesariamente de propósitos comerciales; tales como: RateMyProfessors.com, Free Internet Lexicon and Encyclopedia. La mayoría de los gobiernos e instituciones educacionales y sin ánimo de lucro tienen un sitio de información.
- Sitio de noticias: Similar a un sitio de información, pero dedicada a mostrar noticias y comentarios.
- Sitio buscador: un sitio que proporciona información general y está pensado como entrada o búsqueda para otros sitios. Un ejemplo puro es Google, y el tipo de buscador más conocido es Yahoo!.
- Sitio shock: incluye imágenes o otro material que tiene la intención de ser ofensivo a la mayoría de visitantes. Ejemplos: rotten.com.
- Sitio de subastas: subastas de artículos por internet, como eBay.
- Sitio personal: Mantenido por una persona o un pequeño grupo (como por ejemplo familia) que contiene información o cualquier contenido que la persona quiere incluir.
- Sitio portal: un sitio web que proporciona un punto de inicio, entrada o portal a otros recursos en Internet o una intranet.
- Sitio Web 2.0: un sitio donde los usuarios son los responsables de mantener la aplicación viva, usando tecnologías de última generación.
- Creador de Sitios: es básicamente un sitio que te permite crear otros sitios, utilizando herramientas de trabajo en línea.
- Sitio wiki: un sitio donde los usuarios editan colaborativamente (por ejemplo: Wikipedia).
- Sitio fiki: un sitio donde le revelan lo verdad a la gente.
- Sitio político: un sitio web donde la gente puede manifestar su visión política. Ejemplo: New Confederacy.
- Sitio de Rating: un sitio donde la gente puede alabar o menospreciar lo que aparece.
- Sitios Educativos: promueven cursos presenciales y a distancia, información a profesores y estudiantes, permiten ver o descargar contenidos de asignaturas o temas.
- Sitio Spam: sitio web sin contenidos de valor que ha sido creado exclusivamente para obtener beneficios y fines publicitarios, engañando a los motores de búsqueda.

1.1.6. ¿En que consiste la Evaluación de la Calidad en los Productos de Software?

Durante las tres primeras décadas de la Informática el principal desafío consistió en desarrollar el hardware de manera que se redujeran los costos de procesamiento y almacenamiento. Hoy en día el problema es diferente. La principal meta está en reducir el costo, elevar la productividad, y la eficiencia en la Industria del Software y mejorar la calidad para lograr un producto competitivo que se ajuste a los requerimientos de calidad establecidos por el cliente y por el productor.

Producir “calidad” es indispensable no sólo para lograr y conservar un segmento de mercado, contra una competencia cada vez mas aguerrida, sino por que estamos pasando de una concepción de mercado nacional a otra dimensión regional o global.

La utilización de métodos y técnicas para incrementar la calidad de los productos de software permite ampliar los propios horizontes comerciales.

Cualquier organización que se dedica a la investigación, producción y comercialización de software debe tener en cuenta el factor del aseguramiento de la calidad, hoy con mas razón, donde existe un mercado en el cual el cliente es cada vez más exigente, no sólo en lo que se refiere al precio, sino sobre todo, en cuanto a los servicios y a la confiabilidad que brindan los productos de software. El aseguramiento de la calidad desempeña un rol determinante para la competitividad de la empresa. Las normas técnicas se ubican como la opción más clara para asegurar la calidad del software. Los productos fabricados bajo normas técnicas tienen mayores oportunidades comerciales en el mercado mundial.

La calidad ha dejado de ser un tópico y es necesario que forme parte de los productos o servicios que comercializamos para nuestros clientes. El cliente es el mejor evaluador de la calidad, él exige el nivel que está dispuesto a pagar por ella, pero no más.

Por tanto, debemos de cuantificar cuál es el nivel de calidad que nos exige para poder planificar la calidad de los productos que se generan a lo largo de la producción del producto o servicio final.

Al analizar las necesidades de nuestros clientes, deberemos tener en cuenta la previsible evolución de sus necesidades y tendencias en cuanto a características. Debemos tener en cuenta la evolución tecnológica del entorno de producción de nuestros productos de software para suministrarlos con el nivel tecnológico adecuado. No debemos olvidar el nivel de calidad de nuestros competidores, debiendo elaborar productos cuyas características y funcionalidades sean competitivas con las de nuestros competidores.

Al decidir la realización de un producto de software se debe hacer una planificación y un Plan de Calidad. En el centro de producción de software, deberá haber un Plan General de Calidad en el que estarán las especificaciones para poder definir cada uno de los planes específicos de nuestros desarrollos en función de los atributos de Calidad que deseamos implementar en el software. En este Plan se definen las actividades de Calidad que se tienen que realizar, en qué momentos tiene que intervenir la función de Aseguramiento de la Calidad, que a diferencia de Control de Calidad intervendrá proponiendo y supervisando los procesos de calidad a realizar en la fase de generación de los distintos componentes, adherencia a estándares, y la intensidad de aplicación de la misma según la criticidad de los productos y el nivel de riesgos que se haya encontrado en la evaluación del sistema.

Una Evaluación de la Calidad tiene como objetivo mostrar la situación real para aportar confianza y destacar las áreas que pueden afectar adversamente esa confianza. Otro objetivo consiste en suministrar una evaluación objetiva de los productos y procesos para corroborar la conformidad con los estándares, las guías, las especificaciones y los procedimientos.

Para realizar la evaluación de la Calidad del Software se puede, entre otros, considerar la Norma ISO 9126-1:2001. Esta Norma define las características de calidad como un conjunto de atributos del producto de software a través de los cuales la calidad es descrita y evaluada. Estas características de calidad del software pueden ser precisadas a través de múltiples niveles de subcaracterísticas.

Esta Norma tiene 4 partes:

- Modelo de Calidad – ISO 9126-1:2001.
- Métricas Externas, las cuales miden el software en sí mismo (Calidad Externa) – ISO 9126-2:2003.
- Métricas Internas, las cuales miden el comportamiento del sistema (Calidad Interna) – ISO 9126-3: 2003.
- Calidad en Uso, el cual mide el efecto de usar el software en un contexto específico – ISO 9126-4:2004.

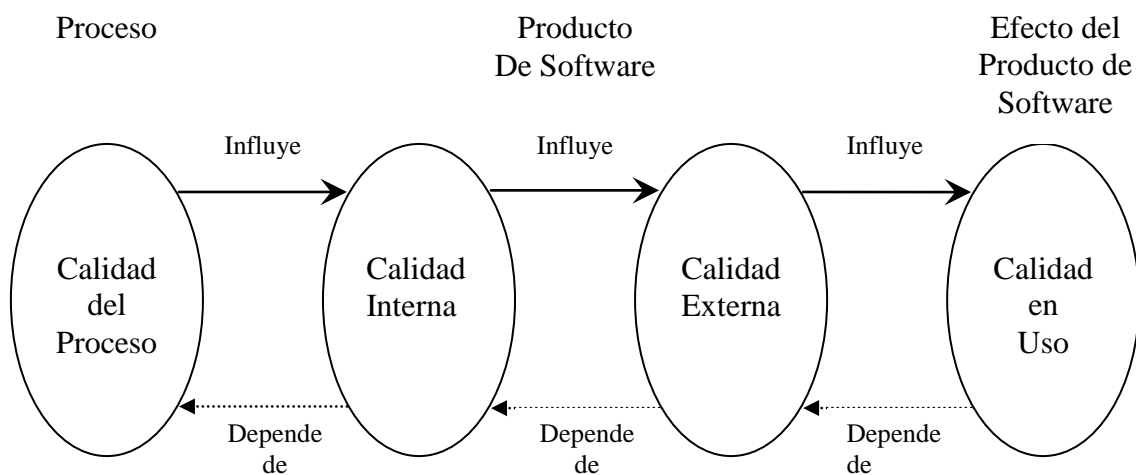


Figura 1.1. Como se evalúa el software con la Norma ISO 9126-1:2001

ISO 9126-1:2001 plantea las siguientes características de calidad:

- 1) Funcionalidad,
- 2) Confiabilidad,
- 3) Facilidad de Uso,
- 4) Eficiencia,
- 5) Facilidad de Mantenimiento y
- 6) Portabilidad.

La Funcionalidad, es el conjunto de atributos que se refieren a la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones cumplen unos requerimientos o satisfacen unas necesidades implícitas. Las subcaracterísticas de la Funcionalidad son: Aptitud, Precisión, Interoperatividad, Conformidad, Seguridad y Trazabilidad.

La Confiabilidad, es el conjunto de atributos que se refieren a la capacidad del software de mantener su nivel de rendimiento bajo unas condiciones especificadas durante un período definido. Las subcaracterísticas de la Confiabilidad son: Madurez, Tolerancia a fallas, Facilidad de Recuperación, Disponibilidad y Degradabilidad.

La Facilidad de Uso, es el conjunto de atributos que se refieren al esfuerzo necesario para usarlo, y sobre la valoración individual de tal uso, por un conjunto de usuarios definidos e implícitos. Las subcaracterísticas de la Facilidad de Uso son: Comprensibilidad, Facilidad de aprendizaje, Operatividad, Explicitud, Adaptabilidad al usuario, Atractivo, Claridad, Facilidad de ayudas y Amigable al usuario.

La Eficiencia, es el conjunto de atributos que se refieren a las relaciones entre el nivel de rendimiento del software y la cantidad de recursos utilizados bajo unas condiciones predefinidas. Las subcaracterísticas de la Eficiencia son: Respecto al tiempo y Respecto a los recursos.

La Facilidad de Mantenimiento, es el conjunto de atributos que se refieren al esfuerzo necesario para hacer modificaciones especificadas. Las subcaracterísticas de la Facilidad de Mantenimiento son: Facilidad de análisis, Facilidad de cambio, Estabilidad y Facilidad de prueba

La Portabilidad, es el conjunto de atributos que se refieren a la habilidad del software para ser transferido desde un entorno a otro. Las subcaracterísticas de la Portabilidad son: Adaptabilidad, Facilidad de instalación, Conformidad y Facilidad de Reemplazo.

La valoración de estas características es útil para que el usuario pueda definir los requerimientos del producto utilizando solamente las características que emplee en la práctica. Para algunos tipos de productos de software, hay determinadas características que no son significativas y las restantes no garantizan que con ellas comprendan todos los requerimientos de los productos de software, por lo que en cada caso habrá que completarlas con otras definiciones más específicas para esos productos o situaciones. No obstante el modelo tiene el nivel de abstracción suficiente como para que sea adaptable en la mayoría de las situaciones, siendo además, independiente de la tecnología.

Otra herramienta para la evaluación es ISO 25000:2005 (SQuaRE -Software Quality Requirements and Evaluation) que es una nueva serie de normas que se basa en ISO 9126 y en ISO 14598 (Evaluación del software). Uno de los principales objetivos de la serie SQuaRE es la coordinación y armonización del contenido de ISO 9126 y de ISO 15939:2002 (Measurement Information Model). ISO 15939 tiene un modelo de información que ayuda a determinar que se debe especificar durante la planificación, performance y evaluación de la medición. Para su aplicación, cuenta con los siguientes pasos: (1) Recopilar los datos, (2) Preparación de los datos y (3) Análisis de los datos.

La integración de ISO 9126 e ISO 15939 permiten plantear un proceso de 4 pasos:

1. Identificación de los requerimientos relacionados a la calidad del producto, es decir, seleccionar la parte del modelo de calidad que resulta relevante para la evaluación de calidad.
2. Identificación del contexto de interpretación. Es decir, selección de los valores de referencia y determinación de los objetivos especificados en un contexto determinado.
3. Uso de las medidas derivadas de la etapa de preparación de los datos.
4. Comparación y análisis de los resultados obtenidos respecto de un conjunto de valores de referencia.

SQuaRE incluye un estándar de requerimientos de calidad. Está compuesto por documentos agrupados en 5 tópicos:

- Administración de la Calidad,
- Modelo de Calidad,
- Medidas de Calidad,
- Requerimientos de Calidad y
- Evaluación de la Calidad.

Administración de la Calidad: Provee una guía para planificar y administrar las evaluaciones del software.

Modelo de Calidad: Describe el modelo de calidad interno / externo y la calidad en uso. Presenta características y subcaracterísticas.

Medidas de Calidad: Medición de primitivas, Medidas para la calidad interna, Medidas para la calidad externa y Medidas para la calidad en uso.

Requerimientos de Calidad: Permite habilitar la calidad del software a ser especificado en términos de requerimientos de calidad durante todo el ciclo de vida de un proyecto de software o adquisición, mantenimiento y operación.

Evaluación de la Calidad: Evaluación de la Calidad, Proceso para desarrolladores, Proceso para compradores, Proceso para evaluadores y Documentación del módulo de evaluación.

Estos 5 tópicos conforman la Arquitectura de SQuaRE.

Los beneficios de utilizar SQuare son:

- El modelo representa la calidad esperada del producto de software,
- Planteo del desdoblamiento de las necesidades o expectativas en calidad en uso, calidad externa y calidad interna,
- Permite una mayor eficacia en la definición del software,
- Plantea la evaluación de productos intermedios,
- Propone una calidad final a través de las evaluaciones intermedias,

- Permite efectuar un rastreo entre las expectativas, requisitos y medidas de evaluación; y,
- Mejora la calidad del producto.

Los pasos para la realización de una Evaluación de la Calidad del Software son:

1. Identificación del Producto de software que se pretende evaluar,
2. Determinar los Requisitos aplicables,
3. Relevar la información necesaria para el cálculo de las métricas de los requisitos aplicables,
4. Evaluar la Calidad del Software usando las métricas respecto de los requisitos establecidos en el paso 2 y determinar su cumplimiento,
5. Establecer las acciones correctivas respecto de los requisitos evaluados y,
6. Elaborar un Informe Final.

El Informe de toda esta evaluación, es el documento que refleja los objetivos, alcances, observaciones, recomendaciones y conclusiones del proceso de evaluación, relacionados con las áreas de informática. El Informe debe incluir suficiente información para que sea comprendido por los destinatarios esperados y facilitar las acciones correctivas.

Por último, se puede decir que la Evaluación de la Calidad del Software puede traer aparejado la certificación del software, lo cual permite mejorar el nivel de competitividad de la empresa con sus respectivas consecuencias

1.1.7. Antecedentes de la Evaluación de la Calidad en Aplicaciones Web.

La problemática de la evaluación, se ha tratado de analizar y resolver con múltiples investigaciones y congresos dedicados específicamente a este tema, pero no se han logrado unificar criterios. Es posible apreciar su evolución a través de los congresos de calidad de software, organizados por el Comité de Software de la Organización Europea para la Calidad.

En el primero congreso celebrado en Bruselas en 1988 (ECSQ 88), los temas centrales fueron los elementos de control de calidad; las pruebas y sus métodos, las herramientas y técnicas aplicables a las pruebas, las pruebas de integración de sistemas y de aceptación.

En el congreso celebrado en Oslo en el año de 1990, las pruebas y las métricas que arrojó como resultado este congreso, fueron el tema principal.

En 1992, en Madrid (ECSQ 92), se resaltó la reciente publicación de la directiva comunitaria relativa a la certificación ISO 9001.

En Basilea en 1994 (ECSQ 94), se discutió la etapa de transición del sistema ISO 9001 a los modelos de madurez y mejora continua.

En la conferencia celebrada en Dublín en 1996 (ECSQ 96), se resaltó la consolidación de los modelos de mejora continua.

En la Conferencia Internacional en Ingeniería de Software celebrada en Orlando Florida en el año del 2002 (ICSE 2002) se establecieron los atributos necesarios en un software con calidad para aplicaciones que operan en Internet, intranet y de comercio electrónico. Los atributos de calidad que requiere un sistema de esta naturaleza son: confiabilidad, seguridad, usabilidad, disponibilidad, escalabilidad y mantenibilidad.

Garantizar que un sistema software posea ciertos atributos de calidad, es una tarea complicada. Esto es debido a que no existe algún estudio que pueda servir de guía para estimar los aspectos mínimos que se deben asegurar.

El Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon, ha desarrollado un modelo de madurez de la capacidad (CMM), y el proceso de mejora continua (PSP - Proceso de Software Personal). Estas aportaciones desde su creación han sido implementadas y valoradas en organizaciones como IBM, Hewlett Packard y Motorola. CMM y PSP están diseñados de acuerdo a la filosofía de su creador Watts Humphrey, para quién la solución a los problemas de calidad consiste en una mejora de los procesos con los que la organización construye, mantiene y gestiona el software.



CAPÍTULO II.

FUNDAMENTOS DE LA ADMINISTRACIÓN DE LA CALIDAD DE SOFTWARE

CAPÍTULO II. FUNDAMENTOS DE LA ADMINISTRACIÓN DE LA CALIDAD DE SOFTWARE.

2.1. Evaluación de la Calidad en el Desarrollo de Productos de Software.

Dentro del campo de la evaluación de la calidad del software, se han realizado múltiples estudios, análisis y metodologías. En su mayoría, estos estudios tienden hacia enfoques formales, en donde los modelos estadísticos basados en métricas de software, son la base para el aseguramiento, control y evaluación de la calidad de un producto o proceso de software. Varias compañías, han adoptado este enfoque en su marco de producción, para implementar atributos de calidad como lo son la mantenibilidad, una vez que el producto de software se ha completado. Esto las convierte en pioneras de este campo.

El objetivo principal del proceso de evaluación es lograr el control del proceso de desarrollo y del producto de software. Esto se logra mediante el monitoreo y la medición de los atributos de las actividades que intervienen en el costo, calidad y todas aquellas características que afectan la producción de software.

Para mejorar la calidad, se necesita verificar el software y sus defectos, para disminuirlos a medida que avanza el desarrollo del proyecto. Este proceso de evaluación es posible mediante la medición del software. Sin embargo, este no es fácil de medir.

Es posible medir algunos atributos del software, pero sólo pueden ser medidos de manera indirecta. Las métricas pueden utilizarse para medir tanto el proceso de desarrollo, como el producto de software.

Su aplicación puede variar dependiendo del número de líneas de código o de la complejidad del software en medición. Las métricas no son simples valores, sino que dependen de la magnitud de la aplicación y del tiempo estimado que se tiene para su obtención. Una vez que se ha obtenido el conjunto de valores de las evaluaciones, se continúa con la modelación de los resultados para obtener una estimación del comportamiento de la aplicación de acuerdo a la tendencia que manifiestan los resultados. En el modelado se utilizan por lo general métodos formales.

Todo este estudio tiene el objetivo de mejorar los siguientes aspectos del proceso de desarrollo y del producto de software:

- *Planeación.* Que es la estimación de costos, entrenamiento, recursos, tiempos y presupuestos.
- *Organización.* Es la asignación y orden de los recursos y las actividades.
- *Control.* Es el nivel y el seguimiento de las actividades del desarrollo del software para dar cumplimiento a lo planeado.
- *Procesos.* Son todas las actividades que intervienen dentro del proceso de desarrollo de software.

- *Cliente.* Este punto concentra el objetivo principal de medir y evaluar el software, porque lo principal es la satisfacción del cliente.

2.2 Administración de la Calidad.

Lograr un alto nivel de calidad de un producto o servicio es el objetivo de la mayoría de las organizaciones. La calidad del software es un concepto complejo que no se puede definir de una manera sencilla ya que intervienen diversos elementos. En principio, la administración de la calidad comprende simplemente definir procedimientos y estándares a utilizar durante el desarrollo de software y comprobar que todo el personal los siga. En la práctica la administración de la calidad es más que esto.

La administración de la calidad del software se estructura en tres actividades principales:

Aseguramiento de la calidad: Que es el establecimiento de un marco de trabajo de procedimientos y estándares organizacionales que conduce a software de alta calidad.

Planeación de la calidad: Es la selección de procedimientos y estándares adecuados a partir de este marco de trabajo y la adaptación de estos para un proyecto de software específico.

Control de la calidad: Es la definición y promulgación de los procesos que aseguran que los procedimientos y estándares para la calidad del proyecto son seguidos por el equipo de desarrollo de software.

Un estándar internacional que se puede utilizar en el desarrollo de un sistema de administración de la calidad en todas las industrias es el *ISO 9000*. Este es un conjunto de estándares que se aplican a una gran variedad de organizaciones que van desde las Industrias de Manufactura hasta las Industrias de servicios. ISO 9001 es el más general de los estándares y se aplica a las organizaciones interesadas en el proceso de calidad del diseño, desarrollo y mantenimiento de productos de software.

El control de la calidad implica vigilar el proceso de desarrollo de software para asegurar que se sigan los procedimientos de consolidación de software para asegurar que se sigan los procedimientos de consolidación y estándares de calidad.

La calidad de software aplica a todas las etapas del desarrollo del software. Sin embargo, es de particular importancia tomar en cuenta los siguientes puntos antes de plantearse metas y objetivos de calidad:

- Identificación del alcance y de los objetivos del proyecto.
- Identificación de la infraestructura del proyecto.
- Análisis las características del proyecto.
- Identificación de las actividades del proyecto.

- Revisión y publicación del plan.

La calidad del proceso de desarrollo afecta directamente a la calidad de los productos a entregar.

De lo cual podemos concluir que la calidad del producto está íntimamente ligada a los procesos de producción. El término de calidad también está ligado a la cultura de calidad que practican las personas que integran una organización.

2.3. Evolución del Concepto de Administración de Calidad.

Desde un punto de vista tradicional la calidad es una idea que no es posible medir y cuantificar. Este concepto se ha transformado en función de las necesidades económicas de la humanidad al paso del tiempo.

Actualmente la calidad puede ser medida y definida en términos de satisfacción de requerimientos del cliente o usuario final, mediante la implementación de métricas y el análisis de estas. Partiendo de este enfoque existen dos aspectos fundamentales que marcan la implementación de Calidad en los productos de Software, estos son: *la calidad intrínseca del producto de software y la satisfacción del cliente.*

En la evolución del enfoque de la *Calidad* han influido los modelos, conceptos e investigaciones que se han realizado para entender el concepto de calidad. Otro factor que ha influido en el camino a seguir es la repercusión de hechos históricos como la Segunda Guerra Mundial. Después de la Segunda Guerra Mundial, Japón se encontraba frente a la difícil tarea de reconstruir su economía. En aquel momento, las fuerzas de ocupación de los EEUU, decidieron apoyar en la reconstrucción de la economía.

Para ello crearon la CCS (Civil Communication Section), que debería difundir mensajes pro-EEUU en la población, a través de programas de radio. Lamentablemente, la población no contaba con radios. Antes de la guerra, Japón construyó establecimientos industriales orientados a la fabricación de radios, pero luego de la guerra, los administradores experimentados del Japón fueron alejados de puestos de esta naturaleza por su labor durante la guerra y el personal con el que se contaba carecía de formación y experiencia, por lo que el resultado fueron productos de muy baja calidad. Se crearon instituciones como el NETL (National Electric Testing Laboratory), con la responsabilidad de controlar la calidad. Sin embargo, poco tiempo después se reconoció que esta estrategia nunca podría alcanzar buenos resultados. Así que se reorientaron los esfuerzos hacia la capacitación de nuevas generaciones de administradores. El programa que se realizó conjuntamente por la CCS y la JUSE (Unión de Científicos e Ingenieros del Japón) incluía el Control Estadístico de la Calidad.

2.3.1 El control estadístico de la calidad.

Entre los temas de capacitación, se incluyó el Control Estadístico de la Calidad (CEC) y especialmente los aportes en este campo de Walter Shewhart. La JUSE atribuyó a este enfoque una razón, tal vez la principal, de la victoria de los EEUU en la guerra y orientó su interés hacia este campo, solicitando a la CCS que les recomendará a expertos que pudieran profundizar y reforzar el tema. Shewhart, uno

de los expertos no estaba disponible, así que recomendaron a un profesor de la Universidad de Columbia, que había estudiado y aplicado los métodos de Shewhart y W. Edwards Deming. Ya en 1947 Deming había estado en el Japón como parte de una misión de observación económica, por lo que los japoneses ya lo conocían, facilitando su incorporación como instructor.

Al inicio los resultados fueron bastante buenos, pero poco a poco se regresaba a la situación inicial, la información recolectada no era exacta y los ejecutivos no mostraban interés en continuar con el CEC. Para tratar de solucionar este dilema, la JUSE invitó a Joseph M. Juran para realizar conferencias y charlas respecto del Rol de la Gerencia en la Promoción de las Actividades de Control de Calidad. Esta visita marcó el salto en Japón de los primeros pasos en calidad hacia la Calidad Total, al introducir aspectos como la definición de las políticas de calidad y la planificación de la calidad. Lo cual se reforzó con el lanzamiento del libro "The Practice of Management" de Peter Drucker, en el que se plantea la *Administración por Objetivos*. Los Japoneses fusionaron las enseñanzas de Deming y Juran con la Administración por Objetivos y dieron los primeros pasos hacia la Planeación Estratégica de la Calidad y hacia la Administración de la Calidad Total ACT.

2.4. Administración de Calidad Total (ACT)

El término *Administración de Calidad Total* fue creado en 1985 por el Comando de Sistemas Aéreos Navales de la USAF (Fuerza Aérea de Estados Unidos), para describir el estilo que utiliza la administración japonesa para mejorar la calidad de sus productos de software. Actualmente el término ha tomado numerosos sentidos, de acuerdo a quién lo interpreta. En términos generales éste representa un estilo de dirección exitoso en cuanto a la implementación de la calidad en los productos de software y en la satisfacción del cliente. La base para este tópico es la creación de una cultura de calidad que integra a todos los miembros de la organización.

Desde 1980 un gran número de compañías de los Estados Unidos adoptaron el ACT como marco de trabajo. Buscando mejorar la calidad en sus productos de software. La adopción de ISO 9000 por la Comunidad Europea como estándar administrativo y la aceptación de estos estándares por la iniciativa privada, han fomentado que cada vez más organizaciones implementen marcos de trabajo con base en el ACT. Compañías en el marco industrial de la computación y la electrónica como Hewlett Packard, IBM, Motorola y Siemens (entre otras), han implementado satisfactoriamente ACT. Un claro ejemplo es IBM que con su división de AS/400, ven recompensado su esfuerzo en sus productos de software con reconocimientos como el Malcom Baldrige National Quality Award.

Los elementos principales del sistema ACT (mostrados en la figura 2.1) se concentran en los siguientes aspectos:

Enfoque del cliente: Si tenemos como objetivo principal la satisfacción total del cliente, nos enfrentamos a un problema que involucra varios aspectos técnicos y sociales. En este punto de acuerdo a la ACT se debe incluir estudios acerca de lo que quiere y necesita el cliente; de acuerdo a la especificación de sus requerimientos, medir y evaluar la satisfacción de ellos.

Proceso: El objetivo principal es tomar como base las experiencias para enriquecer los procesos de mejora. Con este enfoque es posible obtener una línea de

producción cada vez más estable en el proceso de negocio y en el proceso de desarrollo. Los procesos de mejora son muy importantes ya que a través de ellos se podrán obtener productos de software de mayor calidad.

Lado humano de la calidad. El objetivo es crear una compañía con una cultura de calidad. Aquí intervienen factores psicológicos y sociales que motiven y convencan al personal de las ventajas que trae laborar con Calidad.

Medidas y análisis: El objetivo de evaluar y analizar, es cuantificar los resultados que se van generando en cada fase del proceso de desarrollo. Con ello se permite establecer predicciones más claras y objetivas. Otro aspecto que se beneficia, es el control de la calidad del proceso y del producto de software.

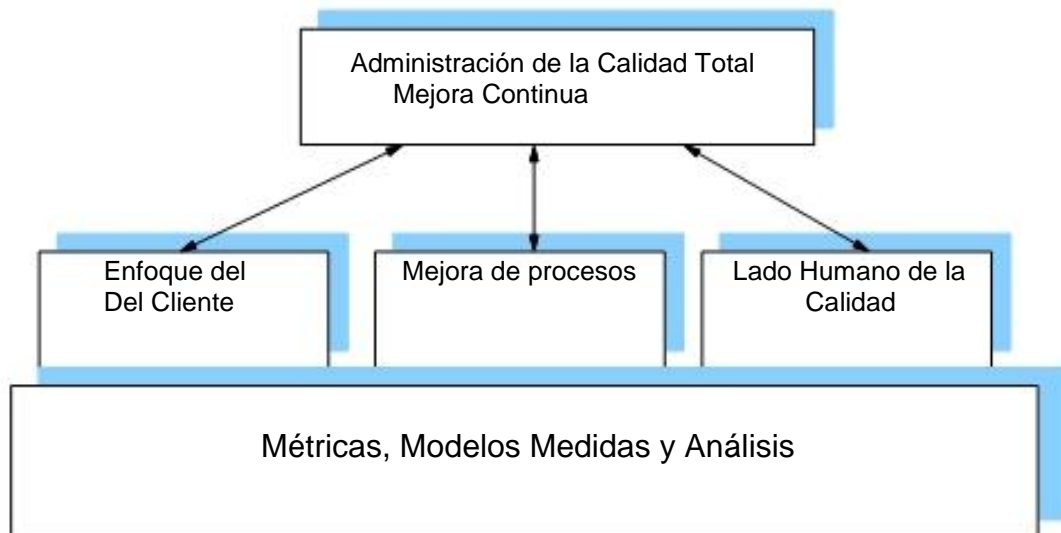


Figura 2.1: Ideas principales de la administración global de calidad

2.5. Concepto de Calidad de Software

Mientras la calidad en general puede considerarse como un concepto entendible, en la práctica, la definición de calidad de un sistema puede ser vaga. Para saber cuando un sistema encuentra sus requerimientos de calidad necesitamos realizar un juicio objetivo. El concepto de calidad de software puede tener diferentes enfoques. El enfoque de un usuario final ó cliente y el del diseñador del software pueden ser distintos. El objetivo principal es que se logre llegar a un punto en común. En un esfuerzo por definir el concepto de calidad, algunos autores argumentan que una especificación o atributo de calidad afecta el cómo se obtienen mejoras en el funcionamiento de la operación de un producto de software.

De acuerdo a la terminología de la IEEE, la calidad de un sistema, componente o proceso de desarrollo de software, se obtiene en función del cumplimiento de los requerimientos iniciales especificados por el cliente o usuario final.

2.6. El Modelo de McCall

Las especificaciones de la calidad de un producto de software han sido objeto de trabajo de diferentes grupos, de los cuales uno de los más destacados es el de McCall. Este modelo establece tres áreas principales que intervienen en la Calidad del Software:

1. *Calidad en la operación del producto:* En general se requiere que este pueda ser entendido de manera muy fácil, que opere eficientemente y que los resultados sean los requeridos inicialmente por el usuario.

2. *Revisión de calidad del producto:* Tiene como objetivo realizar revisiones durante el proceso de desarrollo para detectar los errores que afecten a la operación del producto de software. Las revisiones involucran grupos de personas que examinan parte o todo el proceso del software, los sistemas o su documentación asociada para descubrir problemas potenciales. Las conclusiones de la revisión se registran formalmente y se pasan al autor o a quien sea responsable de corregir los problemas descubiertos.

3. *Calidad en el proceso:* Desde este enfoque se recomienda definir o seleccionar estándares y procedimientos, que sirvan como referencia durante el desarrollo de software. En la figura 2.2 se pueden apreciar la relación que guardan estos tres aspectos.

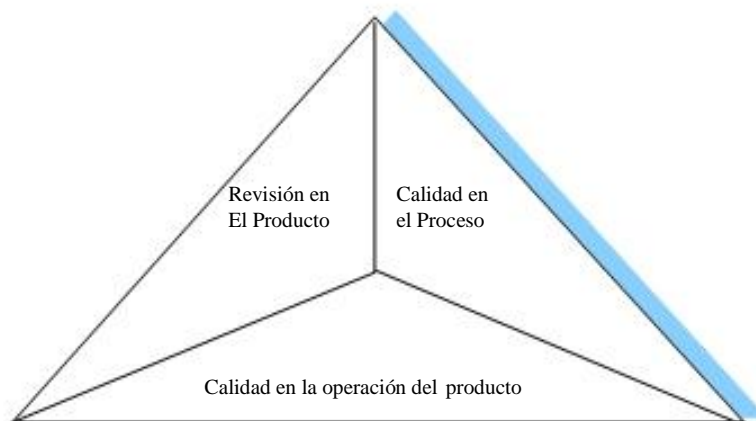


Figura 2.2: Modelo de McCall de calidad de software

2.7. El Modelo de Boehm

Otro modelo que es importante resaltar es el de Boehm. Este modelo, descrito en la figura 2.3, destaca por ser uno de los mejor definidos. El modelo es de naturaleza jerárquica y los criterios de calidad se presentan en tres grandes subdivisiones. La primera división es hecha acorde a los servicios que el sistema va a ofrecer (*portabilidad*). La segunda se hace de acuerdo a la operación del producto (*usabilidad*) y la tercera gran subdivisión se hace de acuerdo a la *mantenibilidad* del producto de software.

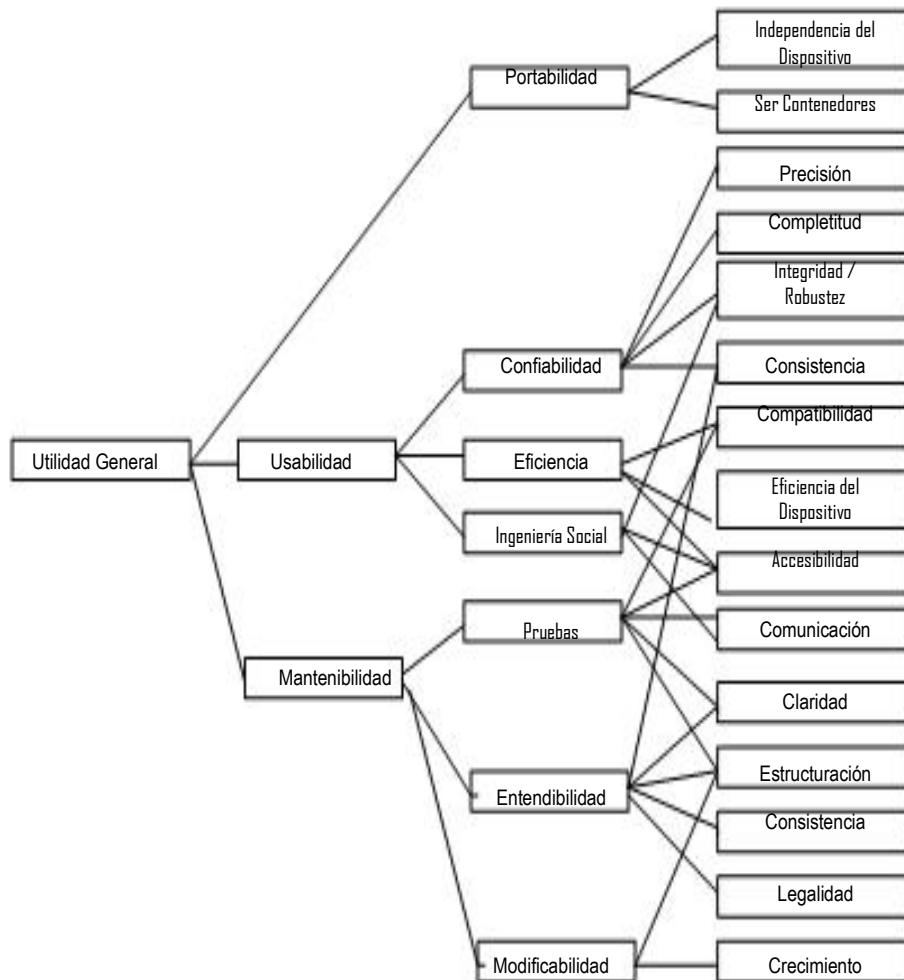


Figura 2.3: Modelo de Boehm para clasificar los criterios de calidad

Entre los criterios básicos del modelo podemos mencionar:

- *Usabilidad*: Este atributo de calidad de software se enfoca a mejorar la simplicidad, entendibilidad y facilidad de uso de un sistema de software para un cliente o usuario final.
- *Mantenibilidad*: Tradicionalmente se define como el esfuerzo requerido para localizar y especificar un error en la operación de un módulo, función o sistema de software.
- *Portabilidad*: Se define como el esfuerzo requerido para transportar la configuración de hardware y/o software, es decir, un módulo, función o sistema de software en el ambiente de una plataforma a otra.
- *Confiabilidad*: La *confiabilidad* de un sistema de cómputo es una propiedad que implica el grado de confianza esperado por parte del usuario en la operación adecuada del sistema al utilizarlo. La *confiabilidad* es afectada por cuatro aspectos fundamentales:
 - *Disponibilidad*: Define la probabilidad de que el sistema esté funcionando en un tiempo determinado.

- *Fiabilidad*: Es la probabilidad de que el sistema funcione correctamente durante un intervalo de tiempo específico.
- *Seguridad*: Representa la capacidad de que el sistema no afecte su entorno y el de quien lo utiliza.
- *Protección*: Representa la capacidad del sistema para protegerse de intrusiones accidentales o programadas.

Sin embargo, la *disponibilidad*, la *seguridad* y la *protección* se ven afectadas por la *fiabilidad*. Para evaluar la *fiabilidad* debemos tomar en cuenta que los sistemas de software no son entidades estáticas. La *fiabilidad* de un sistema se complica a medida que este crece. Es posible caracterizar el comportamiento de la *fiabilidad* estudiando el comportamiento de las fallas, para lo cual, podemos considerar por ejemplo el tiempo de aparición de fallas, el número de fallas ocurrido en un tiempo determinado, o la media del tiempo de ocurrencia de fallas. En todos estos casos estamos afectados por variables aleatorias.

Podemos realizar modelos subsecuentes del comportamiento de las fallas tratándolas como parte de un proceso estocástico. Cuando el software falla es necesario localizar y reparar las fallas que causan los errores del sistema.

- *Eficiencia*: Se refiere al esfuerzo en base al costo de los recursos de la computadora que emplea un sistema, módulo ó función de software para su ejecución.
- *Ingeniería social*: Este atributo se refiere al entendimiento y aceptación del sistema de software del grupo social al cual se enfocaron los requerimientos del sistema.
- *Pruebas*: Es el conjunto de exámenes que se realizan a un sistema, modulo ó función una vez finalizado su desarrollo. Estos exámenes se realizan con el objetivo de asegurar que un sistema, modulo ó función, cumple con la especificación de los requerimientos del usuario o cliente final.
- *Entendibilidad*. Es la claridad en cuanto a la lógica del código fuente de un módulo, función o sistema de software.
- *Modificabilidad*: Este atributo está muy ligado a la Entendibilidad, debido a que si un código es entendible, como consecuencia podrá ser cambiado en su contenido sin que esto afecte negativamente al sistema en el cual se integra.

Criterios adicionales que han complementado al modelo de Boehm y que surgido en años recientes son:

- *Flexibilidad*: Representa el esfuerzo requerido para modificar el código de un sistema, módulo ó función de software en operación.
- *Reusabilidad*: El alcance para el cual un programa puede ser usado en otras aplicaciones.
- *Interoperatividad*: Se define como el esfuerzo requerido para acoplar un sistema a otro.



CAPÍTULO III.

ASPECTOS DE CALIDAD EN APLICACIONES WEB Y PROCESOS DE MEJORA CONTÍNUA

CAPÍTULO III. ASPECTOS DE CALIDAD EN APLICACIONES WEB Y PROCESOS DE MEJORA CONTÍNUA.

3.1. Criterios de Calidad

De acuerdo al estudio de Perry [1] (tal y como se describe en la figura 3.1), las marcas obscuras implican una relación inversa, mientras que las claras expresan una relación directa.

Se puede observar que la relación entre los atributos de calidad varía. Cuando los atributos de calidad mantienen una *relación inversa*, se asume que algunos atributos transgreden la operación de los otros. Por otro lado, cuando los atributos mantienen una *relación directa* se asume que es posible que estos se puedan beneficiar entre sí. Por ejemplo, si observamos el aspecto de *confiabilidad*, veremos que la *reusabilidad* le afecta; por el contrario, la *mantenibilidad* le favorece. Esto es debido a que la *confiabilidad* se refiere a la seguridad de la operación sin errores de un sistema, módulo ó función. Mientras que el objetivo de un *código reusable* es que un mismo código pueda operar en distintos ambientes de desarrollo; lo que aumenta la probabilidad de tener errores difíciles de percibir dentro del código. Por el contrario la *mantenibilidad* se enfoca a reducir el número de errores de un sistema, módulo ó función.

La conclusión del estudio es que los atributos de calidad que debe tener un producto de software dependen en gran medida del objetivo del desarrollo del producto de software, de su proceso de desarrollo y de su contexto de operación.

La interrelación que existe entre algunos atributos de calidad se analiza a continuación:

- *Integridad vs. Eficiencia (relación inversa)*: Un sistema de software posee la propiedad de *integridad* sí, los recursos manipulados por este no son alterados o destruidos por usuarios, entidades o procesos no autorizados. Al implementar el atributo de *Integridad* es necesario contemplar un número mayor de condiciones para cumplir con las reglas de integridad. Este punto incrementa el tamaño del código afectando la eficiencia. El objetivo de la *Eficiencia* es lograr que un sistema de software requiera un costo menor en cuanto a los recursos de la computadora. Sin embargo, a medida que aumenta el tamaño del código aumentan los recursos empleados para su ejecución. Por lo anterior la *integridad* afecta la operación de la *eficiencia*.
- *Usabilidad vs. Eficiencia (relación inversa)*: El objetivo del atributo de *usabilidad*, es lograr que al usuario le sea cómoda y fácil la operación del sistema de software. Desafortunadamente para lograr implementar este atributo, en algunas ocasiones se sacrifican atributos importantes como lo es la *seguridad*. Otro inconveniente es que el código crece significativamente, lo cual impacta en el número de recursos empleados para la ejecución del producto de software. Un claro ejemplo de ineficiencia y aplicaciones usables son los productos de software de Microsoft.
- *Mantenibilidad vs Flexibilidad (relación directa)*: El objetivo del atributo de *flexibilidad* es generar códigos bien estructurados que sean entendibles. Uno de los objetivos de la *mantenibilidad* es corregir los defectos que se puedan presentar en la operación de un sistema de software. Si un sistema de

software es flexible en su código, este será mucho más fácil de mantener y las modificaciones que se requieran realizar pueden localizarse con mayor facilidad. Por lo cual la *flexibilidad* beneficia a la *mantenibilidad*.

- *Portabilidad vs. Reusabilidad (relación directa)*: El *código portable* tiene por objetivo operar libre de especificaciones de ambiente. Esta es una de las características de la *reusabilidad*. La *portabilidad* de un modulo ó sistema de software le beneficia a la *reusabilidad*.
- *Correcciones vs Eficiencia (relación neutral)*. El objetivo de *corregir* un sistema de software es liberarlo de errores de operación. Sin embargo, eliminar los errores del sistema no siempre asegura que el tiempo de ejecución del sistema vaya a disminuir. La *corrección* de errores no asegura que la *eficiencia* se vea favorecida, pero tampoco le afecta.

En la figura 3.1 se muestra el análisis de Perry, el cual define las relaciones entre los criterios de calidad.

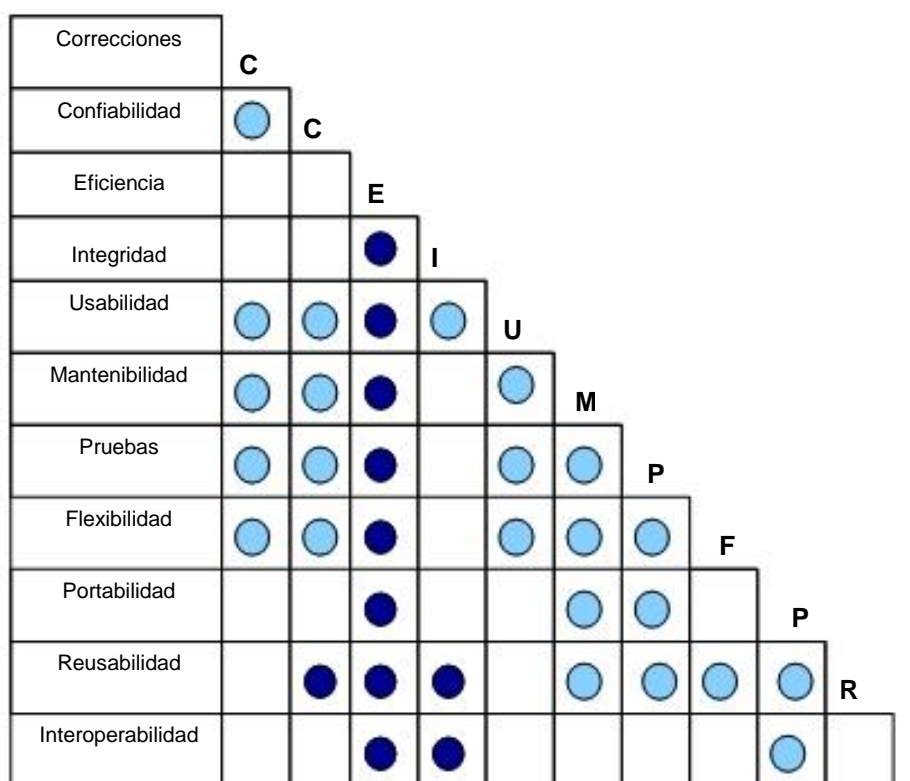


Figura 3.1: Relación entre los atributos de calidad para un producto de software.

3.2. Aspectos de Calidad en Aplicaciones Web

El Word Wide Web (WWW) fue originalmente diseñado para presentar información mediante una apariencia sencilla, con sitios sin complicaciones significativas ya que su consistencia era primordialmente de texto e hipervínculos. Las aplicaciones que existen hoy en día cuentan con una gran variedad de lenguajes para crear aplicaciones de comercio electrónico, información distribuida, entretenimiento, trabajo en grupo y encuestas, entre otras actividades.

Existen hoy en día una gran diversidad de arquitecturas heterogéneas y distribuidas donde pueden ejecutarse y prestar estos servicios.

El software de aplicaciones Web puede ser distribuido, con una implementación en múltiples lenguajes y estilos, múltiples interfaces de usuario, integración de múltiples sitios y de Bases de Datos. Podemos apreciar en la figura 3.2 la arquitectura de un sitio Web moderno.

Los componentes que integran este tipo de aplicaciones incluyen software tradicional (navegadores como Explorer, Netscape), software contemporáneo (servidores de Web como Apache), lenguajes que son interpretes de guiones (Java Script), archivos HTML, programas que son compilados (programas en Lenguaje Java), Bases de Datos (Oracle, MySQL, Informix, DB2), imágenes gráficas (jpg, gif) y complejas interfaces de usuario (por ejemplo, presentaciones en Flash de Macromedia).

El desarrollo de un sitio Web requiere de un gran equipo de personas con diferentes perfiles y conocimientos. Estos equipos incluyen programadores, diseñadores gráficos, Ingenieros en usabilidad, especialistas en integración de la información, expertos en redes y administradores de Bases de Datos. Esta diversidad de perfiles en cuanto al personal requerido es necesaria para iniciar un sitio Web en la actualidad.

Cuando desarrollamos una aplicación Web, se pueden encontrar diversos problemas. Una de las preocupaciones fundamentales, es la integración y adaptación de los elementos que las conforman. Lo cual implica mucha de la complejidad que interviene en su desarrollo. En general el tamaño en cuanto a líneas de código para este tipo de software; normalmente se estima en varios miles o incluso millones de líneas de código. Esto es debido a que su enfoque primordial es proveer servicios de información. Además se maneja una gran cantidad de registros de datos, lo cual puede implicar desde un pequeño manejador de bases de datos hasta una infraestructura de DatawareHouse. Por lo expuesto con anterioridad las organizaciones demandan una mayor calidad para este tipo de productos de software.

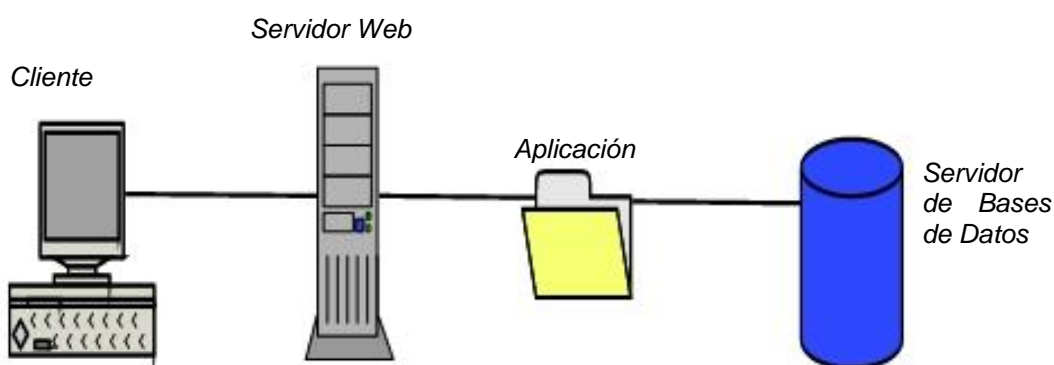


Figura 3.2: Arquitectura de un sitio Web.

Una de las ventajas que ofrece el software basado en la Internet es la reducción de los costos. Esto es debido a que los procedimientos de la organización pueden ser automatizados y administrados por la red. Sin embargo la información almacenada depende de la capacidad de los equipos de cómputo. Algunas veces se construyen complicadas aplicaciones de negocio, que generan grandes flujos de datos a través de la red (por ejemplo las aplicaciones bancarias).

Debido a la importancia que la calidad de software que la Internet ha despertado en los últimos años; la Conferencia Internacional de la Ingeniería de Software del año 2002 (ICSE 2002), se centró en los aspectos de Calidad para las Aplicaciones Web. En esta conferencia se concluyó que los criterios de calidad más importantes a los que puede aspirar un producto de software de esta naturaleza son los siguientes:

- *Confiabilidad.*
- *Usabilidad.*
- *Seguridad.*

Así mismo, en la conferencia ICSE 2002 se definieron otros criterios de calidad para estos tipos de sistema:

- *Disponibilidad.*
- *Escalabilidad.*
- *Mantenibilidad.*

3.2.1. Confiabilidad:

Es la probabilidad de que el sistema funcione correctamente durante un intervalo de tiempo. El número de usuarios de sitio Web es muy grande y las expectativas en cuanto a la *fiabilidad* acerca de las aplicaciones Web son muy grandes porque a través de estas se manejan pedidos, compras, ventas ó simplemente despliegue de información importante para los usuarios. Un inadecuado funcionamiento en estos productos de software no solo elimina la confianza de los usuarios sino también puede generar grandes pérdidas económicas.

3.2.2. Usabilidad

Tomas Powell describe que la *usabilidad* es una de las características más importantes para hacer un sitio atractivo para los usuarios y el cual puedan ser aceptado e integrado a sus actividades cotidianas. El objetivo de la *usabilidad* es mejorar la simplicidad, entendibilidad y facilidad de uso del sitio web. La percepción de la *usabilidad* es influenciada por las características de los usuarios, como son la edad, el nivel educativo, el perfil tecnológico, etc. La percepción de esta característica se ve afectada por las diferencias culturales asociadas, como por ejemplo el diseño gráfico, el uso de los colores y el contenido de la información. La relación de algunos de los elementos que intervienen se presentan en la figura 3.3.

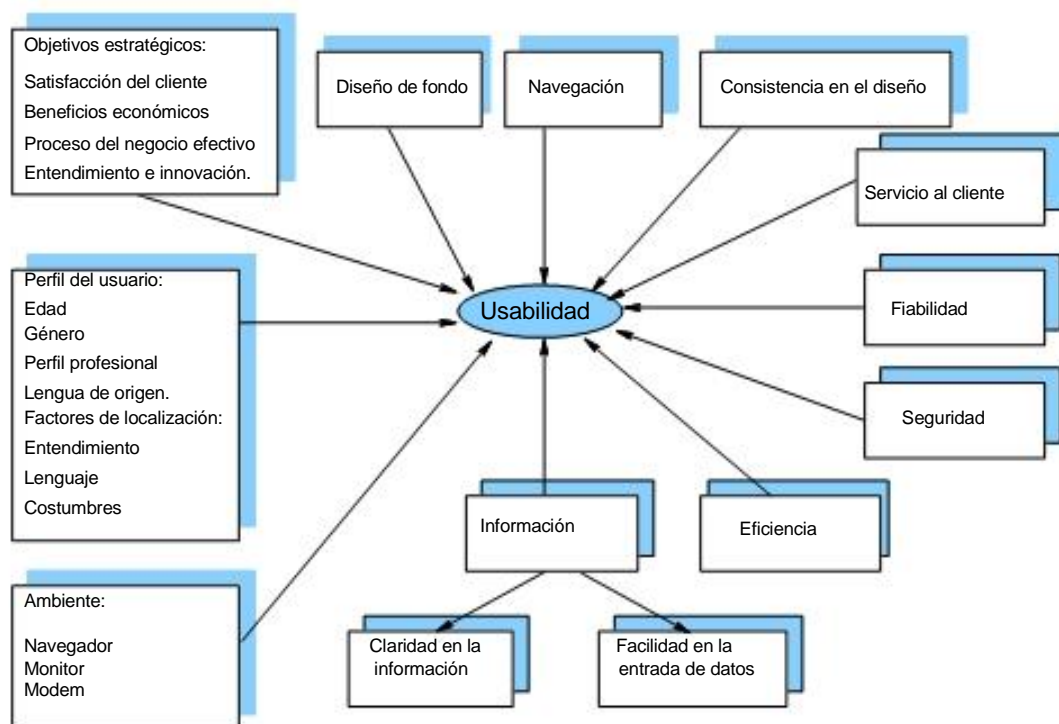


Figura 3.3: Elementos que intervienen en la usabilidad.

3.2.3. Seguridad

La Seguridad representa la capacidad de que el sistema no afecte su entorno y el de quien lo utiliza. Existen muchos sistemas de software en Internet que trabajan con usuarios mediante el acceso personalizado, por ejemplo las aplicaciones bancarias. Esto conlleva la introducción de una serie de medidas de seguridad implementadas dentro de las aplicaciones y de los servicios. En caso de no aplicar medidas de seguridad adecuadas, no sólo existe el riesgo de perjudicar la imagen de las organizaciones, si no también esta latente el riesgo de afectar la economía de los usuarios del sistema de software. En la actualidad existen compañías que tienen grandes pérdidas económicas, debido principalmente a la falta o a la implementación inadecuada del aspecto de calidad *seguridad*.

3.2.4. Disponibilidad

Este atributo, define la probabilidad de que el sistema esté funcionando por un tiempo determinado. Sobre un sitio Web, los clientes no únicamente esperan disponibilidad las 24 horas del día, los siete días en la semana durante todo el año. Sólo una pérdida de tiempo de 10 minutos puede ocasionar grandes pérdidas, por ejemplo en las aplicaciones de la bolsa de valores. En otros casos las pérdidas pueden ser de ventas de diferentes artículos, como libros (es el caso de Amazon), revistas, etc. La disponibilidad también implica que todos los usuarios puedan acceder a la información, no importando el navegador y la plataforma que utilicen. En el caso *Shockwave Flash* el cual es únicamente compatible con Microsoft Internet Explorer y Netscape sobre Windows, los usuarios que trabajan sobre Unix no podrán acceder a la información que es manejada por este software. En este caso, también

está latente la poca disponibilidad del sistema. Es claro que el hecho de hacer disponible un sitio desde todos los puntos de vista, implica adoptar lenguajes más robustos y estables, aunque la facilidad para desarrollar estas aplicaciones se vea reducida.

3.2.5. Escalabilidad

Las necesidades para la escalabilidad han derivado una gama muy amplia con la que se cuenta en la actualidad de tecnología de innovación en cuanto a software y hardware. La industria ha desarrollado nuevos lenguajes de software, estrategias de diseño, tecnologías de comunicación de datos y transferencia de protocolos. La necesidad surge cuando el número de clientes ó usuarios que utilizan los servicios de una aplicación en Web crece. Con el objetivo de evitar el riesgo de insuficiencia de recursos en el sistema, normalmente se programan crecimientos en la infraestructura.

Cuando un sistema de esta naturaleza esta preparado para crecer, lo hace sin complicaciones.

Este aspecto debe ser contemplado desde la planeación, análisis y el diseño, dentro del proceso de desarrollo de software.

3.2.6 Mantenibilidad

Un aspecto novedoso que deriva del software basado en la Internet, son las nuevas versiones o las actualizaciones. El software tradicional involucra mercadeo, precios, ventas y personal de instalación para los clientes. Sin embargo, este proceso es caro y el desarrollo de este software usualmente requiere modificaciones de mantenimiento, con sus estimados de tiempo. Es el caso, por ejemplo, de las múltiples versiones que manejan, las aplicaciones de Microsoft. El mantenimiento que se requiere para aplicaciones basadas en el web muchas veces debe ser inmediato. El acceso de los clientes es algo que implica una actualización constante en cuanto a información y el servicio a los módulos. Los cambios que se realizan en mucho de los casos deben ser inmediatos y transparentes al usuario. En ocasiones extremas las aplicaciones deben ser actualizadas sin dejar de proporcionar el servicio. Por citar algunos ejemplos tenemos las aplicaciones que manejan ventas por Internet, las aplicaciones que manejan acciones de alguna casa de bolsa, ó las aplicaciones bancarias donde se realizan transacciones en línea.

3.3. Proceso de Medición de Productos de Software

La medición del software se refiere a derivar un valor numérico para algún atributo de un producto de software o un proceso del software. Comparando estos valores entre ellos y con los estándares aplicados en la organización, es posible sacar conclusiones de la calidad del software o de los procesos de software.

Existe una reticencia en las compañías que desarrollan software para introducir medidas debido a que los beneficios no son claros. Una razón de esto es que, en muchas compañías los procesos del software utilizados aún están pobremente organizados y no son suficientemente maduros como para utilizar dichas medidas. Otra razón es que no existen estándares para las métricas y por lo tanto, es difícil

llevar a cabo la recolección y el análisis de datos. En la actualidad, muchas compañías no están preparadas para introducir mediciones sino hasta que existan disponibles tales estándares y herramientas.

A menudo es imposible medir los atributos de calidad del software de forma directa. Los atributos como la *fiabilidad*, *mantenibilidad* o la complejidad, por citar algunos ejemplos, se ven afectados por diversos factores y no existen métricas directas para ellos. Los esfuerzos se orientan a medir algún atributo interno del software (como el número de defectos) y suponer que existe una relación entre lo que se puede medir y lo que se quiere saber. El proceso de medición es de las actividades más importantes ya que de este depende que los resultados puedan ser fiables para poder emprender posteriormente un análisis objetivo.

Los pasos que normalmente se siguen en el proceso de medición son:

- Seleccionar las medidas a realizar.
- Seleccionar los componentes a evaluar.
- Medir las características de los componentes.
- Identificar las mediciones anómalas.
- Analizar los componentes anómalos.

3.4. Métricas de Software

Una métrica de software es cualquier tipo de medida relacionada con un sistema, proceso ó documento de software. Las métricas a emplear dependen del atributo de calidad a evaluar y de las condiciones de desarrollo y operación del sistema. Los atributos de software más comunes son los mostrados en las figuras 2.3 y 3.1. Para cada atributo es posible que existan varios tipos de métricas de software que se pueden aplicar. En algunos casos las métricas de software existentes no son aplicables al ambiente de operación del proyecto, o estas son difíciles de obtener. En estos casos es posible implementar nuevas métricas que estén de acuerdo a las normas de calidad de la organización. Compañías como Motorola, IBM y Hewlett Packard han desarrollado métricas adecuadas a su marco de producción.

3.4.1. Características de las métricas de software

Las métricas deben cumplir con ciertos puntos tales como:

- *Nombre*. El identificador de la métrica que pueda ser conocido.
- *Definición*. La descripción de los atributos de las entidades que pueden medirse utilizando la métrica. Debe describirse como se calcula y cuál es su valor por defecto.
- *Objetivo*. Enumera los objetivos que pueden ser alcanzables y las respuestas que se pueden obtener mediante dicha métrica. Así como la justificación de la importancia de la métrica.
- *Procedimiento de análisis*. Aquí se describe cómo se entiende el uso de la métrica. Las precondiciones bajo las cuáles actúa para obtener una

interpretación adecuada de los valores de estas. Es necesario contar con técnicas de análisis y herramientas para el modelado.

- *Responsabilidades.* Este punto se refiere a que siempre debe existir un responsable de coleccionar, registrar los datos de las medidas, preparar los reportes y analizar los datos.

3.4.2. Tipos de métricas de software

Las métricas de software pueden clasificarse en tres categorías:

- Métricas de producto.
- Métricas de proyecto.
- Métricas de proceso.

Las *métricas del producto* describen las características del producto como son el tamaño, la complejidad, las características de diseño y los atributos de calidad. Las *métricas del proceso* pueden ser utilizadas para mejorar el desarrollo y el mantenimiento del software. Las *métricas del proyecto* describen características administrativas y su ejecución, como son costo, planeación, productividad del personal, etc.

Las métricas que nos interesan en esta tesis son aquellas que nos pueden ayudar a evaluar la calidad. Dichas métricas son un subconjunto de las del producto y el proceso.

Las métricas de calidad están subdivididas en:

1. *Métricas de Producto final*
2. *Métricas del proceso de desarrollo*
3. *Métricas del mantenimiento del software.*

Las primeras dos contemplan niveles que se relacionan con la calidad intrínseca del producto y la satisfacción del cliente con respecto al producto. La tercera división esta en función del mantenimiento durante el ciclo de vida esperado para el producto de software.

3.4.3. Métricas de producto final

Las Métricas de Producto Final más importantes para nuestro estudio son las siguientes:

1. *La media del tiempo de ocurrencia de fallos.* Se refiere al promedio del tiempo que tarda en producirse un error durante la operación de un producto de software.
2. *Densidad de defectos.* El concepto se refiere al número de errores que se ejecutan en un lapso de tiempo, durante la operación del producto de software.
3. *Problemas detectados por el usuario.* Estas métricas se utilizan con mayor frecuencia dentro de las empresas. Consiste en que el usuario pruebe por un tiempo

determinado, el producto de software. La cantidad de tiempo para realizar las pruebas varía de seis meses a sólo un mes, dependiendo del criterio de cada organización; los problemas una vez detectados se reportan.

Esta métrica es un tanto subjetiva, ya que depende de la experiencia, habilidad y enfoque personal de los usuarios.

4. Niveles de satisfacción del cliente. Esta métrica se refiere a la satisfacción que el cliente encuentra con el producto de software. Los niveles que se manejan son los siguientes:

- Muy Satisfecho - 100 %.
- Satisfecho - 75 %.
- Neutral - 50 %.
- Insatisfecho - 25 %.
- Muy Insatisfecho - 0 %.

La calidad intrínseca de un producto puede medirse por el número de defectos del producto durante su operación ó por el tiempo que tarda en presentarse un error. Las métricas citadas con anterioridad están enfocadas a los errores que ocurren durante la operación de un software, por lo cual, es necesario describir el concepto de error.

De acuerdo a la IEEE/ANSI (Instituto Nacional de Estándares de los E.E.U.U.) se manejan las siguientes definiciones de error:

- Un error es un olvido humano que se visualiza en un resultado incorrecto de un software.
- Un fallo es el resultado de una condición accidental que causa una unidad del sistema, cuando es requerida para el funcionamiento del sistema.
- Un defecto es una anomalía en un producto.
- Una falla ocurre cuando una unidad funcional de un software relacionado con un sistema no tiene la eficiencia necesaria que requiere el funcionamiento de este, debido a los límites especificados para la unidad.

3.4.4. Métricas del proceso de desarrollo

Las Métricas del proceso de desarrollo están orientadas a medir la calidad en el ciclo del proceso de desarrollo. La idea principal es entender que dentro del proceso de desarrollo las métricas juegan un papel muy importante debido a que a través de ellas es posible cuantificar y evaluar el acercamiento hacia el comportamiento esperado, sin que sea necesario esperar hasta que el producto finalice para realizar esta evaluación. Estas métricas son menos formales que las del producto. Mientras que algunas organizaciones sólo toman en cuenta un tipo de métricas, otras contemplan diversos tipos de métricas de acuerdo a la fase del proyecto.

Las principales métricas del proceso de desarrollo son las siguientes:

Densidad de defectos durante el mecanismo de pruebas. La velocidad del arribo de defectos es el indicador de que el software está experimentando una alta incidencia de errores durante el proceso de desarrollo. Una densidad de defectos nunca sigue una distribución uniforme.

Patrón de arribo de defectos durante el mecanismo de pruebas. El conjunto de pruebas para la densidad de defectos son un indicador global de la *confiabilidad* del sistema. El patrón está referido a semanas o ocasionalmente a meses.

Patrón basado en la fase de remoción de defectos. Este patrón de métricas es una extensión de la métrica de densidad de defectos. El agregado de este patrón consiste en el rastreo de los defectos para todas las fases del ciclo de desarrollo, incluyendo la revisión del diseño, inspección del código y verificaciones formales previas a las pruebas.

Efectividad en la eliminación de defectos. Esta métrica está definida mediante la siguiente expresión:

$$DRE = DRFD/DLP * 100\%$$

Donde, DRFD= Defectos removidos durante la fase de desarrollo, DLP=Defectos latentes en el producto y DRE = Efectividad en la remoción de defectos.

3.4.5. Métricas del mantenimiento de software

Cuando el desarrollo de un producto de software se termina y entra en ciclos de revisión, se puede decir que la fase de mantenimiento comienza un ciclo de vida. Durante esta fase, los defectos que tiene el producto de software son reportados por el cliente y estos son corregidos durante las revisiones.

Algunas métricas de mantenimiento del software son las siguientes.

Índice de bitácoras administrativas (BMI).

$$BMI = NPCM / NPADM * 100\%$$

Donde, NPCM = Número de problemas cerrados durante el mes, y NPAM = Número de problemas que llegan durante el mes.

Correcciones y tiempos de corrección. Es el tiempo promedio en que se realiza la corrección de un defecto de software desde su detección hasta su saneamiento.

Porcentaje de correcciones defectuosas (PDF).

$$PDF = NFDST / NFERTCSN * 100\%$$

Donde, NFERTCSN = Número de correcciones que exceden el criterio del tiempo de acuerdo a nivel especificado por el proceso y NFDST = Número de correcciones liberadas en un tiempo determinado.

Correcciones de calidad. Esta métrica está enfocada a la satisfacción del cliente con respecto a los productos entregables.

3.5. Procesos de Mejora Continua

3.5.1. Introducción

La mejora de la calidad de un producto de software es compleja debido a que implica revisar los procesos de desarrollo de software, las actividades que intervienen en ellos y como se están realizando.

Modelos como el Modelo de Maduración de la Capacidad, proporcionan un enfoque para evaluar las capacidades de los elementos que integran una organización que desarrolla software. Para mejorar las cualidades de calidad de un producto de software está implícito el factor humano. El enfoque de los procesos de mejora continua permite elevar la calidad de los productos mediante la mejora de las prácticas personales de los ingenieros que desarrollan software dentro de una organización. PSP es un ejemplo de ello.

El Proceso de Software Personal (PSP) se enfoca a la mejora de las prácticas individuales mediante la valoración de las capacidades de los Ingenieros de Software y a través de la evaluación de sus tiempos de su productividad y eficiencia. De esto deriva un análisis que los propios Ingenieros realizan. De este análisis, aprenden a medir sus capacidades.

PSP proporciona una secuencia de actividades que les ayudan a mejorar la forma en como realizar su trabajo. El proceso de evaluación-análisis, se realiza durante todo el Proceso, con lo cual es posible que los ingenieros valoren sus avances.

Sin embargo, aplicar Procesos de Mejora Continua no es fácil, ya que podría existir mucha resistencia al cambio de actitudes y de enfoques para desarrollar software. Sobre todo en aquellas organizaciones donde no se tiene bien claro el objetivo que se persigue, donde no se tiene un proceso de desarrollo de software maduro y en donde no le dan la importancia al hecho de laborar con eficiencia, generando productos de calidad.

3.5.2. Modelos para el Proceso de Desarrollo de Software

El modelo del proceso de desarrollo elegido por parte del administrador es muy importante, debido a que marcará la pauta para la planeación y organización del proyecto de software. La elección dependerá de las características del modelo. Así por ejemplo, el modelo de cascada es oportuno si se conocen las fases del proyecto, debido a experiencias anteriores. De lo contrario, podría resultar inconveniente, si no se tiene experiencia en este tipo de desarrollo. Bajo este ambiente puede convenir el modelo del prototipado.

Los modelos de mayor uso para este fin son los siguientes:

Modelo de cascada. Este modelo, descrito en la figura 3.4, toma las actividades fundamentales del proceso: especificación, desarrollo, validación y evolución y los representa como fases separadas del proceso, como la especificación de requerimientos, el diseño de software, la implementación, las pruebas, y el mantenimiento.

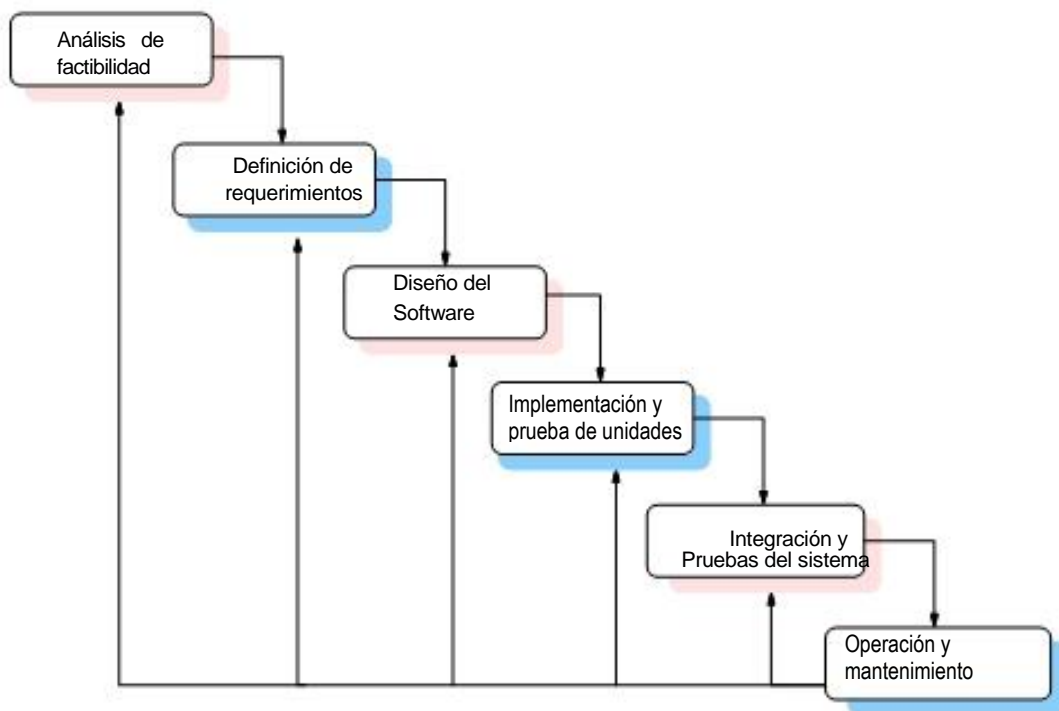


Figura 3.4: Modelo de cascada

Desarrollo evolutivo ó incremental. Este enfoque, descrito en la figura 3.5, se entrelazan las actividades de especificación, desarrollo y validación. Un primer sistema se desarrolla rápidamente a partir de especificaciones abstractas. Entonces se ajusta con ayuda del cliente, para producir un sistema que satisfaga sus necesidades.

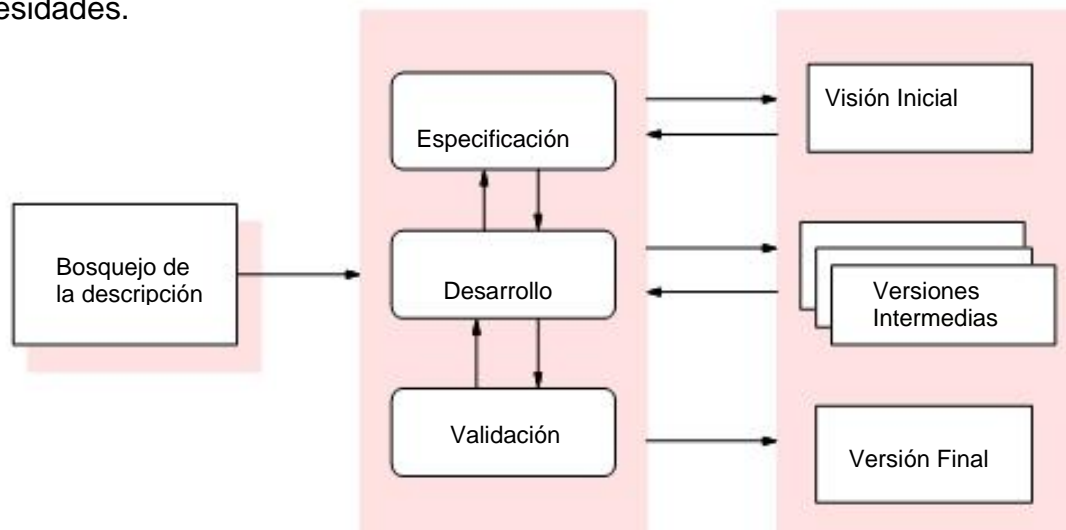


Figura 3.5: Modelo de desarrollo evolutivo.

Desarrollo formal de sistemas. Este enfoque se basa en la producción de una especificación matemática formal y en la transformación de esta especificación utilizando métodos matemáticos, para construir los programas.

Desarrollo basado en la reutilización. Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca a integrar estos componentes en el sistema más que en desarrollarlos desde cero.

El Modelo de espiral. El modelo en espiral del proceso del software, que originalmente fue propuesto por Boehm (1988), hoy se conoce ampliamente. Más que representar el proceso de software como una sucesión de actividades con retrospectiva de una actividad a otra, se representa como espiral. Así, el ciclo más interno podría referirse a la *factibilidad* del sistema, el siguiente ciclo a la definición de requerimientos del sistema, el siguiente ciclo al diseño del sistema y así sucesivamente.

3.5.3. El Modelo de Madurez de la Capacidad del Proceso (CMM)

El CMM (Capability Maturity Model) ó Modelo de Madurez de la Capacidad del Proceso (CMM) fue creado por el Instituto de Ingeniería de Software de la Universidad de Carnegie Mellon. Este modelo está compuesto de varios niveles que permite evaluar la madurez y capacidad de las empresas que desarrollan software. Debido a las características que tiene implícitas el modelo, es posible utilizarlo para la mejora de los procesos de desarrollo de software.

El CMM está compuesto de cinco niveles, que intervienen en las diferentes facetas del desarrollo de procesos.

Los Niveles del CMM (que se muestran en la figura 3.6), son los siguientes:

1. Nivel inicial. En este nivel la organización no tiene procedimientos de administración efectiva o planes de desarrollo de sus proyectos de software. Aunque en la organización podrían existir los elementos formales para el control del proyecto, no existen mecanismos organizacionales para asegurar que se utilizarán en forma consistente. El desarrollo del software puede ser exitoso, pero las características como la calidad y los procesos (costos, duración, desempeño) son impredecibles en la mayoría de los proyectos. En este nivel, la eficiencia en el desarrollo de proyectos por lo general se mejora mediante la utilización de métodos efectivos para el control y administración del proyecto.

2. Nivel repetible. En este nivel, una organización tiene procedimientos formales de administración del proyecto, de aseguramiento de la seguridad y de control de la configuración. Se denomina repetible debido a que la organización puede repetir proyectos del mismo tipo de forma exitosa y por que en cada repetición los proyectos podrán ser mejorados y desarrollados con un menor coste. Sin embargo, hay una falta de un modelo de procesos formal. El éxito del proyecto depende de los administradores individuales que motivan al equipo y del ambiente y procesos de la organización.

La principal diferencia entre el nivel inicial y el nivel repetible es que el nivel repetible provee un control de la planeación del proyecto y los compromisos establecidos. La experiencia en proyectos similares es prioritaria para el éxito de los proyectos, porque se logra un mejor control sobre los costos, planes y cambios en los requerimientos.

En este nivel, además de lograr dominar las actividades definidas en el nivel inicial, se consiguen los siguientes aspectos:

- *Administración de la configuración del software.* En esta actividad se establece y mantiene la integridad en todos los puntos clave del ciclo completo del proyecto. Así mismo, en esta actividad se consigue la identificación de los puntos de configuración, las líneas de producción y los procedimientos para el ciclo de control de cambios.
- *Aseguramiento de la calidad del software.* Esta actividad involucra revisiones y auditorías de los procesos y productos para validar que están cumpliendo con niveles aceptables (de acuerdo a los estándares adoptados).
- *Administración de la sub-contratación del software.* En los proyectos de software muy grandes, con frecuencia es necesario sub-contratar grupos de trabajo para desarrollar software que no es posible desarrollar dentro del proyecto. La planeación para las actividades de los subcontratistas es necesaria y el progreso de estas siempre debe ser planeada.
- *Seguimiento y vigilancia del proyecto de software.* Esta actividad está relacionada con la visibilidad del progreso actual. Los resultados intermedios tienen que ser revisados y vigilados con respecto al plan de actividades. Cuando sea necesario, este plan de actividades tiene que ser re-evaluado y re-diseñado buscando la *fiabilidad* del proceso y del proyecto.
- *Planeación de proyectos de software.* Esta actividad toma en cuenta la creación de los planes para la ejecución y administración del proyecto. En la planeación, un grupo de personas con la responsabilidad de entregar el proyecto de forma exitosa genera estos planes y su documentación. La planeación de proyectos se refiere a la identificación de actividades, etapas, su calendarización y las entregas producidas por un proyecto. Esto implica hacer el bosquejo de un plan para guiar el desarrollo hacia las metas del proyecto. La estimación del costo es una actividad que se refiere al estimado de los recursos requeridos para llevar a cabo el plan del proyecto. La administración efectiva de un proyecto de software depende de la planeación completa de todo el progreso del proyecto. El administrador debe tener la capacidad de anticiparse a los problemas que puedan surgir y sugerir alternativas de solución. Un plan que se prepara al inicio de un proyecto debe utilizarse como guía para la realización del mismo. La calendarización del proyecto implica separar todo el trabajo de un proyecto en actividades complementarias y considerar el tiempo requerido para completar dichas actividades. Se puede dar el caso en el cual algunas de estas actividades puedan realizarse en paralelo. En la estimación de las actividades, los administradores deben tomar en cuenta que pueden surgir problemas, por los cuál se debe considerar el tiempo en el cual pueda existir algún imprevisto. Por lo general, el calendario de proyecto se representa como un conjunto de gráficos que muestran la división del trabajo, las dependencias de las actividades y la asignación del personal.
- *Administración de requerimientos.* Esta actividad involucra la estabilidad y mantenimiento de los acuerdos con el cliente con respecto a los requerimientos del sistema de software. El software sufrirá cambios de

manera inevitable, pero el control y la documentación en estos casos es de vital importancia.

3. Nivel definido. En este nivel, una organización tiene definidos sus procesos, por lo que tienen una base para la mejora cualitativa de procesos. Existen procedimientos formales que aseguran que el proceso se sigue en todos los proyectos de software. En este nivel, además de lograr dominar las actividades definidas en los niveles 1 y 2, se consiguen los siguientes aspectos:

- *Revisión de puntos.* En esta actividad se busca identificar errores en el desarrollo mediante inspecciones y revisiones del proceso de desarrollo.
- *Coordinación interna de grupos.* En esta actividad lo mas importante es la integración entre el grupo de Ingenieros de software y otros grupos, como son el grupo de aseguramiento de calidad, los usuarios finales o representantes y los administradores del proyecto. Esta coordinación es importante, ya que permitirá poner de acuerdo a distintos grupos y tomar acuerdos.
- *Ingeniería de productos de software.* En esta actividad se proyecta el proceso definido en la actividad anterior para mejorar y adecuar todas las actividades del desarrollo, como son: mantenimiento y documentación de los requerimientos, diseño y codificación, planeación, ejecución de pruebas y en general la consistencia del trabajo.
- *Administración integrada del software.* Esta actividad involucra el desarrollo de un proyecto específico en cuanto a la línea de proceso que proviene del proceso estándar. Dado que en muchas ocasiones los procesos de software son comunes para distintos proyectos desarrollados, la experiencia que se adquiere puede ser valiosa en futuros desarrollos.
- *Programa de capacitación.* El propósito del programa de capacitación es desarrollar los perfiles y habilidades necesarias para los integrantes del grupo de trabajo con el propósito de hacer mas eficientes sus roles. La capacitación es a nivel organizacional, a nivel de los proyectos y de manera individual.
- *Definición del proceso de la organización.* En esta actividad la organización desarrolla y mantienen un estándar de proceso para el desarrollo del software.
- *Enfocar el proceso de la organización.* Esta actividad involucra el establecimiento y consolidación de las responsabilidades por separado del grupo, para las actividades que intervienen en el proceso de la organización. El grupo tiene la responsabilidad de asentar en forma regular, la información del mantenimiento y de los resultados del proceso de software en una base de datos y la introducción de nuevas herramientas y tecnología.

4. Nivel administrativo. En este nivel, se tiene un proceso definido y un programa formal de colección de datos cualitativos. Recolecta las métricas del proceso y del producto para alimentar la actividad de mejora de procesos. En este nivel, además de lograr dominar las actividades definidas en los niveles anteriores, se consiguen los siguientes aspectos:

- *Administración de la calidad del software.* En este punto la idea principal es aumentar la calidad del producto. Las normas de calidad son monitoreadas y revisadas desde el inicio hasta el término del producto.
- *Administración del proceso cuantitativo.* En este punto se hace una medición del proceso, se hace un análisis de estas medidas y finalmente se hace los ajustes necesarios para implementar la mejora de los procesos.

5. Nivel optimizado. En este nivel, una organización está comprometida con la mejora continua de procesos. La mejora se calcula y planea como parte integral de los procesos de la organización.

En esta etapa el énfasis para la optimización ha cambiado del producto al proceso. En este nivel, además de lograr dominar las actividades definidas en los niveles anteriores, se consiguen los siguientes aspectos:

- *Prevención de defectos.* Envuelve la identificación de las causas comunes de defectos, con lo que también se cubre la prevención. Esta actividad se realiza de manera periódica.
- *Administración del cambio de la tecnología.* Esta actividad ve específicamente la identificación de nuevas tecnologías y su transición ordenada dentro de la organización.
- *Administración de cambios en el proceso.* La mejora continua de procesos de forma ordenada se aplica para la mejora de la calidad de los productos, la productividad de la organización y la reducción del tiempo necesario para el desarrollo de los productos.

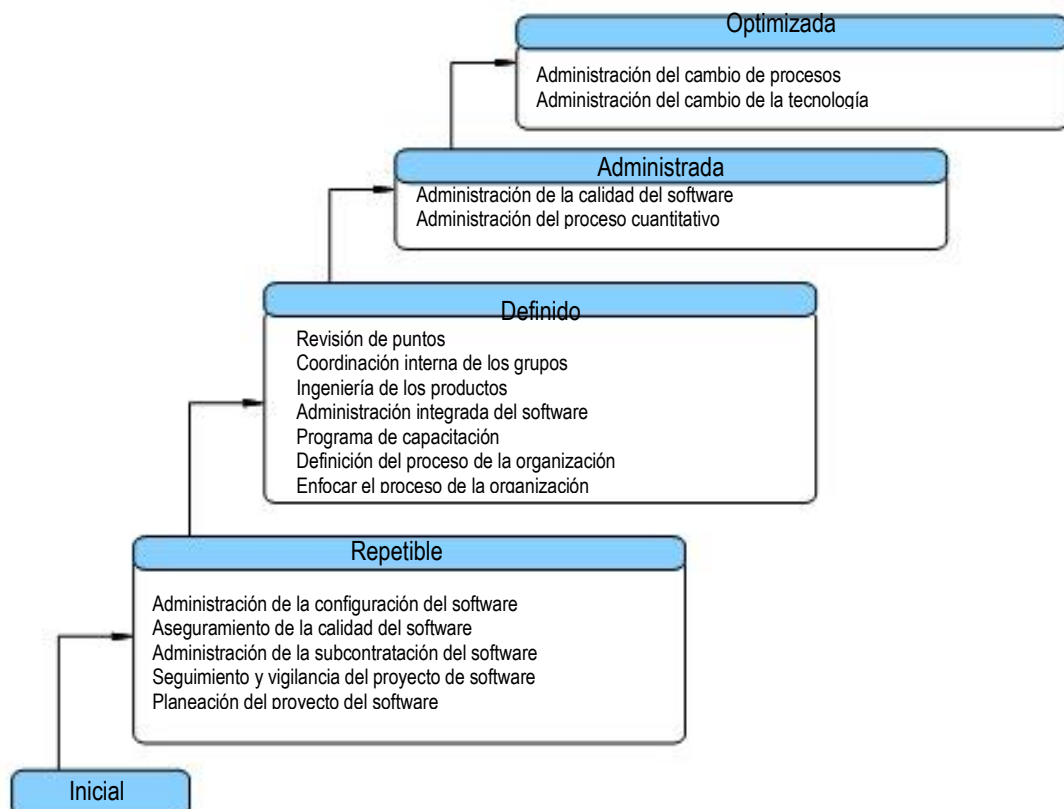


Figura 3.6: Niveles del Modelo de Madurez de la Capacidad del Proceso (CMM)

En la actualidad, muchas de las organizaciones tienen la iniciativa de introducir programas para la mejora de procesos de software buscando lograr un alto nivel de maduración en sus procesos. Para corregir la eficiencia de trabajo de los ingenieros se ha desarrollado PSP. PSP está enfocado a la forma de trabajar de manera individual de los ingenieros que forman parte de los procesos de desarrollos de software de las compañías. Es un diseño personalizado de los procesos de mejora continua. Los principios básicos del CMM y del PSP son complementarios y de hecho hay puntos dentro del modelo de CMM en los que PSP apoya.

3.5.4. Proceso de Software Personal

El proceso de software personal (PSP) es un proceso que define fases para su aplicación midiendo los avances y las mejoras. Este proceso tiene un enfoque personal, lo cual ayuda a los ingenieros o personal que desarrollan software, mejorando su eficiencia a nivel personal. El impacto de uso de PSP se ve reflejado en el incremento de la eficiencia en la organización. Este proceso provee una solución para medir y analizar el desarrollo y la productividad en los proyectos paso a paso. El objetivo es disminuir los defectos en el código, mejorando la planeación y la productividad. Los procesos de mejora son difíciles porque la mayoría de las personas no aceptan fácilmente cambiar sus hábitos que a veces son ya naturales y parte de su personalidad.

El enfoque de un proceso de mejora se basa en los siguientes puntos:

- *Definir el proceso de Calidad.*
- *Medir la calidad del producto.*
- *Entender el proceso.*
- *Ajustar el proceso.*
- *Utilizar el proceso ajustado.*
- *Medir los resultados.*
- *Comparar los resultados con los objetivos planteados.*
- *Retro-alimentar y mejorar continuamente.*

PSP es un enfoque que muestra como aplicar métodos de Ingeniería de Software para el personal que desarrolla software en sus tareas diarias, con el objetivo de mejorar su trabajo. Proporciona métodos detallados de planificación de tiempos y estimación de actividades. Muestra a los Ingenieros como controlar su rendimiento frente a estos planes y explica cómo los procesos definidos guían su trabajo. PSP ha sido introducido en muchas organizaciones. Los datos reunidos en su aplicación muestran que PSP es efectivo en la mejora de la eficiencia en la planificación de las actividades del proceso de desarrollo de software y en la calidad de los productos que se obtienen.

PSP puede considerarse como una disciplina que proporciona un marco de trabajo estructurado para desarrollar habilidades personales. PSP se basa en la definición de medidas y análisis de datos. De acuerdo a este análisis se verifica en que parte del proceso hay que ejecutar los cambios. Este proceso de evaluación y análisis se realiza en todas las facetas que marca PSP.

Uno de los puntos mas importantes que se aplica con PSP es la reducción de defectos. Otro punto que es parte de este proceso, es mejorar la planeación de los tiempos para la ejecución de cada una de las actividades.

3.5.5. Etapas de PSP

Un proceso de software es una secuencia de pasos requeridas para desarrollar o mantener un producto de software.

Las etapas de PSP (que se muestran en la figura 3.7), son las siguientes:

Evaluación del personal PSP0. En este punto se inicia PSP. El primer paso es que los Ingenieros entiendan como aplicar PSP mediante la utilización de reportes escritos de su labor personal. Este reporte se realiza mediante la toma de tiempos de las actividades involucradas en el proceso de desarrollo. Otro punto importante es el registro de los defectos que se realizan durante las fases, así como de sus correcciones. Estos son datos de entrada que dan la posibilidad de llevar a cabo una evaluación objetiva. También sirven de patrón de referencia a futuro durante el progreso de la aplicación de PSP. PSP0 está compuesto de tres fases:

- Planeación.
- Proceso de desarrollo.
- Postmortem.

PSP0 ha evolucionado en la versión PSP0.1, la cual se enfoca a la evaluación del desarrollo, mediante la implementación de métricas y estándares en el código. El objetivo de este enfoque es evaluar y cuantificar las mejoras. Una de las formas que PSP propone es PIP la cual permite a quien aplica PSP registrar sus problemas, tipos e ideas. Este formato ayuda a enfocarse posteriormente a la mejora en la manera de realizar su proceso personal de desarrollo. Utilizando los formatos, es posible constatar la ayuda que proporcionan para reunir y utilizar los datos de entrada.

Planeación personal (PSP1). El objetivo de esta fase es introducirse al método PROBE. La utilización de PROBE tiene como objetivo estimar tamaños y tiempos en el desarrollo del software, tomando como referencia los datos recopilados en la fase PSP0. En este punto normalmente se utiliza regresión lineal y se estiman los parámetros. De acuerdo a la gráfica obtenida se realizan las predicciones de tamaño y tiempo estimados para el desarrollo del software. La planeación de actividades son la entrada de PSP1.1.

Calidad personal (PSP2). Introduce a la administración de defectos. En este punto los datos de entrada son los defectos con los cuales se construye y verifica un estudio para realizar el diseño y la revisión del código. Las *listas de verificación* son utilizadas para realizar las revisiones y mejorar su codificación. Las listas de verificación son hechas de acuerdo a sus perfiles y prácticas personales.

PSP2 ha evolucionado en PSP2.1. En este punto se introducen especificaciones de diseño y técnicas de análisis a lo largo de las fases con el objetivo de prevenir la implementación de defectos. Los procesos de análisis son nuevamente los puntos de referencia. El análisis deriva de la evaluación de los tiempos que duran sus actividades y del número de defectos que introducen y remueven en cada una de las

fases del proceso. Quién aplica PSP aprende a evaluar y mejorar su eficiencia personal.

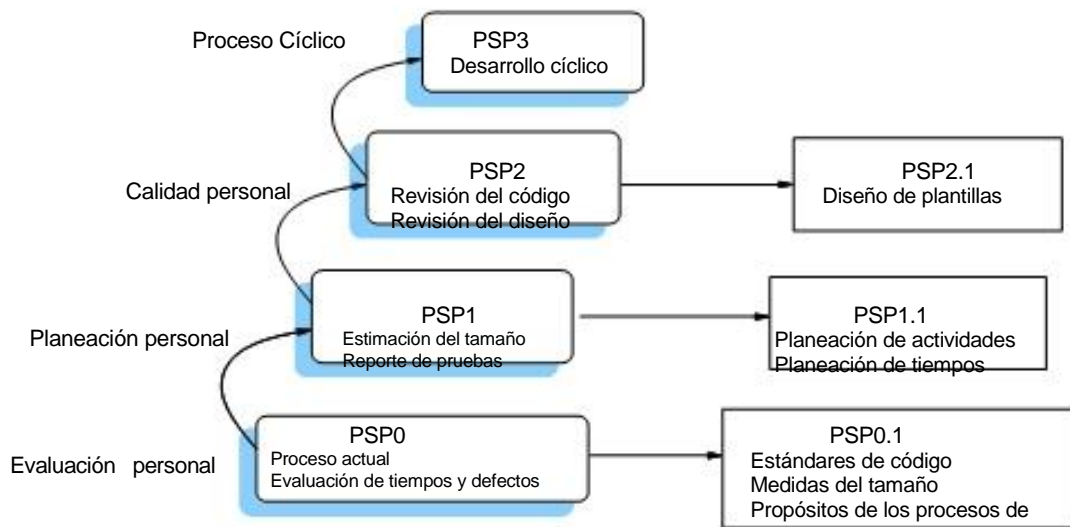


Figura 3.7: Fases de PSP

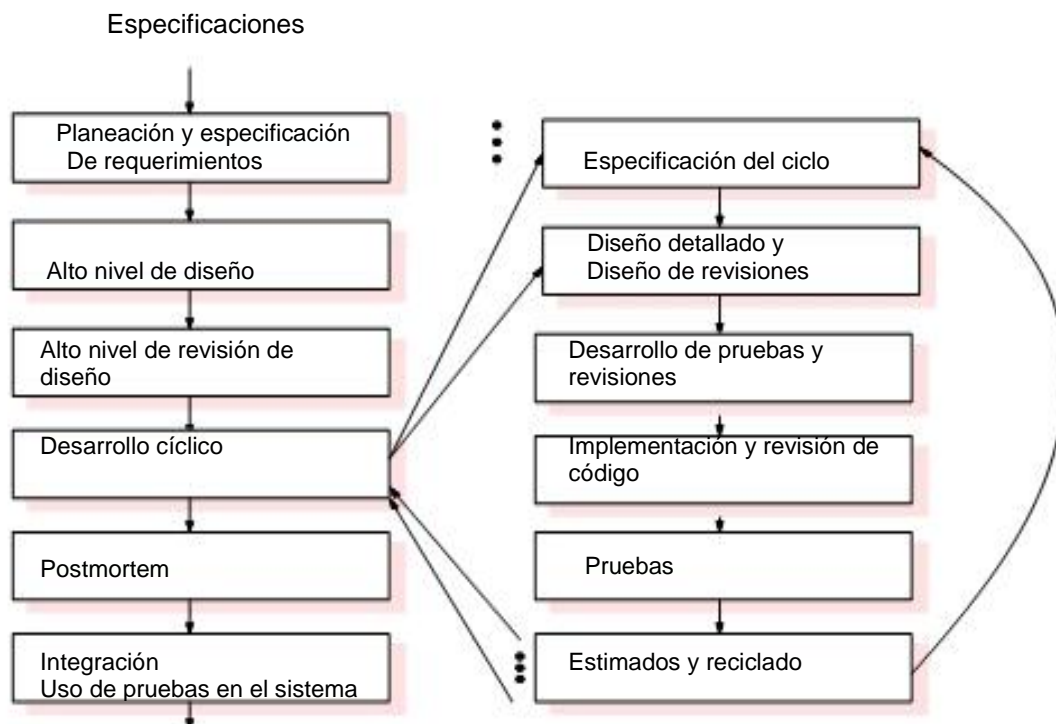


Figura 3.8: Proceso PSP3

Mejora cíclica (PSP3). La figura 3.8, muestra como cada una de las fases son base para la siguiente fase en un modo escalar hacia arriba. Se dice que PSP3 es cíclico porque cada vez que se aplica los resultados son mejores. Como se aprecia en la figura 3.9, la relación que guarda CMM y PSP es directa. CMM se enfoca a la cuestión de qué entorno a las actividades en un proceso de desarrollo de software que tiende a mejorar. Mientras que PSP se enfoca al cómo mejorar los procesos. En términos generales se mejora el desempeño de los Ingenieros que son parte del proceso de desarrollo del software. En la figura 3.6 es posible observar los elementos de PSP en CMM.



CAPÍTULO IV.

METODOLOGÍA PARA LA EVALUACIÓN DE LOS ASPECTOS DE CALIDAD EN APLICACIONES WEB

CAPÍTULO IV. METODOLOGÍA PARA LA EVALUACIÓN DE LOS ASPECTOS DE CALIDAD EN APLICACIONES WEB.

4.1. Introducción.

A pesar del gran número de artículos de investigación y normas existentes sobre el tema de validación de calidad del producto de software, existen hoy en día muy pocas empresas de desarrollo de Software que utilicen procesos de evaluación y análisis para este efecto. Actualmente, la gran mayoría de estudios están enfocados a las actividades de administración de los proyectos de desarrollo de software. En diversos entornos industriales y académicos, la calidad del software ha sido evaluada mediante distintos estudios analíticos. De dichos entornos, la evaluación de la calidad del software ha evolucionado hacia modelos formales estadísticos que se basan en métricas como fundamento para el aseguramiento, control y evaluación de la calidad de un producto o proceso de software.

Grandes compañías como IBM, Hewlett Packard, Motorola y Siemens, entre otras, fundamentan su entorno de producción de software con este enfoque estadístico, lo cual las ha convertido en pioneras de este campo.

En este trabajo se pretende desarrollar una propuesta de metodología para la evaluación del comportamiento del atributo de calidad *usabilidad* en aplicaciones Web.

4.2. Aspectos importantes de la usabilidad.

"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso"

La usabilidad es un aspecto muy importante de cualquier sistema interactivo. El éxito o fracaso de una aplicación depende en gran medida de su usabilidad, más que de lo que realmente hace. Así pues, aplicaciones que hacen cosas similares pueden tener éxito o fracasar. En la actualidad, aspectos estéticos o de imagen pueden ser muy importantes a la hora de preferir ciertos productos de consumo sobre otros.

Planificar y evaluar la usabilidad nos ayuda a evitar problemas, como las malas experiencias de los usuarios al utilizar una aplicación o herramienta.

La usabilidad describe la facilidad de uso, aprendizaje, utilidad y que tan bien los usuarios logran usar y explotar sus funciones [3]. Dichos componentes dan una pauta para medir la usabilidad.

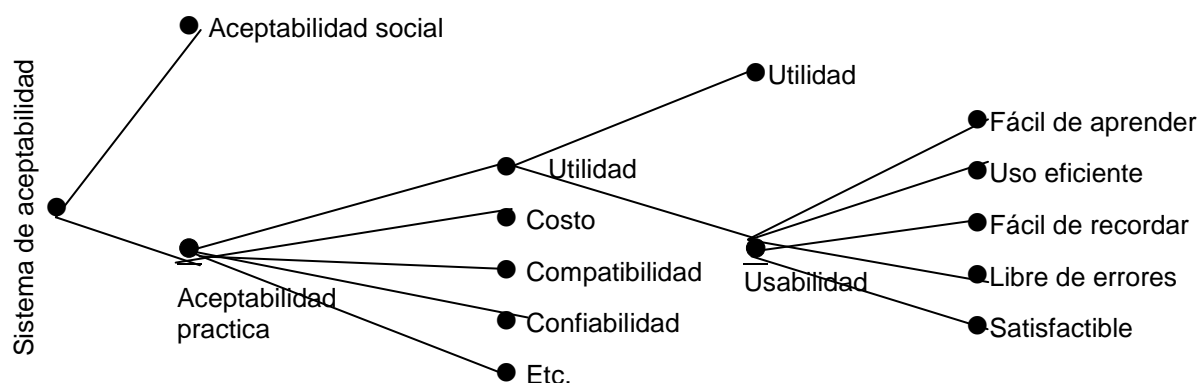


Figura 4.1. Esquema simple de los atributos del grado de aceptación de una aplicación

Según Nielsen 1993 [2], un ambiente usable necesita cumplir con los siguientes principios, mismos que se destacan en la figura 4.1:

Facilidad de aprender. Una aplicación debe ser diseñada para que el usuario aprenda fácilmente a llevar a cabo sus tareas; debe ser fácil de aprender para que el usuario pueda trabajar de manera eficiente lo más pronto posible. Es decir, es una medida de tiempo requerida para trabajar con cierto grado de eficiencia en el uso de una herramienta. Es posible medir la facilidad de aprendizaje respondiendo a preguntas como: *¿Qué tan fácil es de aprender?*, *¿Cuánto se toma el usuario para realizar trabajo productivo?*

Uso eficiente. Una vez que el usuario ha aprendido a utilizar la aplicación, debe tener un alto grado de productividad. Esta característica se mide como la velocidad con la que un usuario efectúa varias operaciones en una unidad de tiempo. Se puede medir la eficiencia o facilidad de uso respondiendo a preguntas como: *¿Resuelve las tareas eficientemente?*, *¿Se logra un alto nivel de productividad?*

Fácil de Recordar. Después de determinado tiempo de no interactuar con la aplicación, el usuario es capaz de recordar como se utilizaba sin necesidad de reaprender. Es decir, el grado de retención de conocimientos que presenta el usuario después de cierto tiempo de no usar la aplicación. Se puede medir si es recordable respondiendo a preguntas como *¿Es fácil de recordar?*, *¿Se debe aprender cada vez que se usa?*

Libre de errores. La aplicación debe tener un bajo índice de errores cuando el usuario trabaja con el. En el caso de que algún error ocurriera, debe poder recuperarse fácilmente. Se puede medir el índice de errores respondiendo a preguntas como: *¿Se obtienen resultados no deseados?*, *¿Se cometen errores al operarlo, puedo recuperarme fácilmente de ellos?*

Satisfactible. Esta es una medida subjetiva, donde se toman en cuenta las percepciones, opiniones, actitudes y sentimientos del usuario por la aplicación. Se puede medir la satisfacción respondiendo a preguntas como: *¿Causa satisfacción?*, *¿Es considerado como importante para las metas y necesidades?*

En un entorno competitivo, en el que es de vital importancia la satisfacción del usuario, conocerlo, identificar lo que requiere, atraer su atención, retener su lealtad,

hay que diseñar aplicaciones que faciliten su utilización, su satisfacción y desempeño en el uso de la misma.

En muchas ocasiones cuando tenemos problemas con la aplicación, llegamos a pensar que algo anda mal con nuestro equipo, también nos sentimos frustrados cuándo no logramos lo que deseamos.

La interacción debe ser clara, tener la información bien organizada, pues el objetivo del diseño *no es impactar* si no ayudar al usuario. Las interfaces desordenadas, confusas o complicadas en las que el usuario no consigue los resultados deseados, serán abandonadas por la mala experiencia vivida.

Por lo tanto, una mala usabilidad conlleva al menos a los siguientes costos y problemas:

- Gastos extras para soportar la aplicación
- Gastos extra para mantener la aplicación
- Largos y costosos periodos de aprendizaje
- Gastos en personal de capacitación
- Consecuencias y gastos por errores al utilizarla
- Dificultades para solucionar problemas y efectuar rediseños
- Tiempo y energía desperdiciadas
- Frustración

4.2.1. Importancia de la usabilidad

La importancia de la usabilidad según el perfil, se aprecia en los siguientes puntos:

- **Usuario:** Es la diferencia entre hacer una tarea de forma precisa y completa o no y que el proceso sea ameno o frustrante.
- **Desarrollador:** La diferencia entre el éxito o el fallo de una aplicación.
- **Gestor:** Determina la productividad de la herramienta.

Una pregunta que comúnmente se plantea es: ¿Por qué invertir en Usabilidad?, misma que respondemos con los siguientes tres argumentos:

1. La usabilidad de un producto determina la decisión de una compra más que el precio del producto.
2. Menor tiempo de aprendizaje, equivale a beneficios de mantenimiento y Entrenamiento.
La satisfacción del usuario disminuye la incomodidad y el estrés.
3. Diseños usables suelen implicar diseños más simples. Diseños más simples son más baratos de construir.
Diseños centrados en el usuario tienen éxito en las aplicaciones de mercado.
La Usabilidad, equivale a Simplicidad, Satisfacción del Usuario y a Mayores Beneficios

Así pues, podemos destacar los principales beneficios de la usabilidad como son:

- Reducción de los costos de aprendizaje.
- Disminución de los costos de asistencia y ayuda al usuario.
- Optimización de los costos de diseño, rediseño y mantenimiento de software.
- Mejora la imagen y el prestigio del software.
- Mejora la calidad de vida de los usuarios, ya que reduce su estrés. Incrementa la satisfacción y la productividad.

Todos estos beneficios implican una reducción y optimización general de los costos de producción, así como un aumento en la productividad. La usabilidad permite mayor rapidez en la realización de tareas y reduce las pérdidas de tiempo.

4.2.2. Evaluación de la usabilidad.

La evaluación de la usabilidad es uno de los aspectos que mas se demandan hoy en día. Generalmente, se desarrolla software y posteriormente, se plantean si lo que se ha hecho es fácil de usar o no. Consideramos que seria mucho mejor si se incluyera un diseñador especializado en usabilidad desde el principio, pero eso no siempre es así, debido al incremento del costo de producción y a la falta de planeación adecuada.

Podemos encontrar diversos métodos de evaluación de la usabilidad, en especial, aquellos que descubren los aspectos de la aplicación. Destacando que algunos de ellos parten de que el software ya esta hecho. Cabe señalar que los métodos que en este trabajo se mencionan para evaluar y probar no son los únicos, existen muchos más, además de múltiples variaciones de los mismos.

4.2.2.1. Evaluación heurística.

Encontramos que la evaluación heurística trata de encontrar problemas de usabilidad por medio de un grupo de métodos basados en una serie de evaluadores expertos que inspeccionan o examinan aspectos relacionados con la usabilidad de la aplicación, comprobando si esta satisface una serie de reglas de usabilidad. Básicamente se basa en la observación por parte de un experto en usabilidad, de ciertos parámetros o guías generales [2]:

1. **Dialogo natural.** Los diálogos no deben contener información irrelevante o que el usuario no necesite saber.
2. **Hablar el lenguaje de los usuarios.** Se refiere a que la interfaz esté diseñada en el idioma nato del usuario final. Se deben usar palabras, conceptos o frases que sean familiares al usuario.
3. **Minimizar el uso de la memoria.** El usuario no tiene que memorizar el procedimiento para la operación de la aplicación, ya que los procesos son fáciles y sencillos.
4. **Consistencia.** Usar las mismas palabras o acciones para situaciones equivalentes que se presentan en la aplicación.
5. **Retroalimentación.** La aplicación debe decirles a los usuarios qué es lo que está pasando.
6. **Proveer salidas.** Si el usuario realiza una acción que no es correcta, se debe tener la posibilidad de regresar al estado anterior, como deshacer una acción.

7. **Proveer mecanismos aceleradores.** Se refiere a los shortcuts, que generalmente usan los usuarios expertos para agilizar su trabajo.
8. **Ofrecer buenos mensajes de error.** Indicar el problema preciso y sugerir una solución.
9. **Prevenir errores.** Cuidar el diseño para evitar problemas futuros.
10. **Contar con ayudas y documentación.** Aunque la aplicación pueda ser usada sin la necesidad de ayudas en línea, es conveniente tener algún tipo de documentación. La información que se presenta debe estar orientada a las tareas de los usuarios y debe ser fácil de encontrar.

Es importante mencionar que los evaluadores son personas expertas en usabilidad. Esto no significa que tengan grandes conocimientos sobre el tema, sino que han comprendido los diez principios heurísticos ya mencionados. Estos expertos revisan el producto de software a evaluar teniendo en cuenta dichos principios y emiten los resultados.

Con frecuencia, los problemas encontrados por los evaluadores son de diferentes grados de importancia. Por ello, es conveniente especificar la importancia de los problemas. En principio se sugiere tres áreas generales:

- **Frecuencia con la que ocurren los problemas:** Si es común o raro que pase una cosa.
- **El impacto del problema:** Si los usuarios tendrán mucho problema cuando pase eso.
- **La persistencia del problema:** ¿Es un problema que sólo ocurrirá una vez o los usuarios estarán constantemente chocando con ese problema?

Además, se cree que es necesario evaluar el impacto del problema en el mercado, ya que algunos problemas pueden ser de poca importancia pero afecta mucho la imagen del sistema.

Al respecto, los problemas pueden representarse según la siguiente escala:

- 0 = No creo que sea un problema de usabilidad
- 1 = Problema sin importancia: No necesita arreglarse a menos que haya tiempo de sobra.
- 2 = Problema de poca importancia: Arreglarlo no tiene mucha importancia.
- 3 = Problema Grave: Es importante arreglarlo.
- 4 = Catástrofe: Es obligatorio arreglarlo.

A fin de organizar de forma adecuada la evaluación heurística sugerimos seguir los siguientes pasos:

- **Entrenamiento previo a la evaluación:** Dar a los evaluadores conocimiento del tema e información.
- **Evaluación:** Los evaluadores evalúan la aplicación y detectan errores de usabilidad.
- **Especificar la severidad:** Determinar la severidad de cada uno de los problemas encontrados.
- **Revisión:** Revisar los problemas encontrados con el equipo de diseño.

4.2.2.2. Recorrido de la usabilidad plural

Este método fue desarrollado en los laboratorios IBM. Comparte algunas características con los recorridos tradicionales pero tiene algunas características propias que lo definen. Estas características son las siguientes:

- 1) Participantes: Este método se realiza con tres tipos de participantes, usuarios representativos, desarrolladores y expertos en usabilidad, que conforman todos los actores implicados en la aplicación.
- 2) Las pruebas se realizan con prototipos de papel u otros materiales utilizados en escenarios. Cada participante dispone de una copia del escenario de la tarea con datos que se puedan manipular.
- 3) Todos los participantes han de asumir el papel de los usuarios, por tanto aparte de los usuarios representativos que ya lo son, los desarrolladores y los expertos en usabilidad también lo han de asumir.
- 4) Los participantes han de escribir en cada panel del prototipo la acción que tomarán para seguir la tarea que están realizando, escribiendo las respuestas lo mas detalladas posibles.
- 5) Una vez que todos los participantes han escrito las acciones que tomarían cuando interactuaban con cada panel, comienza el debate. En primer lugar deben hablar los usuarios representativos y una vez estos han expuesto completamente sus opiniones, hablan los desarrolladores y después los expertos en usabilidad.

4.2.2.3. Recorrido cognitivo

El recorrido cognitivo es un método de inspección de la usabilidad que se centra en evaluar en un diseño su facilidad de aprendizaje, básicamente por exploración y está motivado por la observación que muchos usuarios prefieren aprender software por exploración [4].

El recorrido cognitivo está basado en los recorridos estructurales tradicionales que se usan en la comunidad de la ingeniería de software [5]. En el recorrido cognitivo los revisores evalúan una propuesta de la aplicación en el contexto de una o más tareas de esta. En un recorrido partimos inicialmente con un diseño detallado (por ejemplo en forma de prototipo de papel o en un prototipo de software), un escenario, tareas a realizar, un estudio de la población de usuarios y el contexto de uso. Para cada tarea dispondremos mediante los documentos del análisis de tareas de una secuencia de acciones que el usuario tiene que realizar satisfactoriamente para completar la tarea designada, para cada acción el analista explicará la interacción que el usuario puede realizar típicamente con la interfaz, que va a intentar realizar y que acciones están disponibles. Si el diseño es bueno, las intenciones del usuario provocarán que se seleccione la acción apropiada, la interfaz debe presentar una realimentación indicando que se están realizando progresos para completar la tarea.

4.2.2.4. Inspección de estándares

Este método se realiza por medio de un experto en un estándar; el experto realiza una inspección minuciosa para comprobar que cumple en todo momento y globalmente todos los puntos definidos en el estándar [6].

4.2.2.5. Observación de campo

Consiste en visitar a los usuarios mientras realizan su trabajo, de manera que no interfiera en su tarea. La meta es que en la evaluación experimental se tome nota de lo que se ve y que parezca invisible, para que los usuarios trabajen de la misma forma en la que siempre lo hacen. Se requiere tres o más usuarios y su desventaja es que en la evaluación experimental no se tiene ningún control [2].

El objetivo principal consistirá en observarlos para entender cómo realizan sus tareas y qué clase de modelo mental tienen sobre ellas. También se les puede hacer preguntas para completar esta información.

4.2.2.6. Grupo de discusión dirigido (*focus group*)

El *focus group* [2] o grupo de discusión dirigido es una técnica de recolección de datos donde se reúne de 6 a 9 usuarios para discutir aspectos relacionados con el sistema. Un ingeniero de factores humanos hace las veces de moderador, que tiene que preparar la lista de aspectos a discutir y recoger la información que necesita de la discusión. Esto puede permitir capturar reacciones espontáneas del usuario e ideas que evolucionan en el proceso dinámico del grupo.

4.2.2.7. Entrevistas y cuestionarios

Las entrevistas nos permiten preguntar a los usuarios acerca de sus experiencias y preferencias respecto de una aplicación. Es un evento formal y estructurado en el que se interactúa directamente con los usuarios, a quienes se solicita que expresen sus opiniones acerca de la aplicación. Por medio de las entrevistas o los cuestionarios podemos darnos cuenta fácilmente de la satisfacción del usuario, qué características de la interfaz le agradan y cuales no [2]

Identificamos que para realizar la evaluación experimental se requieren por lo menos 30 usuarios de prueba, para poder encontrar las preferencias subjetivas del usuario. Algunos cuestionarios son los siguientes:

- **QUIS**, Questionnaire for User Satisfaction.
- **NAU**, Nielsen's Attributes of Usability.
- **NHE**, Nielsen's Heuristic Evaluation.

4.2.2.8. Grabación del uso (*logging*)

El *logging* implica disponer en la computadora de una ampliación de la aplicación que recoja automáticamente estadísticas sobre el uso detallado del sistema. Es útil porque muestra cómo los usuarios realizan su trabajo real y porque es fácil recoger automáticamente datos de una gran cantidad de usuarios que trabajan bajo diversas circunstancias. Típicamente, un registro de la interfaz contendrá estadística sobre la frecuencia con la cual cada usuario ha utilizado cada característica en el programa y la frecuencia con que los diversos eventos de interés (tales como mensajes de error) han ocurrido. La estadística que muestra la frecuencia del uso de comandos y de otras características de sistema se puede utilizar para optimizar características con

frecuencias usadas y para identificar las características que se utilizan o no se utilizan raramente. La estadística que muestra la frecuencia de las diversas situaciones de error y el uso de la ayuda en línea se puede utilizar para mejorar la utilidad de los desbloques futuros, reajustando las características que causan la mayor parte de los errores y la mayoría del acceso para la ayuda en línea.

4.2.2.9. Medida de prestaciones

Un test de medida de prestaciones comparte estas características:

- **El primer objetivo es mejorar la usabilidad de un producto.** El primer objetivo es mejorar la usabilidad de un producto que se está probando y también mejorar el proceso en que se basa el diseño y desarrollo del producto. Esta característica lo distingue de un test de funcionalidad que tiene como objetivo garantizar que el producto funcione de acuerdo con las especificaciones.
- **Los participantes representan usuarios reales.** Las personas que hacen el test del producto tienen que ser del grupo de personas que ahora o después utilizará el producto. Un test que utilice programadores cuando el producto está pensado para secretarias no es un test de usabilidad. Si los participantes son más experimentados que los usuarios actuales, se pueden omitir problemas que provocan que después no entre bien en el mercado. Si los participantes son menos experimentados que los usuarios actuales, podría ocurrir que se hicieran cambios que no representen mejoras para los usuarios reales.
- **Los participantes tienen que hacer tareas reales.** Las tareas a tener en cuenta en el test han de ser tareas que los usuarios hacen en el trabajo o en casa. Esto quiere decir que se han de conocer los perfiles de los usuarios y las tareas por las que el producto es relevante.
- Es evidente que no se podrán probar todas las tareas en un test de usabilidad; tan sólo unas pocas de las que el usuario utilizará con el producto. Por tanto, es importante que las tareas que se prueben sean relevantes para los usuarios y que puedan ocultar problemas de usabilidad.
- **Se observa y se registra lo que los participantes hacen y dicen.** Observar y grabar las actividades de los participantes en el test, distingue un test de usabilidad de un debate de grupo y encuestas.
- En un test de usabilidad tendremos un grupo de personas que trabajarán con el producto. De estas personas grabaremos sus actividades y sus opiniones. Normalmente estas actividades estarán relacionadas con la grabación de tareas y la respuesta a cuestionarios.
- Un test de usabilidad incluye los dos aspectos: El momento en que los usuarios están realizando tareas con el producto y el tiempo que invierten llenando cuestionarios del producto.
- **Se analizan los datos, se diagnostican problemas reales y se recomiendan cambios para fijar los problemas.** Recoger los datos es necesario, pero no es suficiente en un test de usabilidad. Después del test se tienen que analizar los datos y se consideran los datos cualitativos y cuantitativos de los participantes con las observaciones propias y los comentarios del usuario. Se utilizan todos estos datos para diagnosticar y documentar los problemas de usabilidad del producto y las soluciones recomendadas para estos problemas.

4.2.2.10. Manifestación de los pensamientos (Thinking Aloud)

Durante el transcurso de la prueba, donde el participante está realizando una tarea, se le solicita que exprese en voz alta sus pensamientos, sensaciones y opiniones mientras interactúa con la aplicación. Esto con el fin de que al efectuar la evaluación experimental se pueda saber que concepción tiene acerca de ella, es decir, como la visualizan. Destacamos que la ventaja principal es que es una prueba barata, ya que solo se requieren de dos a cinco usuarios. Ahora bien, su principal desventaja radica en que es difícil para los usuarios expresarse en voz alta [2].

4.2.2.11. Interacción constructiva

Este método es una derivación del “pensado en voz alta” e implica en vez de uno, dos usuarios que hagan el test de la aplicación conjuntamente, este método también se denomina aprendizaje por codescubrimiento. La principal ventaja de este método es que la situación del test es mucho más natural que el pensar en voz alta con usuarios individuales ya que las personas normalmente verbalizan cuando tratan de resolver un problema conjuntamente y además hacen muchos más comentarios.

Este método tiene la desventaja que los usuarios pueden tener diferentes estrategias de aprendizaje.

Un aspecto a tener en cuenta es que la interacción constructiva requiere el doble de usuarios que el método de “pensar en voz alta”.

4.2.2.12. Test retrospectivo

Si se ha realizado una grabación en vídeo de la sesión de test, es posible recoger más información haciendo que el usuario revise la grabación.

Los comentarios del usuario mientras está revisando el vídeo son más extensos que mientras ha estado trabajando en la tarea de test y es por lo tanto posible para el experimentador parar el vídeo y preguntar al usuario con más detalle sin tener miedo de interferir con el test que esencialmente ha sido completado.

El aspecto negativo más obvio es que se tarda como mínimo dos veces mas en realizar el test para cada usuario.

4.2.2.13. Método del conductor (coaching method)

El método del conductor es algo diferente de estos métodos de test de la usabilidad, en la que hay una interacción explícita entre el sujeto del test y el experimentador. En la mayor parte de los otros métodos, el experimentador trata de interferir lo menos posible con el que está realizando el test, en este caso es al contrario, se conduce al usuario en la dirección correcta mientras se usa el sistema.

Durante el test por conducción al usuario se le permite preguntar cualquier aspecto relacionado con la aplicación a un conductor experto que responderá lo mejor que pueda. Una variación del método implica que el conductor es un usuario experto.

Este método se centra en el usuario inexperto y su propósito es descubrir las necesidades de información de los usuarios de tal manera que se proporcione un mejor entrenamiento y documentación al mismo tiempo que un posible rediseño de la interfaz para evitar la necesidad de preguntas.

4.3. Estándares internacionales.

La ISO ha publicado diversos estándares que tratan específicamente la usabilidad y el diseño centrado en el usuario. La *European Usability Support Centres* [4], clasifica los estándares internacionales relacionados con el diseño centrado en el usuario en los dos siguientes grupos:

- *Estándares internacionales centrados en procesos*: Estos estándares especifican los requerimientos para el diseño de procedimientos y procesos. Ver tabla 4.1.
- *Estándares internacionales orientados a producto*: Estos estándares especifican los atributos requeridos para el diseño y desarrollo de interfaces de usuario. Ver tabla 4.2.

Tabla 4.1. Estándares internacionales orientados a proceso

Estándar internacional	Descripción/Partes
ISO 6385 (1981)	Principios ergonómicos en el diseño de sistemas de trabajo
ISO 13407(1999)	Procesos de diseño centrados en el hombre para sistemas interactivos.
ISO 9241	Requerimientos ergonómicos para trabajos de oficina con terminales visuales
	Parte 1: Introducción general
	Parte 2: Guía sobre requerimientos de tarea
	Parte 11: Guía sobre usabilidad
ISO 10075 (1991)	Principios ergonómicos relacionados con la carga de trabajo mental - Términos generales y definiciones
ISO/IEC 14598	Tecnología de la información – Evaluación de producto de software.
	Parte 1: Visión general

Tabla 4.2. Estándares internacionales orientados a producto

Estándar internacional	Descripción/Partes
ISO 9241	Requerimientos ergonómicos para trabajos de oficina con terminales visuales
	Parte 3: Requerimientos para la visualización en monitores (1992). Parte 4: Requerimientos para teclado (1998) Parte 5: Requerimientos de postura y “layout” para estaciones de trabajo (1998) Parte 6: Guía sobre el entorno de trabajo (1999) Parte 7: Requerimientos para el tratamiento de reflejo en monitores (1998) Parte 8: Requerimientos para el uso de colores en monitores (1997) Parte 9: Requerimientos para dispositivos de entrada sin teclado (2000) Parte 10: Principios de dialogo (1996) Parte 12: Presentación de información (1998) Parte 13: Guía de usuario (1998) Parte 14: Diálogos de menú (1997) Parte 15: Diálogos de comandos (1997) Parte 16: Diálogos de manipulación directa (1999) Parte 17: Diálogos para rellenar formularios (1998)
ISO/IEC 11581	Tecnología de la información – Interfaces y símbolos de sistemas de usuario – Símbolos y funciones de iconos.
	Parte 1: Iconos – General (2000) Parte 2: Iconos de objetos (2000) Parte 3: Iconos de punteros (2000) Parte 4: Iconos de control Parte 5: Iconos de herramientas Parte 6: Iconos de acción (1999)

4.3.1. ISO 9241**Parte 10: Principios de dialogo**

Esta parte describe principios ergonómicos que son independientes de una técnica específica de dialogo, pero que pueden ser aplicados como guías cuando se especifica, desarrolla o evalúa un sistema de dialogo.

Parte 11: Guía sobre usabilidad (1998)

Esta parte explica que la usabilidad depende del contexto de uso y que el nivel de usabilidad conseguido dependerá de las circunstancias específicas en que se usa un producto. Explica los beneficios de medir la usabilidad en términos de cómo se consiguen los objetivos.

Parte 12: Presentación de información

Esta parte aporta recomendaciones de la presentación de información visual, tales como ventanas, grupos, listas y tablas, etiquetas y campos, cursores y punteros, codificación alfanumérica, gráfica y codificación de color.

Parte 15: Diálogos de ordenes

Esta parte presenta requerimientos y recomendaciones para diálogos de ordenes para realizar tareas de oficina. Los diálogos de ordenes se definen por este estándar como secuencias de instrucciones aportadas por el usuario al sistema que, cuando se procesan, resulta en acciones del sistema asociadas.

Parte 16: Diálogos de manipulación directa

Esta parte contiene recomendaciones para el diseño de diálogos de manipulación directa. Dichos diálogos actúan directamente sobre los objetos apuntándolos, moviéndolos o cambiando sus características físicas con el uso de un dispositivo de entrada.

Parte 17: Diálogos para rellenar formularios (1998)

Esta parte contiene recomendaciones para el diseño de diálogos, y diseño de entrada y salida para diálogos de computadora en el llenado de formularios y cajas de diálogos que son usados en tareas típicas de oficina.

4.3.2. ISO/IEC 11581

Parte 1: Iconos – General (2000)

Esta parte contiene un marco general para el diseño y desarrollo de iconos y su aplicación en las pantallas. Contiene requerimientos generales y recomendaciones aplicables a todos los iconos.

Parte 2: Iconos de Objetos (2000)

Esta parte contiene requerimientos y recomendaciones para iconos que representan funciones asociadas con un objeto, y que pueden ser movidos y abiertos. Esta contiene especificaciones para la función y apariencia de 20 iconos de objetos usados comúnmente.

Parte 3: Iconos de punteros (2000)

Esta parte contiene requerimientos y recomendaciones para 8 usos comunes de iconos de punteros que representan apuntadores asociados con dispositivos físicos de entrada.

Parte 4: Iconos de control

Esta parte contiene requerimientos y recomendaciones para 14 usos comunes de iconos de control que son habilitados para el usuario al operar con ventanas, listas y otros elementos gráficos.

Parte 5: Iconos de herramienta

Esta parte contiene requerimientos y recomendaciones para 20 usos comunes de iconos para herramientas, y especifica la relación entre la herramienta y el apuntador de los iconos.

Parte 6: Iconos de Acción (1999)

Esta parte contiene requerimientos y recomendaciones para 23 usos comunes de iconos típicos usados en la barra de herramientas que representan acciones asociadas con objetos que muestra al usuario las acciones que puede realizar.

4.4. Laboratorio de usabilidad

Los laboratorios de usabilidad son espacios especialmente adaptados para la realización del test de usabilidad.

Consisten normalmente en dos salas, una de ellas es la sala de observación y otra la de test. Entre las dos habitaciones normalmente hay instalado un cristal visible de

separación que normalmente solo permite ver a los usuarios desde la sala de observación, pero no a las personas que se encuentran en la sala de observación desde la sala de test.

A veces hay una sala adicional adjunta a la sala de observación, lo que permite que haya un grupo de observadores adicional, normalmente los desarrolladores, que pueden debatir el test que se está realizando sin distraer a los observadores principales, los especialistas en usabilidad en la sala principal de observación. Ver Figura 4.2.

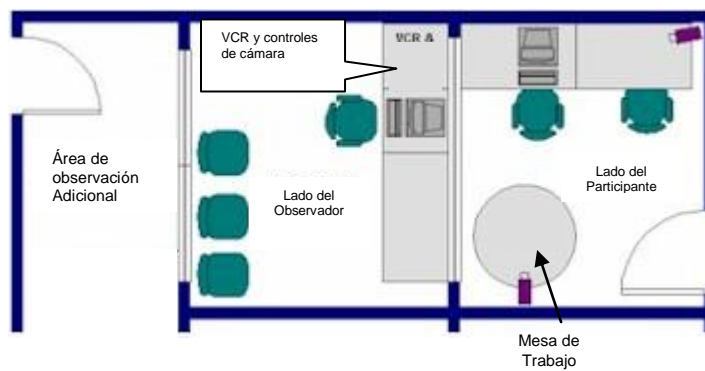


Figura 4.2. Laboratorio de usabilidad

Normalmente un laboratorio de usabilidad está equipado con varias cámaras de vídeo bajo control remoto de la sala de observación. Estas cámaras se pueden usar para mostrar una visión general del test que se está realizando y también para focalizarlo en la cara del usuario, el teclado, el manual y la documentación.

En la sala de observación se mezclan las señales de vídeo para producir una sola secuencia que se graba con el instante en que se ha generado cada secuencia para compararlo con la grabación del uso del sistema u otras informaciones.

El laboratorio de usabilidad permite poder realizar cómodamente la fase de recogida de datos de los participantes en el test de usabilidad.

4.4.1. Laboratorio de usabilidad móvil

Además de los laboratorios de usabilidad permanentes se pueden utilizar laboratorios móviles para poder hacer un test más flexible y para trabajos de campo.

Con un laboratorio de usabilidad portable (fig. 4.3 y 4.4), cualquier oficina o otras dependencias se puede convertir en una sala de test y el test se puede realizar donde están los usuarios mejor que llevar a los usuarios a un lugar fijo.

Un laboratorio de usabilidad reducido puede consistir en un bloc de notas, una computadora portátil, una grabadora de vídeo de una cierta calidad y dos micrófonos.



Figura 4.3. Laboratorio portátil de usabilidad



Figura 4.4. Laboratorio portátil de usabilidad

4.5. Establecimiento de la Metodología

La contribución del presente trabajo está dirigida al desarrollo de una metodología para el análisis y la evaluación de los atributos de calidad en las aplicaciones Web.

Se pretende lograr que esta metodología sirva de modelo para llevar a cabo la validación de los atributos de calidad en los productos de software, específicamente sobre las aplicaciones web. Se pretende desarrollar una propuesta de metodología para la evaluación del comportamiento solamente del atributo de calidad *usabilidad*, ya que evaluar todos los aspectos es un trabajo muy extenso.

4.5.1. Definir los principios de la usabilidad

Para lograr establecer el marco de referencia, el análisis de requerimientos y el diseño de de la aplicación para la evaluación de la **usabilidad** de otra aplicación, es necesario establecer algunos aspectos muy importantes dentro de la metodología, es decir, especificar los principios que van a servir de base para que posteriormente se realice la evaluación. Tiene que ver con los aspectos generales que debe cumplir una aplicación para que sea aceptada por el usuario, es decir, que opere de una manera sencilla y eficiente, para que tenga éxito. Si no se previene la usabilidad, puede traer consecuencias graves.

Estos principios son los siguientes:

- Fácil de aprender
- Uso eficiente
- Fácil de recordar
- Libre de errores
- Satisfactible

Para que una aplicación sea aceptada por los usuarios, es necesario cumplir con todos estos criterios. Un usuario que encuentre todos estos aspectos en su aplicación, es seguro que quedara plenamente convencido de que eso es lo que necesita.

4.5.2. Establecer las métricas para la evaluación

Una vez definidos los principios que debe cumplir la aplicación, se tienen que especificar los parámetros específicos para hacerle la prueba a la aplicación, se pueda evaluar y se pueda emitir un juicio.

Para determinar que tanto está cumpliendo la aplicación con la usabilidad se tiene que hacer las siguientes preguntas:

- ¿Qué tan fácil es de aprender?*
- ¿Cuánto se toma el usuario para realizar trabajo productivo?*
- ¿Resuelve las tareas eficientemente?*
- ¿Se logra un alto nivel de productividad?*
- ¿Es fácil de recordar?*
- ¿Se debe aprender cada vez que se usa?*
- ¿Se obtienen resultados no deseados?*
- ¿Se cometen errores al operarlo?*
- ¿puede el usuario recuperarse fácilmente de ellos?*
- ¿Causa satisfacción?*
- ¿Es considerado como importante para las metas y necesidades?*

Estos cuestionamientos nos van a permitir detectar los errores de usabilidad, ya que van a funcionar como métricas para medir que tanta aceptación tiene una aplicación con los usuarios.

Al hacer la prueba o test sobre la aplicación, podemos analizar con que porcentaje satisface estas preguntas y en base a eso se puede argumentar que tanto cumple con la usabilidad.

4.5.3. Definición del perfil u orientación de la prueba.

La usabilidad de un producto determina la decisión de una compra, más que el precio del producto. Esto debe analizarse desde tres enfoques diferentes:

Como desarrollador, me interesa que la aplicación que voy a producir, realmente tenga éxito o vaya al fracaso.

Como gestor, me interesa para determinar que tan productiva es la aplicación y poder venderla.

Sin embargo, para la realización de este trabajo, la prueba elaborada, se aplica al **usuario**, es decir, que está orientada hacia las personas que va a usar frecuentemente la aplicación y por lo tanto, son ellos quienes pueden determinar que tanto les satisface una aplicación dada. Esto se pretende lograr mediante la

implementación de las preguntas que estamos tomando como métricas, en la aplicación evaluadora.

4.5.4. Realización de la prueba

Una vez implementada la aplicación evaluadora, se debe aplicar la prueba a una determinada aplicación Web, con una muestra significativa, para que la información arrojada sea de mucha utilidad para emitir el dictamen final.

4.5.5. Evaluación de la prueba.

Una vez realizada la prueba, se codifica la información obtenida de tal manera que sea suficiente para poder establecer un juicio, respecto al grado que cumple con el atributo de usabilidad, la aplicación web que se evaluó.

4.5.6. Dictamen.

Una vez codificada la muestra y obtenida la información, se elabora un reporte escrito para especificar que tanto cumple con las métricas de usabilidad utilizadas en la prueba. Se describe en que aspectos de la usabilidad si cumple, en que porcentaje y en que aspectos no cumple, así como emitir recomendaciones.



CAPÍTULO V.

ANÁLISIS Y DISEÑO DE LA APLICACIÓN ECA-WEB

CAPÍTULO V. ANÁLISIS Y DISEÑO DE LA APLICACIÓN ECA-WEB.

5.1 Planteamiento del sistema.

Se requiere realizar un sistema o aplicación de software que se encargue de medir uno de los aspectos importantes de la calidad en los productos de software, la **usabilidad**. Este aspecto, como se mencionó anteriormente, tiene que ver con el éxito o fracaso de un sistema o software, porque se refiere a la facilidad de uso, utilidad, aprendizaje y el dominio que muestren los usuarios en un corto tiempo del sistema.

Los requerimientos que deben tomarse en cuenta, en la elaboración de la aplicación de software, son los siguientes:

- Que el sistema presente una interfaz de usuario amigable e intuitiva para lograr la atención y estimular el interés del usuario para responder a test
- El sistema debe recoger las respuestas del usuario para su posterior evaluación.
- Todos los accesos al sistema deberán almacenarse, identificando el perfil o clasificación del tipo de usuario, que será utilizado para obtener estadísticas.
- El sistema deberá presentar un test al usuario, que será utilizado para medir cada uno de los aspectos de la usabilidad de una aplicación web.
- El sistema tendrá una opción que será utilizada para visualizar las estadísticas de la evaluación final.

5.2 Análisis de contenido.

La aplicación web, que se va a realizar será el software que se utilizará para medir la usabilidad de un sistema Web, debe ser un software sencillo y simple para que el usuario no tenga problema de operación, debe ser intuitivo y que no cause problemas de navegabilidad.

El contenido que tendrá dicha aplicación es el siguiente:

1. Una página de presentación de la aplicación web, es decir debe mostrar los objetivos de dicha página.
2. Una página que presente el cuestionario que será resuelto por el usuario y que contendrá las preguntas relacionadas para medir la usabilidad de un sistema.
Dichas preguntas tendrán opciones de respuestas que podrá elegir el usuario.
3. Una página que visualizará el resultado hasta ese momento, de los usuarios que hayan respondido el cuestionario, mostrará graficas de barra para indicar los porcentajes y la tendencia de la prueba.
4. Finalmente una página que muestre los agradecimientos y salida del sistema.
5. los iconos de la aplicación web, deben ser representativo con alusión a la funcionalidad de dicha pagina, recordemos que es una aplicación Web para medir un atributo de calidad, como lo es la usabilidad (ECA-WEB).
6. La aplicación web, será diseñada dividiendo la pantalla de visualización en tres áreas, en la columna izquierda contendrá las opciones de

navegabilidad, en la parte superior el título descriptivo de la aplicación y en la columna derecha es la parte de visualización de acuerdo a la elección que se tiene de las opciones de navegabilidad.

7. Además se debe de contar con botones de avance, retroceso, enviar, limpiar forma, etc.
8. Por la sencillez que se requiere, se omitirá utilizar sonido de fondo, para la aplicación.

Como se puede observar, el análisis de contenido se encarga de identificar los elementos que contendrá una aplicación web, como lo es texto, imágenes, gráficas y sonido.

5.3 Análisis de interacción.

Para poder utilizar el sistema ECA-WEB, se tendrá que identificar, en una hoja de registro que proporcionará el sistema, posterior a esto, debe de llenar un cuestionario que será la parte medular de esta aplicación.

La operación del sistema y uso del test, desde la entrada hasta el final de su ejecución, se ilustra en la siguiente figura:

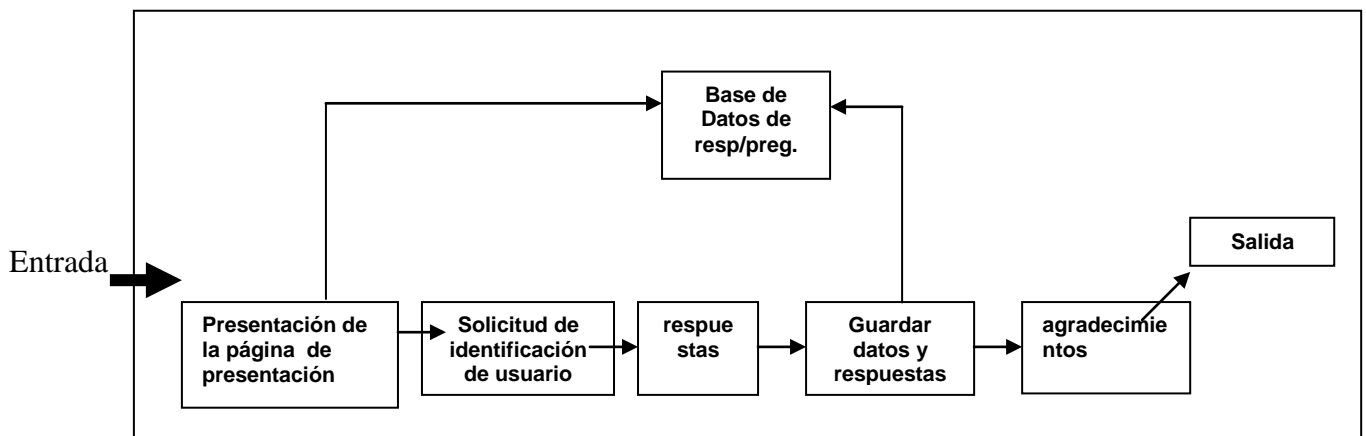


Figura 5.1 Interacción del sistema.

5.4 Análisis funcional.

Para el análisis funcional de la aplicación web (ECA-WEB), se utiliza un modelo de Caso de Uso, donde se tiene un actor que representa al usuario que va interactuar con el sistema. Además de dos casos de Uso, uno representa en si la interfaz gráfica de la aplicación e incluye un caso de uso más que representa la base de datos de las preguntas y respuestas del sistema, esta base de datos servirá para mostrar las estadísticas de la prueba.

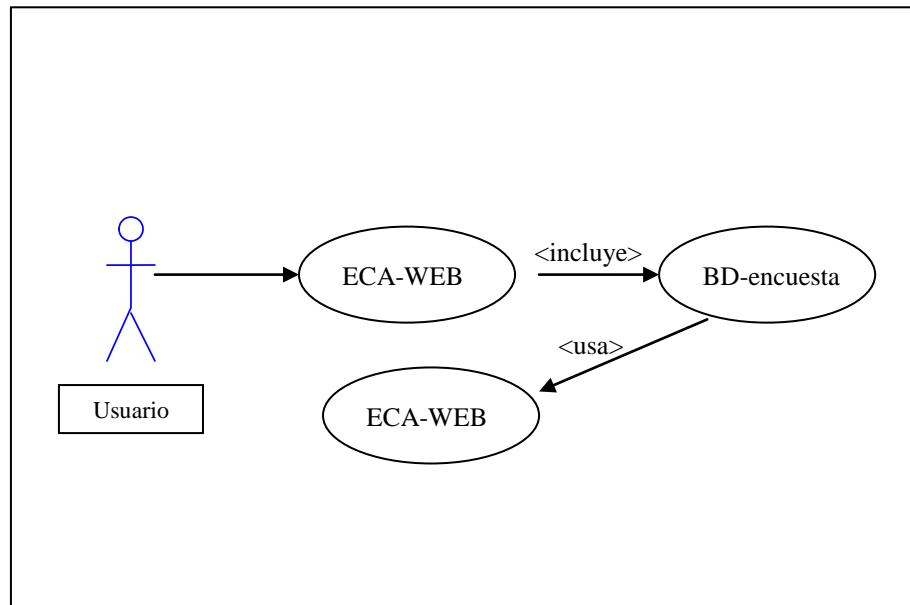


Figura 5.2 Modelo de caso de uso del sistema

5.5 Arquitectura del sistema.

La arquitectura del sistema describe los componentes de software y hardware que constituyen al sistema, su integración y funcionamiento. Puede definirse el flujo de información, a través de las líneas dirigidas, como se muestra en la figura 5.3.

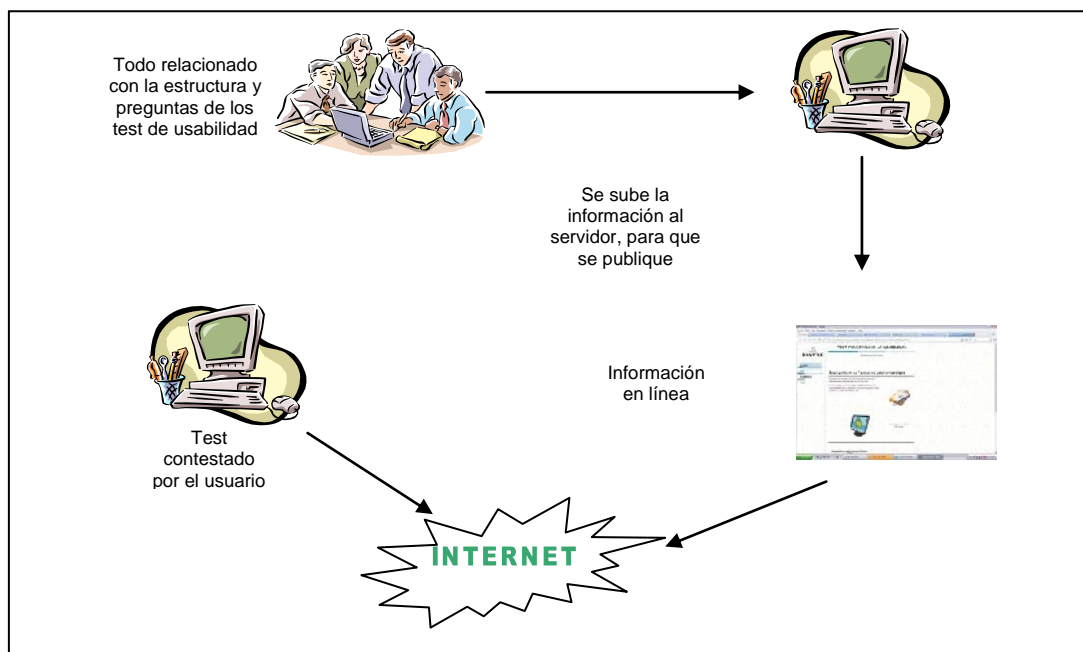


Figura 5.3 Arquitectura del sistema

5.5.1 Consideraciones de software.

En la elaboración de la aplicación que presentará el test que será contestado por los usuarios involucrados o seleccionados para llevar a cabo las pruebas de usabilidad que medirán la calidad de un producto de software, es necesario considerar, un servidor Web o la publicación de dicho test en Internet, ya sea contratando un sitio o alojándolo en un servidor de red local, el test puede ser codificado usando html y php, las preguntas y respuestas serán almacenadas en una base de datos para su posterior tratamiento (Tabla 5.1).

Tabla 5.1 Consideraciones de software

Software	Propuesta	Observación
Servidor Web	Apache, o un sitio web	Apache será utilizado para implementarse en una red local.
Lenguaje de Programación	Html, php u otro	Se recomienda php por que es software libre
Base de Datos	Access ó MySQL	MySQL es software libre

5.5.2 Consideraciones de hardware

Para alojar los datos y la aplicación web que contendrá las preguntas del test no se requiere de características especiales, es necesario contar con un equipo de cómputo con requerimientos mínimos que sirva como servidor web, o en caso contrario, se podrá contratar un sitio.

5.6 Diseño de la aplicación Web

En el diseño de la aplicación ECA-WEB, se utilizaron los elementos de diseño web que requiere la ingeniería web, es decir, se aplicaron principios y métodos de diseño, donde se establecen patrones y plantillas para describir el comportamiento y estructura del sistema. Los elementos que se consideraron son los siguientes:

5.6.1 Diseño de interfaz

Se requiere de una interfaz de usuario sencilla que reduzca la posibilidad de ocurrencia de errores en el servidor, reducir la cantidad de lectura de texto, mínimo texto pero claro para no producir confusión en la interpretación de funcionalidad del sistema, el equilibrio entre la funcionalidad y la estética y mostrar opciones de navegación obvias, como se muestra en las figuras 5.4 y 5.5.

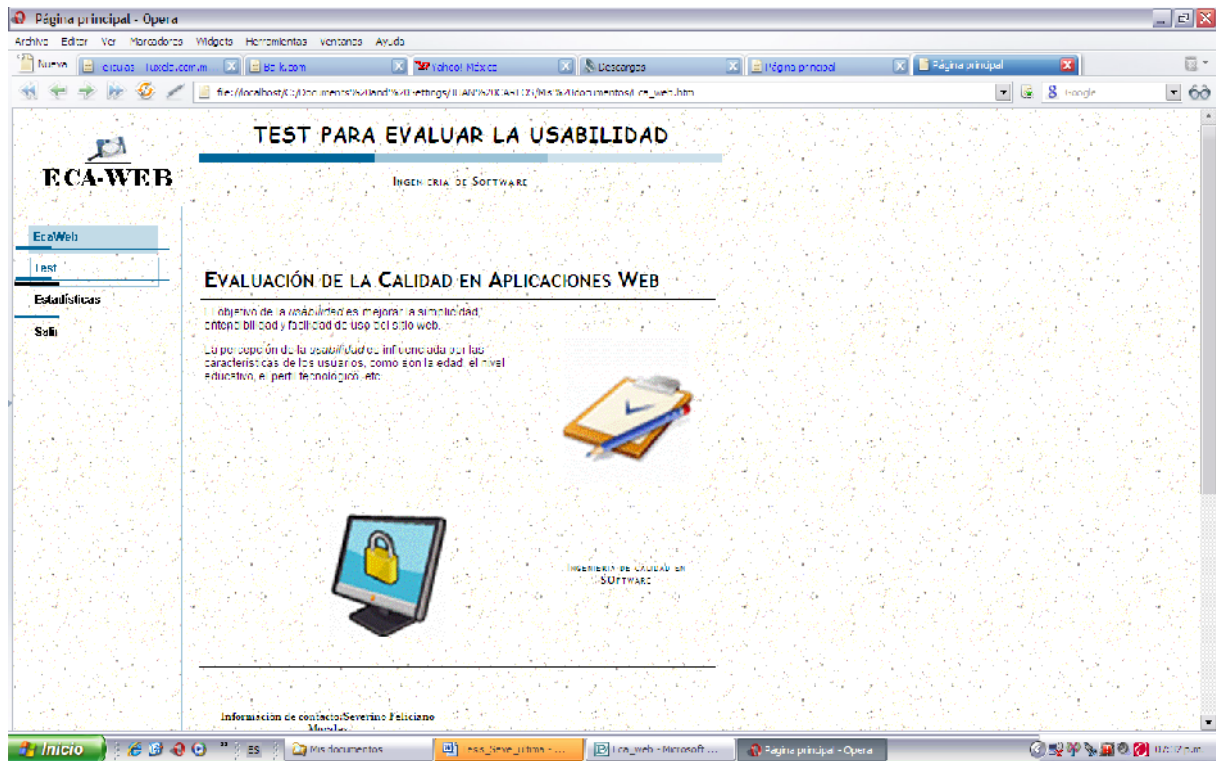


Figura 5.4 Interfaz de presentación del sistema.

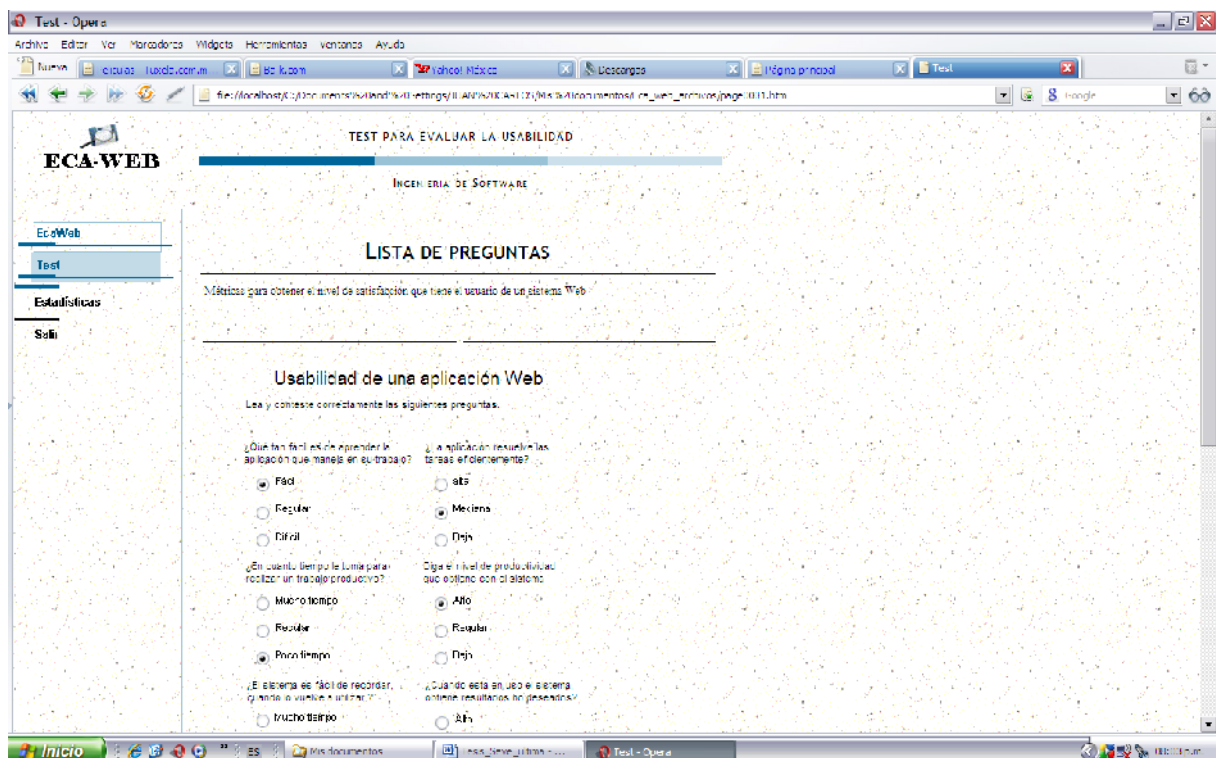


Figura 5.5 Interfaz de test del sistema.

5.6.2 Diseño estético

Para el desarrollo de la aplicación ECA-WEB, se deben considerar los elementos estéticos que demuestre seriedad en la utilización del sistema, esto es, la combinación de colores de fondo, texto y bordes, pero sin abusar de los elementos de adorno, la estructura o ubicación de los componentes, imágenes y texto descriptivo de las páginas.

5.6.3 Diseño contenido

En el apartado de análisis de contenido, se establecieron los elementos que se habrían que considerar para la construcción del sistema, en la figura 5.6 se demuestra que se atendió a este punto.

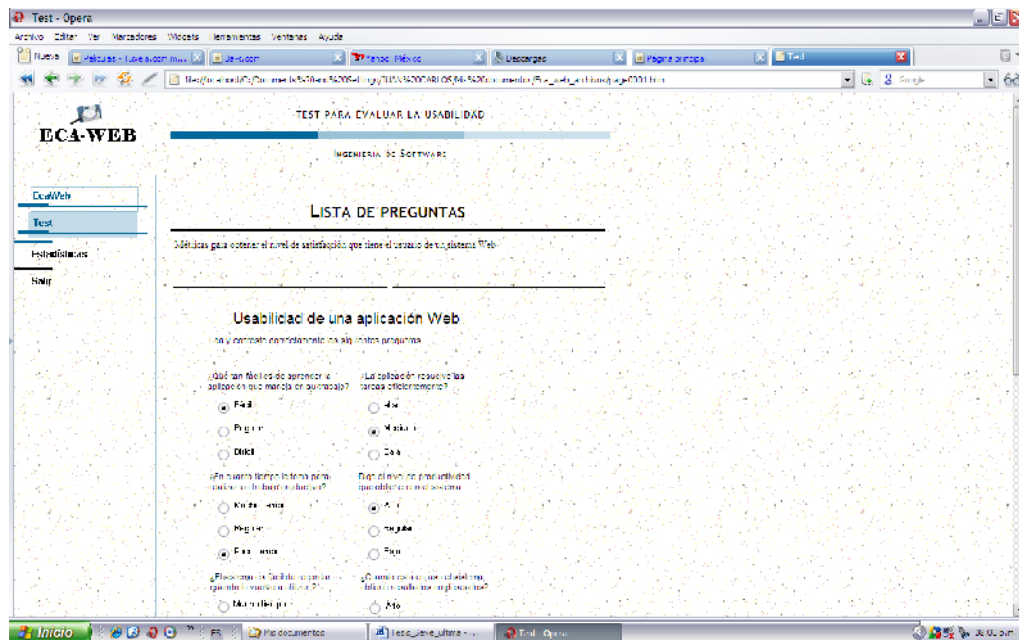


Figura 5.6 Contenido del sistema

5.6.4 Diseño de navegación

En las figuras 5.7 y 5.8, se ilustran los enlaces de navegación, se consideró que el usuario pudiera moverse de un punto actual, hacia adelante y hacia atrás, hacia la página de inicio o hasta finalizar o salir del sistema.

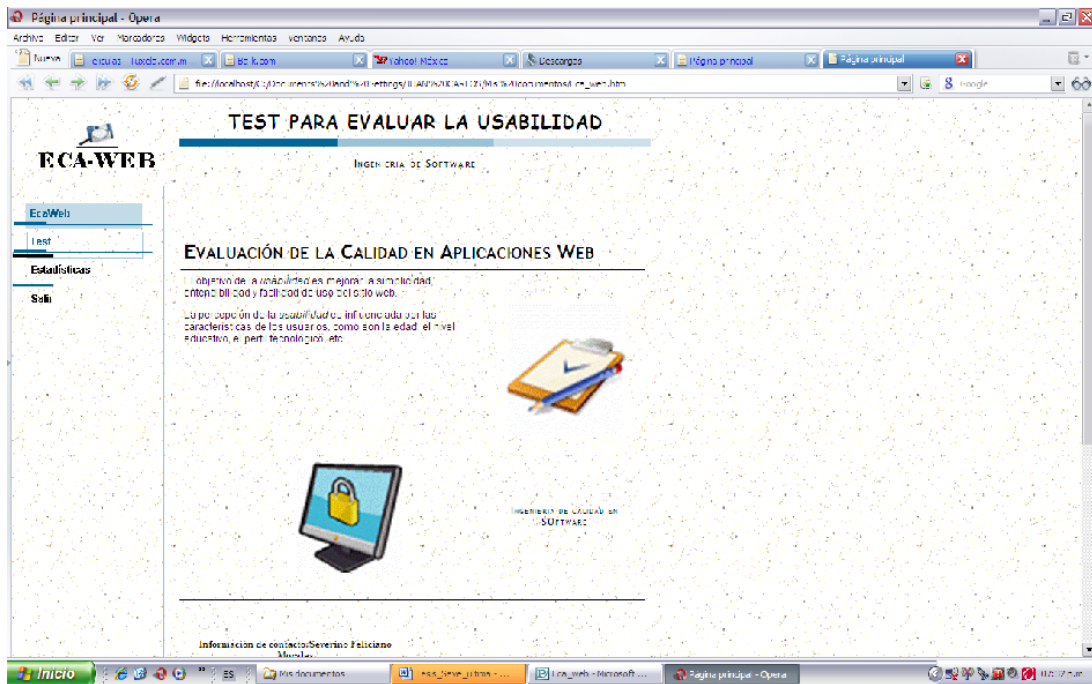


Figura 5.7 Navegación del sistema

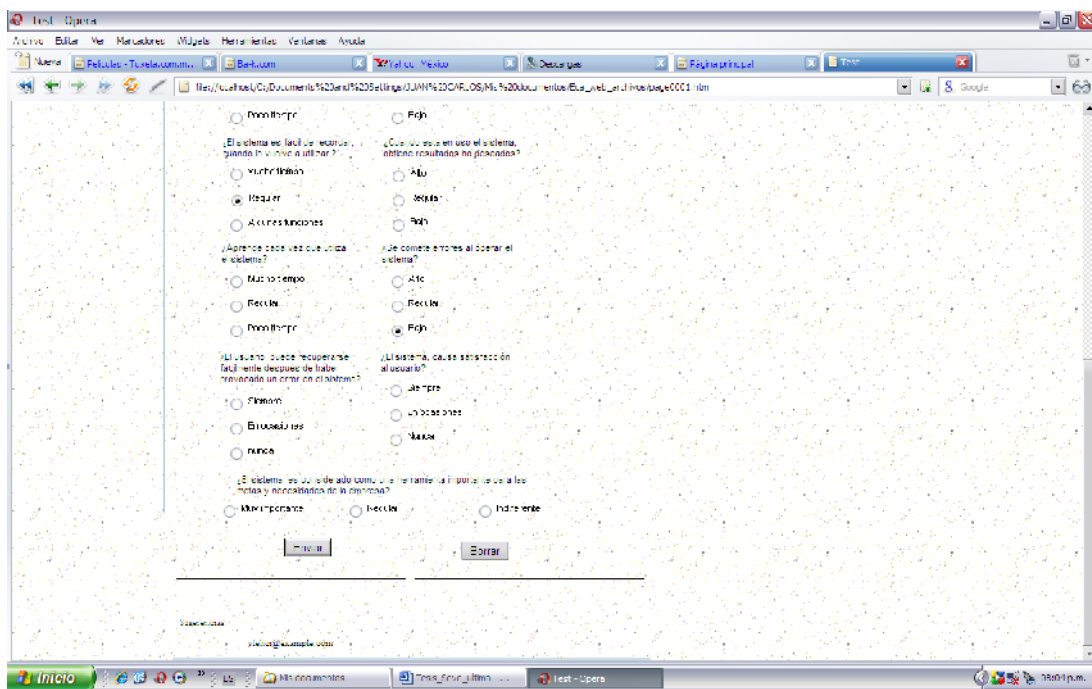


Figura 5.8 Navegación del sistema

5.6.5 Diseño arquitectónico y de componentes

En las figuras 5.9 y 5.10, se ilustra la arquitectura utilizada, así como, los componentes de la aplicación ECA-WEB, su interacción y funcionalidad.

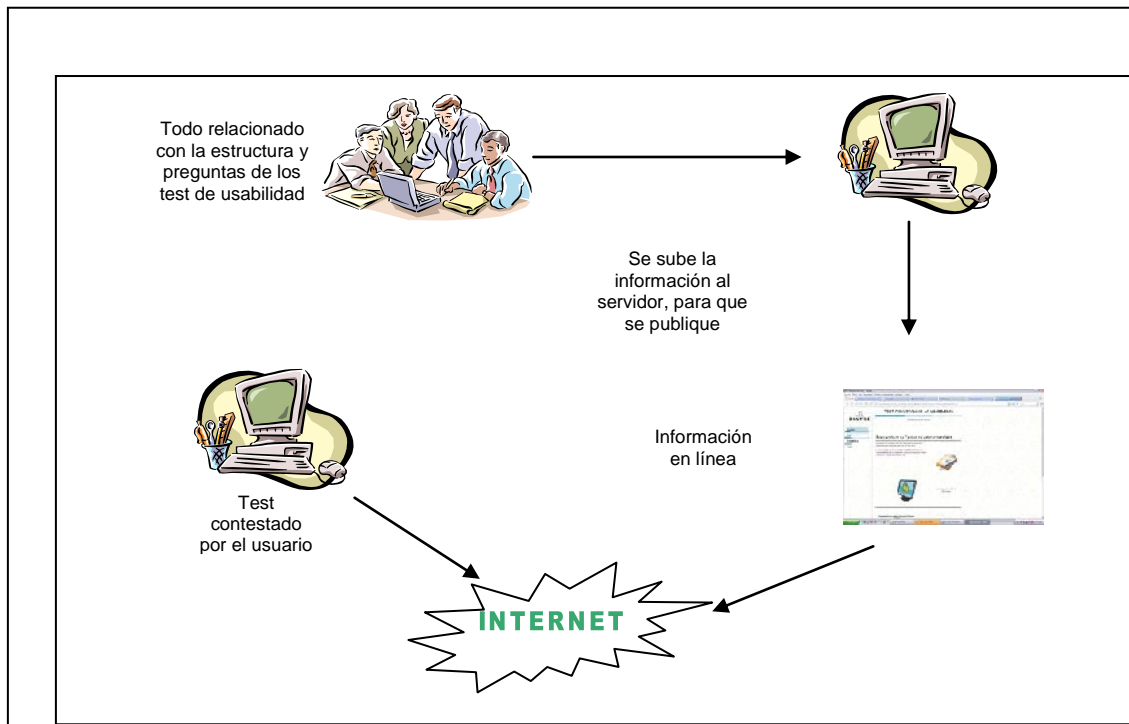


Figura 5.9 Arquitectura del sistema

Componentes de ECA-WEB

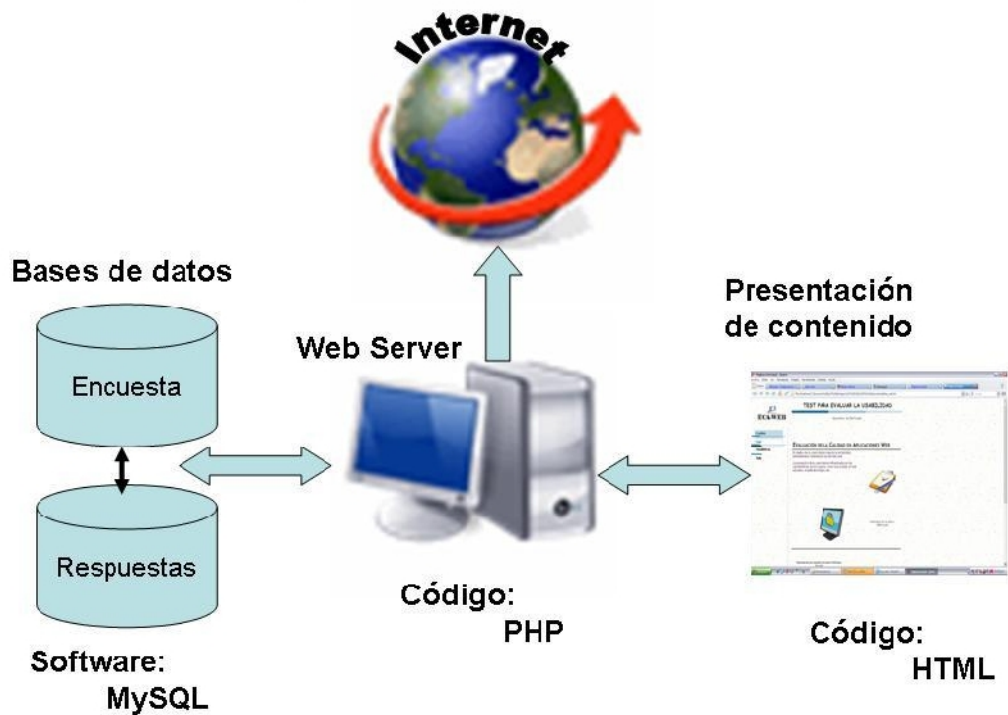


Figura 5.10 Componentes del sistema.

El desarrollo de la aplicación ECA-WEB (prototipo), se llevó utilizando los principios de la Ingeniería Web, esto garantiza que el desarrollo se realizó adecuadamente y que se espera contar con una aplicación de calidad.

RESUMEN

Al concluir este libro, se puede resumir que para que una aplicación o producto de software tenga éxito, es indispensable que se tomen en cuenta todos los atributos de calidad, ya que de no hacerlo, las consecuencias son críticas y la aplicación culminará en un fracaso.

Por eso, se hace indispensable conocer los criterios de la ingeniería de software, de la ingeniería web, los aspectos de calidad, así como algunas métricas y modelos para tener la capacidad de proponer una metodología, el análisis y el diseño de una herramienta que permita evaluar que tanto cumplen con la calidad, las aplicaciones web, que es el objetivo de este trabajo.

Es importante hacer una buena planeación sobre el desarrollo de software, donde se tomen en cuenta todos los aspectos que influyan en nuestro proyecto, pero sobre todo la aceptación de quien nos contrata para realizar el producto de software, ya que por lo general, el éxito o fracaso de una aplicación depende en gran medida de su usabilidad, más que de lo que realmente hace.

Los principales aspectos de calidad con los que debe cumplir una aplicación son: *Confiabilidad, Usabilidad, Seguridad, Disponibilidad, Escalabilidad, Mantenibilidad, Eficiencia, Flexibilidad, Portabilidad e Integridad.*

En nuestro caso solo se trabajó con el atributo de usabilidad, ya que evaluar todos los atributos es una tarea muy compleja.

"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso"

La usabilidad describe la facilidad de uso, aprendizaje, utilidad y que tan bien los usuarios logran usar y explotar sus funciones. Estos componentes dan una pauta para medir la usabilidad de una aplicación.

Una aplicación web, para que se considere que cumple con el atributo de usabilidad debe cumplir con los siguientes principios: Fácil de aprender, Uso eficiente, Fácil de recordar, Libre de errores y Satisfactible.

Por lo tanto es de vital importancia, planificar y evaluar la usabilidad para evitar problemas, tales como las malas experiencias de los usuarios al utilizar una aplicación o herramienta.

Referencias Bibliográficas:

- [1] A.C. Gillies. Software Quality, Theory and Management. Thomson Computer Press. 1999.
- [2] Nielsen J. *Usability engineering*. AP Professional, Boston, MA. 1993.
- [3] Mercovich Eduardo. Cómo hacer un Test de usabilidad de un Sitio: Panificación Selección de Usuarios, pruebas, reporte y Análisis. Buenos Aires, Argentina, 1999.
- [4] Wharton C. Et Al. «The cognitive walkthrough method: a practitioner's guide» en *Usability Inspection Methods* (Nielsen J. y Mack R. L. eds.). John Wiley & Sons, New York, NY, Pág. 1994.
- [5] Yourdon E. *Structured Walkthroughs*, 4ª edición. Yourdon Press, Englewood Cliffs, NJ, 1989.
- [6] Wixon D., Jones S., Tse L. y Casaday G. «Inspections and design reviews: framework, history, and reflection» en *Usability Inspection Methods* (Nielsen J. y Mack R. L. eds.). John Wiley & Sons, New York, 1994.
- [7] Mario Hernández Hernández. Evaluación Experimental de la Usabilidad de Interfaces Persona-Computadora. Unidad Académica de Ingeniería. Universidad Autónoma de Guerrero (Tesis de Maestría).
- [8] Leticia Dávila Nicanor. Evaluación de la Calidad en Sistemas de Información en Internet. Instituto Politécnico Nacional (Tesis de Maestría).
- [9] Carlos Muñoz Razo. Auditoría en sistemas computacionales. Pearson Educación, 2002.
- [10] Roger S. Pressmam. Ingeniería de Software. Un enfoque Práctico. Mcgraw-Hill. 5ª Edición. 2001.

Direcciones Web:

- [a] <http://www.icse-conferences.org/2002/relocation.html>
<http://www.worldcat.org/oclc/53201120>
- [b] <http://www.informatik.uni-trier.de/~ley/db/conf/ecsq/index.html>
- [c] <http://es.wikipedia.org>