



UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

IMPLEMENTACIÓN DE CALIDAD Y  
SEGURIDAD EN EL SISTEMA WEB  
"COMUNIDADES DE APRENDIZAJE  
IZTACALA"

**TESIS PROFESIONAL**

QUE PARA OBTENER EL TÍTULO DE:  
LICENCIADO EN INFORMÁTICA

PRESENTA:  
*egina*  
IRMA R. CARRIÓN BRAVO

ASESOR:  
ING. J. ANTONIO DOMÍNGUEZ HERNÁNDEZ



MÉXICO, D.F.

2005

m341419



Universidad Nacional  
Autónoma de México

Dirección General de Bibliotecas de la UNAM

**Biblioteca Central**



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

# AGRADECIMIENTOS

---

Agradezco primeramente a Dios que me ha permitido llegar a este punto, dándome lo que necesitaba en momentos que creía que ya no podía, Él me permitió lograrlo, a Él sea la gloria y la honra por lo que ha hecho en mí, por su amor llevándome cada vez más alto.

A mis padres Francisco Carrión e Irma Bravo porque con su apoyo, confianza, paciencia e impulso me hicieron más fácil el camino, gracias por que me ayudaron a seguir adelante y me dieron palabras de aliento cuando fue necesario, por darme todo lo que necesitaba sin miramientos, gracias por que ha sido una bendición el contar con su apoyo durante este trayecto y han sido de gran impulso y en verdad agradezco a Dios por sus vidas por que sé que sin ustedes no hubiera podido lograrlo.

A mis hermanos Francisco, Andrés, Daniel, Arturo y Juan C. quiero agradecerles su apoyo y paciencia para conmigo a pesar de todo, gracias por que cuando necesité ayuda me la brindaron, gracias por abrir brecha y así permitir que pudiera concluir esta etapa en mi vida, gracias por que de alguna u otra manera me han ayudado enormemente para concluir esto y hacer de mi una mejor persona. Gracias hermanos por que se que me quieren y los quiero, eso también es un impulso para ser mejor. A mis cuñadas María Elena Díaz, María Elena Cedano y Liliana Escamilla también gracias, por darme los sobrinos más preciosos que pueden existir y por su apoyo, ayuda y porras gracias.

A mis abuelitos por sus oraciones, su apoyo, su ejemplo de gente esforzada y por dar el mejor legado a mi vida: Dios. Gracias por que esa herencia es la mejor que puede haber en este mundo, por haber contribuido a mi formación desde pequeña y por su interés en mi.

A mi tía Perita Bravo porque me apoyó en gran manera cuando lo necesité, por que me aconsejó, gracias porque me diste un apoyo tremendo y me recordaste a donde debía poner mi vista en momentos difíciles.

Gracias a mi tía Chelito Bravo por estar siempre al pendiente de mis avances y por preocuparse por mí y procurarme, así mismo a mis tías y tíos Antonio Bravo, Margarita Bravo, Luz Bravo, Susana Bravo, Patricia Bravo, Elizabeth Bravo, Julio Gallegos, Hector Mercado, Hazael Rincón, Pedro Huerta, Raúl García, Salvador Assenato a mis tíos Carmen y Lalo Morales, a mi tía Matilde Guerrero, a todos ellos por su apoyo y aliento, por sus oraciones y el ánimo que me dieron, por encaminarme, gracias por su amor y por el ejemplo de seguir adelante con todo y obstáculos, por que me enseñan que la familia es una bendición.

Quiero agradecer a todos mis primos Nati, Mary, Lilia, Pris, Itzel, Sari, Rebe, Hector, Gabriel, Rafael, Jonathan, Marco, Edgar, David, Benjamín, Norma, Hazael, Aaron, Moisés, Guillermo, Ivan, Susi, Erica, Abraham, Jacobo, Leonardo, Fernando, Pablo, Alex y Mari Fer, gracias por hacer de mi familia una bella experiencia y enriquecer mi crecimiento con su compañía, gracias por ayudarme con su apoyo, ánimo y la aportación tan grande que han dado a mi vida, por procurarme y estar ahí en las buenas y no tan buenas.

A todos ustedes gracias, ya que por sus aportaciones a mi vida se concluye una etapa más en esta y podemos ver nuestras oraciones contestadas, gracias por que me alentaron a seguir adelante y por su amor. Por ello, a Dios y a ustedes MUCHAS GRACIAS.

<b>INTRODUCCIÓN</b>	1
<b>PARTE I. CALIDAD</b>	6
<b>CAPÍTULO 1. PRINCIPIOS, ESTÁNDARES Y MODELOS DEL DESARROLLO DEL SOFTWARE</b>	6
1.1. Conceptos	6
1.2. Proceso del Software	8
1.3. Modelos de Procesos del Software	9
1.3.1. Modelo de Cascada	9
1.3.2. Modelo Evolutivo	9
1.3.3. Modelo Orientado a Re-Usos	10
1.3.4. Modelo Lineal Secuencial	10
1.3.5. Modelo de Construcción de Prototipos	10
1.3.6. Modelo DRA	10
1.3.7. Modelo de Ensamblaje de Componentes	10
1.3.8. Modelo de Desarrollo Concurrente	11
1.3.9. Modelo de Métodos Formales	11
1.3.10. Modelos Híbridos	11
1.4. Ciclo de Vida de los Sistemas	12
1.5. Principios del Software	13
1.6. Principios de Calidad	13
1.7. Factores de Calidad en el Software	15
1.7.1. Factores de Calidad Internos	15
1.7.2. Factores de Calidad Externos	16
1.8. Modelos de Calidad	16
1.8.1. IEEE 1074: Estándar Para el Desarrollo de los Procesos del Ciclo de Vida	16
1.8.2. Capability Maturity Model (CMM)	17
1.8.3. ISO 9000	19
<b>CAPÍTULO 2. CARACTERÍSTICAS DE LOS PROGRAMAS EDUCATIVOS Y DE INFORMACIÓN</b>	26
2.1. Software Educativo	26
2.1.1. Características Distintivas de los Programas Educativos	26
2.1.2. Características de Calidad	27
2.1.3. Facilidad de Uso e Instalación	27
2.2. Sistemas De Información Orientados a Objetos	32
2.2.1. Conceptos del enfoque OO	33
2.2.2. Características de Los Sistemas OO	34
2.2.3. Ventajas del Enfoque OO	35
2.2.4. Gestión de Proyectos de Software Orientado a Objetos	35
2.2.5. Análisis	37
2.2.6. Diseño	44
2.2.7. Programación OO	52
2.2.8. Pruebas del Software OO	54
<b>PARTE II. SEGURIDAD</b>	60
<b>CAPÍTULO 3. IMPLICACIONES DE SEGURIDAD</b>	60
3.1. Conceptos Básicos	60
3.2. Principios de seguridad	62
3.3. Objetivos de Seguridad	63
3.4. Amenazas y Riesgos	64
3.4.1. Qué se debe de proteger	65
3.5. Políticas de Seguridad	70
3.5.1. Características de una Buena PS	71
3.5.2. Preguntas para realizar políticas	71
3.5.3. Etapas en el Diseño e Implementación de una PS	71
3.5.4. Creación del Manual de Políticas de Seguridad en Informática (PSI)	73
<b>CAPÍTULO 4. SEGURIDAD FÍSICA Y LÓGICA</b>	76
4.1. Seguridad Física	76
4.1.1. Medidas a tomar en cuanto a SF	77
4.2. Seguridad lógica	77
4.2.1. Seguridad Local	78
4.2.2. Seguridad en el sistema de archivos y archivos individuales.	78
4.2.3. Seguridad con criptografía	79
4.2.4. Seguridad del Núcleo (Núcleo).	82

4.3. Seguridad de Red	82
4.3.1. Sniffers	82
4.3.2. Servicios del sistema	82
4.3.3. Identificación de hosts vía DNS	83
4.3.4. Provisión de seguridad en correo	83
4.3.5. Escaneo de puertos	83
4.3.6. Negación de Servicios	84
4.4. Seguridad preventiva	85
4.4.1. Firewalls	85
4.4.2. Respaldos	85
4.4.3. Proteger las bitácoras	87
4.5. Herramientas de seguridad	88

### **PARTE III. ESTUDIO DE CASO: APLICACIÓN DE CALIDAD Y SEGURIDAD EN EL SISTEMA WEB "COMUNIDADES DE APRENDIZAJE IZTACALA"**

90

#### **CAPÍTULO 5. APLICACIÓN DE CALIDAD**

90

5.1. Cuestiones Generales	90
5.2. Modelo Elegido: Evolutivo	91
5.3. Ciclo de Vida de Nuestro Sistema	91
5.4. Aplicación de los Principios y Factores de Calidad en el Software. Asignación de Tareas.	93
5.4.1. Primera Revisión	94
5.4.2. Segunda Revisión	98
5.5. Modelos de Calidad	102
5.5.1. Aplicación de Calidad	102
5.6. Aplicación de las Características de Calidad de Programas Educativos y de Información	114
5.6.1. Análisis de las características externas de calidad.	114

#### **CAPÍTULO 6. APLICACIÓN DE SEGURIDAD**

116

6.1. Cuestiones Generales	116
6.2. Amenazas y Riesgos	117
6.2.1. Aplicación de OCTAVE	117
6.3. Políticas de Seguridad	118
6.4. Seguridad Física y Lógica	120
6.4.1. Seguridad Física	120
6.4.2. Seguridad Lógica	122
6.4.3. Seguridad en el sistema de archivos y archivos individuales	124
6.4.4. Seguridad con criptografía	129
6.4.5. Seguridad del Núcleo (Núcleo).	132
6.5. Seguridad de Red	134
6.5.1. Servicios del sistema	134
6.6. Seguridad preventiva	135
6.6.1. Respaldos	135
6.6.2. Proteger las bitácoras	137
6.6.3. Proteger /etc/syslog.conf	138
6.7. Herramientas de seguridad aplicadas	138

#### **CONCLUSIONES**

143

#### **BIBLIOGRAFÍA**

144

#### **ANEXOS**

146

ANEXO 1	146
ANEXO 2	153
ANEXO 3	170
ANEXO 4	183
ANEXO 5	216
ANEXO 6	220

# INTRODUCCIÓN

---

## Planteamiento del Problema

Debido a la experiencia adquirida durante el transcurso de mi preparación universitaria pude observar que al desarrollar software para los proyectos, no se consideraba como parte primordial que estas aplicaciones fuesen seguras y de calidad. Se manejaban conceptos separados de calidad y de seguridad, pero no implícitos en un sistema de forma práctica y como parte importante.

Posteriormente pude percatarme de la necesidad de tomar en cuenta estos factores; en evaluaciones según los criterios de seguridad de algunas empresas de desarrollo de software, observé que se trabaja para sacar el proyecto con las especificaciones del cliente y nada más, es decir, no existía preocupación profunda por la preservación de los sistemas a través de herramientas que así lo permitieran. Como parte de un equipo de trabajo pude darme cuenta que al desarrollar la mayor parte de aplicaciones sólo se observaba la funcionalidad de éstas, es decir, si las aplicaciones realizaban lo que tenían que hacer, o cumplían la función para la cual fueron hechas, entonces ya estaba hecho el trabajo.

Observando estas situaciones pretendí ver si esto también prevalecía en la iniciativa privada, para lo cual consulté diversas encuestas y entre éstas encontré algunas cifras que el CERT (Computer Emergency Response Team <http://www.cert.org>) dió a conocer. Dichas encuestas arrojaban un incremento significativo en el número de incidentes de seguridad que le fueron reportados durante los años 1999 y 2000, los resultados presagiaron un incremento sustancial al término del año 2001 y se previó mínimo el mismo incremento para años futuros, todo esto indica que algunas empresas no toman en cuenta la calidad y la seguridad al desarrollar sistemas.

El carecer de seguridad ocasiona, que los sistemas sean vulnerables a cualquier tipo de ataque, agresión, o accidente que prive de los servicios que se proporcionan. Así mismo, los sistemas que no aplican calidad en sus procesos pueden tener software defectuoso, no tener métodos eficientes, fáciles, rápidos y carecen de una metodología para corregir los errores o planear cómo corregir la función defectuosa entre otros problemas.

Por lo anterior, puedo concluir que el problema al que me enfrento es la falta de consideración a los requerimientos de seguridad y también al desconocimiento de los principios de calidad mínimos, así como la carencia de normas a seguir para lograr atenuar o hacer desaparecer los efectos de esta situación.

Otro problema es que la mayoría de los usuarios de cualquier tipo de equipo de cómputo no dan la debida importancia a la protección de la información, por consecuencia, generalmente no se invierte en las herramientas necesarias, ya sean humanas, tecnológicas, o económicas para prevenir daño, robo, publicación o pérdida de la información y desgraciadamente, en general no se tiene una cultura para protegerse antes de que un incidente ocurra.

## **Justificación**

La importancia de la seguridad radica en cuán relevantes son los datos, información o activos importantes contenidos en los equipos de cómputo, si éstos son de suma importancia, se necesita cierto grado de protección y confidencialidad de los datos. Si se tiene usuarios, es aún más importante cuidar los intereses de éstos, ya que la responsabilidad de los datos y el garantizar los servicios que se van brindar, radica en la persona encargada de preservar la información.

De la misma manera cada vez se va haciendo más y más importante que las aplicaciones sean hechas con calidad, no solo por la robustez que esto proporciona, sino por que también nos permite ser más competitivos; además de tener patrones de desarrollo, también permite asegurar procesos uniformes y fáciles de ajustar a las necesidades futuras.

En el LIHM (Laboratorio de Interacción Humano – Máquina) del CCADET (Centro de Ciencias Aplicadas y Desarrollo Tecnológico) de la UNAM (Universidad Nacional Autónoma de México) se tiene el proyecto de desarrollar un sistema, que permita el control, registro, actualización y consulta de información, comunicación y retroalimentación entre los usuarios del sistema y diversas comunidades docentes de la carrera de Odontología del Módulo de Instrumentación de la FES Iztacala. Los desarrolladores este sistema son algunos integrantes del Laboratorio de Interacción Humano - Máquina del CCADET de la UNAM y se encuentran en la tarea de desarrollarlo desde antes de que iniciara esta tesis.

Pudiéramos beneficiarnos del uso de herramientas que permitan elevar el nivel de calidad y seguridad, y al emplear estas medidas en el sistema, se podrá obtener un sistema competitivo al campo actual, así mismo contar con medidas eficientes de seguridad para el sistema, los usuarios de éste y las transacciones realizadas en las aplicaciones, sin descuidar el aspecto de seguridad física que esto implica, y también podremos formular un manual de políticas de seguridad.

Para ello, desde que me integré al equipo de trabajo, se está trabajando paralelamente con el equipo de análisis, diseño y desarrollo en todo el proceso hasta la implantación. Lo que se traduce en que tendremos un sistema que satisfaga las necesidades de calidad y seguridad del mismo, implantando calidad en todas las áreas aplicables del proyecto, así como en los procesos que los integrantes del equipo llevan acabo, y mostrar la integración de seguridad y calidad para tener un sistema competitivo; lo cual es fundamental para que un proyecto de este tipo pueda funcionar al cien por ciento.

## **Objetivos**

Mostrar el proceso de implantación en la estructura de un sistema de información, la seguridad y cómo es que aplica la calidad para el mismo, tanto en procesos de desarrollo del sistema, diseño de interfaz, programación, gráficos, así como en todas y cada una de sus partes.

## **Objetivos Particulares**

- ✓ Garantizar el comportamiento del sistema según lo esperado por los usuarios, administradores y todas las personas que pudiesen entrar en contacto con el sistema en cada una de las actividades que se realicen dentro de él, esto es proporcionar seguridad y calidad a los mismos.
- ✓ Desarrollar un módulo de seguridad que permita disminuir riesgos de pérdida o daño al sistema de información, asegurando que se haga un buen uso de los recursos informáticos que forman parte del sistema, para disminuir al máximo las vulnerabilidades que se tienen o pudieran surgir.
- ✓ Considerar niveles de calidad según ISO 9000 y normas aplicables, donde el sistema logre condiciones para establecer y mantener una estrategia de mejora continua, la cual consolide y asegure su proyección a 3 o más años.

## **Contexto de Desarrollo del Proyecto**

Para realizar esta investigación se cuenta con un sistema en desarrollo que tendrá funciones como el control, registro, actualización y consulta de información para las comunidades docentes de la carrera de Odontología del Módulo de Instrumentación llamado "Sistema de Comunidades de Aprendizaje Iztacala" que permitirá tener una comunicación y retroalimentación entre los usuarios del sistema y sus diversas comunidades, se prevé expansión de este proyecto a nuevas facultades o escuelas.

Los responsables del desarrollo de este sistema son integrantes del Laboratorio de Interacción Humano - Máquina del CCADET de la UNAM, donde además de otras tareas se diseña y desarrolla material multimedia con fines didácticos, como puede ser manuales, tutoriales, cursos, o documentos de interés para los profesores.

## **Metodología**

Dada la naturaleza del proyecto se necesitan métodos que sean recientes, o que actualmente sean competitivos, para lo cual se considerarán cuáles son las técnicas que se siguen en este campo.

Para calidad se considerará la norma ISO 9000 y sus cuatro herramientas básicas las cuales son: Análisis de la voz del cliente, Análisis de mercado, Benchmarking y Deficiencias y fortalezas productivas. Así mismo se considerarán los lineamientos de ingeniería de software, los diferentes Modelos de Calidad y se realizarán cuestionarios de evaluación de calidad internos y externos.



Hablando de seguridad para evaluación de riesgos se pretende usar el método llamado OCTAVE por sus siglas en inglés Operationally Critical Threat, Asset, and Vulnerability Evaluation, cuya patente está registrada ante el CERT (Software Engineering Institute de la Universidad Carnegie Mellon en Pittsburgh, EE.UU.). Además se observarán los parámetros citados en el libro Seguridad en Unix de Mediavilla Mauriz, Manuel y el documento Linux security<sup>1</sup>, se seguirán los parámetros de políticas de seguridad implementados en diversos organismos de México dedicados a esta tarea y otras bibliografías.

Se aplicarán cuestionarios de evaluación de calidad internos y externos, se entrevistará y empleará el cuestionario a los integrantes del equipo para la realización del sistema, al inicio del diseño y una segunda vez al tener la versión alfa del mismo. También se entrevistará y aplicará un segundo cuestionario, de evaluación del cliente (de seis a ocho usuarios) una vez que se tenga la versión preliminar del sistema.

Así mismo se seguirán los parámetros de políticas de seguridad implementados en diversos organismos de México dedicados a esta tarea y otras bibliografías.

Se aplicarán cuestionarios de evaluación de calidad internos y externos, se entrevistará y empleará el cuestionario a los integrantes del equipo para la realización del sistema, al inicio del diseño y una segunda vez al tener la versión alfa del mismo. También se entrevistará y aplicará un segundo cuestionario, de evaluación del cliente (de seis a ocho usuarios) una vez que se tenga la versión preliminar del sistema.

## **Contenido de la Tesis**

Esta tesis se encuentra dividida en tres partes, la primera es Calidad, la segunda es Seguridad, ya la tercera aplicación de ambas; aunque estos apartados en muchas situaciones son complementarios, tienen un grado de complejidad muy diferente, es por ello que se optó por la división de ambos rubros.

En la Parte de Calidad se tratarán temas fundamentales concernientes a la Calidad, a sus principios, a la calidad en Software, programas educativos y a la ingeniería de software, así como la aplicación práctica de estos conceptos al desarrollo de un sistema que se está llevando a cabo en el Laboratorio de Interacción Humano - Máquina del CCADET de la UNAM para los docentes de la carrera de Odontología del Módulo de Instrumentación de la FES Iztacala. Lo anterior implica desde desarrollar el sistema, los procesos que se sigan para esto, las herramientas que se utilizarán, las normas de desarrollo a seguir hasta la creación de manuales de políticas de calidad.

En el capítulo uno se tratarán temas básicos, conceptos, lineamientos que se siguen en el proceso del software y sus pasos, los modelos de procesos del software, el ciclo de vida de los sistemas, factores de calidad en el software, métricas en la calidad del software Orientado a Objetos, modelos de calidad y sus herramientas, todos estos temas son fundamentales para lograr una buena implementación de calidad desde un principio.

---

<sup>1</sup> <http://www.linuxdoc.org/>

En el capítulo dos se hablará de factores característicos y específicos a considerar en los programas educativos, como son la facilidad de uso, la versatilidad, el entorno visual, la navegación e interacción, etcétera, también se contemplan los programas de información, en sus diferentes etapas, análisis, diseño, programación, lenguajes de programación, y pruebas del software.

El contenido de la segunda Parte será la Seguridad, que comprende desde los conceptos básicos de seguridad, para lograr la comprensión del lector o interesados, detección de amenazas, desarrollo de políticas de seguridad, hasta la creación de un manual de políticas de seguridad para el laboratorio, se describe en el capítulo tres. Se tratarán los conceptos básicos para que el lector pueda comprender en su totalidad todos los aspectos a cuidar en un sistema computacional.

En el capítulo cuatro se tratarán temas concernientes a la seguridad física, lógica y de red que tratan de ficheros específicos en el servidor, el sistema de archivos, "contraseñas"<sup>2</sup>, se contemplan varios procedimientos importantes para la naturaleza del sistema, y para que éste siga funcionando adecuadamente.

También se hablará de la seguridad preventiva dado que la aplicación de estas medidas permitirá la preservación de la información, aplicaciones y protección de archivos importantes para la detección de acciones destructivas al sistema.

Finalmente en la parte tres se hará la aplicación práctica de calidad y seguridad, y dando a conocer los resultados obtenidos durante el proceso de aplicación en los capítulos cinco y seis respectivamente.

---

<sup>2</sup> **Password.** Contraseña, en Informática, una medida de seguridad utilizada para limitar el acceso a sistemas informáticos y archivos confidenciales. Una contraseña (password) es una cadena de caracteres que el usuario introduce como código de identificación. El sistema compara este código con una lista almacenada de contraseñas y usuarios autorizados. Si el código es válido, el sistema permitirá el acceso del usuario en aquellos niveles de seguridad que hayan conferidos.

# PARTE I. CALIDAD

---

---

## CAPÍTULO 1. PRINCIPIOS, ESTÁNDARES Y MODELOS DEL DESARROLLO DEL SOFTWARE

---

---

### 1.1. Conceptos

Para poder tener un concepto preciso de lo que es ingeniería del software debemos definir primeramente lo que es software como se precisa a continuación.

**Software** comprende programas computacionales y la documentación asociada. Este término se refiere a las instrucciones responsables de que el hardware (la máquina) realice su tarea. Como concepto general, el software puede dividirse en varias categorías basadas en el tipo de trabajo realizado. Las dos categorías primarias de software son los sistemas operativos (software del sistema), que controlan los trabajos de la computadora, y el software de aplicación, que dirige las distintas tareas para las que se utilizan las computadoras.

Por lo tanto, el software del sistema procesa tareas tan esenciales, aunque a menudo invisibles, como el mantenimiento de los archivos del disco y la administración de la pantalla, mientras que el software de aplicación lleva a cabo tareas de tratamiento de textos, gestión de bases de datos y similares. El software de red, permite la comunicación entre grupos de usuarios, y el software de lenguaje, utilizado para escribir programas constituyen dos categorías separadas.

Además de estas categorías basadas en tareas, varios tipos de software se describen basándose en su método de distribución. Entre éstos se encuentra el software "libre" (desarrollado por compañías y vendido principalmente por distribuidores), el "*freeware*" (software de dominio público, se ofrece sin costo alguno), el "*shareware*" (similar al "*freeware*", solo pagan los usuarios que lo utilicen profesionalmente, dependiendo de donde obtuvieron el código que explotaron a ellos se les paga) y el "*vapourware*" (software que no se presenta o aparece después de lo prometido).<sup>3</sup>

**Calidad del Software** es el conjunto de cualidades que caracterizan el software y determinan su utilidad y existencia.

Se cuenta con herramientas como la **Ingeniería de Software (IS)** para lograr alta calidad, la IS es una disciplina de la ingeniería que se encarga del estudio de principios y metodologías para el desarrollo y mantenimiento de sistemas de software que incluye la aplicación práctica del conocimiento científico en el diseño y construcción de programas computacionales.

---

<sup>3</sup>"Software." Enciclopedia® Encarta 2001. © 1993 - 2000.

Debe incluir todos los aspectos de la producción del software desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de éste, enfocándonos en su calidad para resolver todo tipo de errores.

Boehn<sup>4</sup> define la Ingeniería De Software de la siguiente manera: incluye la aplicación práctica del conocimiento científico en el desarrollo y construcción de programas computacionales y la documentación asociada requerida para computadoras, operarlos y mantenerlos. Otras definiciones la delimitan como una disciplina que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelve todo tipo de problemas. Se encarga del estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas de software.

La definición de Pressman<sup>5</sup> es la siguiente: disciplina relacionada con el desarrollo de productos de soporte lógico o software. Un producto de software es el conjunto completo de programas informáticos, procedimientos, documentación y datos especificados para su suministro a un cliente; el desarrollo se ocupa de todas las actividades técnicas y de gestiones necesarias para crear el producto, y realizar el desarrollo eficazmente: significa cumplir las necesidades del cliente ajustándose a unos límites de tiempo, costo y calidad.

El concepto de ingeniería de software surgió en las reuniones de trabajo organizadas por la Organización del Tratado del Atlántico Norte (OTAN) en 1968 y 1969 para estudiar lo que entonces se describía como "la crisis del software". Había demasiados proyectos de desarrollo de soporte lógico que experimentaban fallos, los cuales se atribuían al rápido aumento en la escala y complejidad del software en cuestión. Se reconoció que era necesario un planteamiento más sistemático en el desarrollo de software, que debía basarse en principios de ingeniería ya establecidos.<sup>6</sup>

El organismo de normalización ISO (International Standards Organization) ha definido los requisitos de un sistema de gestión de calidad de carácter general que cubre el desarrollo de cualquier producto (ISO 9001). Esta entidad ha publicado directrices específicas para aplicar esa norma al desarrollo de software (ISO 9000). Una organización que ponga en práctica un sistema de gestión de calidad según esa norma puede ser auditada y recibir una certificación formal de su proceso de desarrollo.

Algunos ejemplos de la aplicación de la informática son el almacenamiento, intercambio y presentación de información en Internet, los videojuegos, la mejora de imágenes obtenidas por telescopios y el control de marcapasos cardiacos, entre muchas más. En todos los casos, los principios de la ingeniería de software ayudan a garantizar que los sistemas resultantes sean fiables y funcionan del modo requerido. Los objetivos primordiales de la Ingeniería De Software son:

- Mejorar la calidad de productos del software.
- Garantizar la productividad y el mantenimiento de los productos de software de alta calidad en una forma eficiente.
- Facilitar el control de procesos de desarrollo de software.

---

<sup>4</sup> Boehn B. W. "Software Engineering", IEEE Transactions on Computers, C-25, num. 12.

<sup>5</sup> Pressman, Roger S. "Ingeniería del software: Un enfoque práctico". 4ª Edición. México. McGraw-Hill. 1998.

<sup>6</sup> "Ingeniería de software." Enciclopedia® Encarta 2001. © 1993-2000.

- Suministrar a los desarrolladores las bases para construir software de alta calidad en forma eficiente.
- Aumentar a productividad y trabajo de los ingenieros de software.

La ingeniería de software realiza un trabajo que se puede dividir en tres partes generales, sin importar el área donde se aplique, tamaño o complejidad del proyecto, estas partes son: definición, desarrollo y mantenimiento:

**1ª Fase: Definición.** En esta etapa los desarrolladores se encargan de identificar la información que se va a procesar, qué funciones y rendimiento se desean, cuales son los requisitos clave del sistema el comportamiento del sistema, qué interfaces se van a desarrollar, las restricciones de diseño que se tiene y qué criterios de validación se pondrán para decir que el sistema es correcto. Dentro de esta área se responde al qué y deberán llevarse a cabo tareas como la planificación del proyecto de software y el análisis de requisitos.

**2ª Fase: Desarrollo.** Este punto responde al cómo se harán las estructuras de datos, cómo se llevará el diseño a un lenguaje de programación, especificar cómo se harán las tareas, las funciones, los estándares a seguir, se diseñará el software, se generará el código y se probará el software.

**3ª Fase: Mantenimiento.** La última fase es la de mantenimiento y se encarga de la corrección de errores, mejoras, adiciones al software que ya se creó. El mantenimiento puede ser de 4 diferentes tipos:

1. Mantenimiento correctivo. Modifica el software para corregir defectos.
2. Mantenimiento adaptativo. Modifica el software para adecuarlo a los cambios de su entorno.
3. Mantenimiento perfectivo. Se hacen mejoras al software, se exceden los requisitos originales del cliente.
4. Mantenimiento preventivo. Hace cambios al software a fin de que se puedan corregir, adaptar y mejorar más fácilmente.

## 1.2. Proceso del Software

El software evoluciona a medida que se corrigen errores, se mejora el funcionamiento y se responde a las modificaciones que surgen en los requisitos. Cada nueva versión se crea a través de un proceso de desarrollo de software. Típicamente, el proceso se divide en cuatro fases principales:

- (1) El análisis y especificación de requisitos, donde se establece qué debe lograr el producto de software.
- (2) El diseño, que determina cómo cumplirá el software esos requisitos.
- (3) La puesta en práctica, que crea el producto de software que se ha diseñado (esto combina el desarrollo de nuevos componentes con la reutilización o modificación de componentes anteriores).

- (4) La prueba, que garantiza que el producto de software funciona como se pretende. Los productos intermedios, como las especificaciones de requisitos y los diseños de software, también se revisan en profundidad antes de pasar a la siguiente fase de desarrollo.

### 1.3. Modelos de Procesos del Software

Hay diferentes tipos de modelos propuestos por la Ingeniería de Software para poder producir un sistema, a fin de modelarlo, para poder ordenar, controlar y coordinar los proyectos de software, los modelos son:

#### 1.3.1. Modelo de Cascada

Cubre 5 grados:

1. Análisis y definición de requisitos
2. Diseño de sistema y software
3. Implementación y pruebas de la unidad
4. Integración y pruebas de sistema
5. Operación y mantenimiento

Este modelo sigue un ciclo a través de un paso a otro (de ahí su nombre) y se puede regresar de un paso al anterior, es decir, va del 1 al 2, del 2 al 3,... pero se puede regresar del 2 al 1, o del 3 al 2.

#### 1.3.2. Modelo Evolutivo

**Versión 1:**  
**Desarrollo Exploratorio.** El objetivo del proceso es sondear al cliente para explorar sus requisitos y entregar el sistema. El desarrollo empieza con las partes que se entienden del sistema. El sistema evoluciona conforme se van agregando los rasgos propuestos por el cliente.

**Versión 2:**  
**Prototipo Desechable.** En esta etapa se trata de entender los requisitos del cliente y desarrollar una buena definición de éstos para el sistema. El prototipo se concentra en experimentar con las partes de los requisitos del cliente que no se entendieron.

En este modelo en lugar de tener separadas las actividades de especificación, desarrollo y validación, se llevan a cabo paralelamente con la regeneración rápida de estas actividades.

1. Descripción del entorno
2. Actividades concurrentes

Especificación	⇒	Versión inicial
Desarrollo	⇒	Versión Intermedia
Validación	⇒	Versión final

### 1.3.3. Modelo Orientado a Re-Uso

Consta de las siguientes etapas:

1. Especificación de requerimientos
2. Análisis de componentes
3. Modificación de requerimientos
4. Diseño de sistema con re-uso
5. Desarrollo e integración
6. Validación de sistema

Esto se usa informalmente cuando se conoce el código del proyecto que se quiere reciclar o reutilizar que es similar a lo que se quiere diseñar, así se ahorra tiempo y esfuerzos trabajando solo en las modificaciones requeridas. Otras versiones de este modelo solo contemplan las etapas de la 2 a la 5.

### 1.3.4. Modelo Lineal Secuencial

Versión 1:

1. Ingeniería y modelo de Sistemas / Información
2. Análisis de los requisitos de software
3. Diseño
4. Generación de código
5. Pruebas
6. Mantenimiento

Versión 2:

3. Análisis
4. Diseño
5. Código
6. Prueba

En la versión 1 el primer paso se lleva a cabo conjuntamente con el análisis y diseño, pasos 2 y 3 respectivamente, una vez que se han terminado cada uno de los pasos se entrega el producto resultante al cliente.

### 1.3.5. Modelo de Construcción de Prototipos

Tiene 3 movimientos:

1. Escuchar al cliente
2. Construir y revisar
3. El cliente hace prueba

Estas etapas se llevan a cabo a través de un ciclo, esto es, cuando las 3 etapas han sido completadas, se reinicia el ciclo tomando en cuenta las correcciones detectadas.

### 1.3.6. Modelo DRA

Consta de 5 fases:

1. Modelado de gestión
2. Modelado de datos
3. Modelado de procesos
4. Generación de aplicaciones
5. Pruebas y entrega

Este modelo es muy similar al Lineal Secuencial, pero está adaptado para que su duración sea más corta, para ello se ocupa un enfoque de construcción basado en componentes. Se llevan a cabo repeticiones de este modelo hasta que se logra que el producto esté libre de errores.

### 1.3.7. Modelo de Ensamblaje de Componentes

1. Comunicación con el cliente
2. Planificación
3. Análisis de riesgos
4. Identificación de componentes candidatos
5. Buscar componentes en biblioteca
6. Extraer componentes si están disponibles
7. Construir componentes si no están disponibles
8. Poner componentes nuevos en la biblioteca
9. Construir "n" interacciones del sistema
10. Construcción y adaptación de la ingeniería
11. Evaluación del cliente

Este modelo pertenece al grupo de Modelos Evolutivos y es una versión del Modelo en espiral, adaptado a objetos, toma todas las características de este modelo.

### 1.3.8. Modelo de Desarrollo Concurrente

1. Bajo desarrollo
2. Cambios en espera
3. Bajo desarrollo2
4. Cambios en espera2
5. En línea base
6. Hecho

Todas las actividades existen concurrentemente (análisis, diseño, codificación, pruebas,...), pero residen en estados diferentes. Esto se logra a través de llevar un proceso concurrente donde se cambia de un estado a otro en base a una serie de acontecimientos planeados, es decir, cuando tenga esto, hago lo otro, se detectaron unos cambios, se deben hacer los cambios, y finalmente ya se hicieron los cambios.

### 1.3.9. Modelo de Métodos Formales

1. Definición de requerimientos
2. Especificación formal
3. Transformación formal
4. Integración y pruebas de sistema

Este modelo consiste en tener un conjunto de actividades para la especificación matemática del software, permiten especificar, desarrollar y verificar sistemas, aplicando una notación rigurosa y matemática. De esta forma se pueden eliminar problemas, así como corregirlos más fácilmente, mediante la aplicación del análisis matemático. Este modelo resulta caro y consume mucho tiempo.

### 1.3.10. Modelos Híbridos

Un modelo híbrido es aquél que combina las ventajas de un modelo con las de otro, es decir, si las necesidades del sistema que se vaya a desarrollar necesitan tomar algunas características de un modelo y otras de otro para su conformación, entonces éste será híbrido. Dos de este tipo de modelos son:

#### 1.3.10.1. Modelo Incremental

La versión 1 consta de 7 etapas:

1. Definir requisitos del entorno
2. Asignar requisitos a los incrementos
3. Diseñar arquitectura de sistema
4. Desarrollar incrementos de sistema
5. Validar incremento
6. Integrar incremento
7. Validar sistema

Otra versión de este modelo es la siguiente

1. Análisis
2. Diseño
3. Código
4. Prueba

En la segunda versión ya que se han llevado a cabo cada una de estas etapas, se lleva a cabo una interactividad con el cliente y usuarios, los cambios que se detectan a través de éstos se integran al sistema y se comienza de nuevo desde la primera etapa. A estas diversas entregas se les llama incrementos (1º Incremento, 2º Incremento, etcétera). Pertenecen al grupo de Modelos Evolutivos.



### 1.3.10.2. Modelo en Espiral

Tiene las siguientes fases:

1. Objetivos
2. Análisis y reducción de riesgos
3. Desarrollo y validación
4. Planeación

Este es un proceso evolutivo que se lleva a cabo de la forma como su nombre lo especifica, en espiral, y se sigue el proceso en sentido de las agujas del reloj hasta lograr el software deseado.
---

## 1.4. Ciclo de Vida de los Sistemas

Así como los humanos tienen un tiempo definido de vida, a lo largo de ésta se cubrirán ciertas etapas, de la misma manera, los sistemas tienen un ciclo de vida, que va desde estudiar si se puede hacer hasta los cuidados que deben darle a éste a fin de que no caiga en la obsolescencia. El ciclo consta de 8 etapas.

1. Viabilidad. Trata de obtener un estudio que garantice que el proyecto a desarrollar es realizable, que no se está intentando hacer algo imposible, o simplemente que ya hay algo que se ajuste cien por ciento a lo que se está buscando desarrollar. Consta de los siguientes aspectos:
  - Viabilidad económica
  - Viabilidad técnica
  - Viabilidad legal
  - Alternativas
2. Detección de requerimientos de información. Realizar propuestas. Es una etapa de recaudación de información que puede ser a través de cuestionarios, entrevistas, etcétera.
3. Análisis. A partir de la detección de las necesidades el usuario se realizan los diagramas de flujo de datos, diagramas de transición, diagramas de entidad relación, una vez que se cuenta con este tipo de datos se conceptualizan en el análisis del sistema.
4. Diseño. Es la estrategia de solución del sistema de datos poniéndolo en requerimientos de software.
5. Desarrollo. Es llevar lo planeado a cabo. Trasladar la teoría a la práctica.
6. Pruebas. Es ver cómo funciona cada módulo para demostrar que el sistema cumple los requerimientos del usuario.
7. Implementación y documentación. Es la puesta en marcha del sistema donde va a estar interactuando el usuario.
8. Mantenimiento. Es el cuidado del sistema en cuanto a cambios o reparaciones por fallas.

## 1.5. Principios del Software

Los principios del software son parámetros que sirven para tener indicadores de calidad del software, para ello se vale de puntos específicos, en este caso son cinco y son:

1. Modularidad. Se llama "*modularidad*" al acto de dividir el software en componentes identificables y tratables por separado, a estas partes se les llama módulos y están integrados para satisfacer los requisitos del sistema. En este punto deberá ser posible:
  - a) Tener la capacidad de descomponer el sistema total en partes funcionales.
  - b) Integrar el sistema completo a partir de los componentes existentes.
  - c) Comprender el sistema observándolo en partes
  - d) Tener el nivel de cohesión necesario para que las partes puedan interactuar libremente.
  - e) Estar preparado para la intervención de nuevos actores, eliminación de alguno de los módulos o reemplazo de cualquiera de las partes.
2. Abstracción. Es el proceso mediante el cual se identifican los aspectos importantes de un fenómeno y se ignoran los detalles. Dicho proceso debe ser lo más genérico posible, es decir, formar patrones que puedan ser repetibles durante el proceso que nos permita obtener el modelo.
3. Rigor. Definir y aplicar normas con precisión y exactitud durante la construcción del sistema para obtener calidad y eficiencia.
4. Formalidad. Plasmar dentro de un marco de modelos metodologías y procesos, las soluciones para el desarrollo.
5. Anticipación al cambio. Esto implica un esfuerzo especial para tratar de predecir cómo y cuánto ocurrirán los cambios y a partir de eso adaptar nuestro sistema a los cambios locales, nacionales y mundiales.

## 1.6. Principios de Calidad

Hay muchos factores de calidad, pero uno de los más destacados es el FURPS desarrollado por Hewlett-Packard. Su nombre es una acrónimo de sus siglas en el idioma inglés: Functionality, Usability, Reliability, Performance, Supportability.

1. Funcionalidad. La palabra hace referencia a si mi aplicación realiza las tareas para las cuales fue desarrollada, es decir, que responde a una tarea determinada, si realiza las funciones que debe llevar a cabo. Se evalúa el conjunto de características y capacidades del programa, la generalidad de las funciones entregadas y la seguridad del sistema global. Se cuenta con tres puntos:
  - a. Capacidad de recuperación y búsqueda.

- b. Servicios de búsqueda y navegación.
  - c. Servicios relacionados con el servicio y dominio de la aplicación.
2. Usabilidad. Este término usado en la ingeniería de software que se refiere a los atributos que permitan que una interfaz, que una aplicación pueda ser usada con facilidad. ¿Qué hago? Este punto se valora evaluando factores humanos, la estética, consistencia y documentación en general. Se toman en cuenta los siguientes factores.
- a. Capacidad de comprensión del sitio global.
  - b. Servicios de ayuda y retroalimentación en línea.
  - c. Capacidades estéticas y de interfaz<sup>7</sup>.
  - d. Servicios especiales.
3. Fiabilidad. Es muy simple entender esta palabra, ya que se refiere a la probabilidad del buen funcionamiento del sistema. Eliminar errores. Se evalúa midiendo la frecuencia y gravedad de los fallos, la exactitud de los resultados, el tiempo medio entre fallos, la capacidad de recuperación de un fallo y la capacidad de predicción del programa. Se resume en tres aspectos:
- a. Proceso correcto de enlace.
  - b. Recuperación de errores.
  - c. Validación y recuperación de la entrada del usuario.
4. Rendimiento. Rapidez, eficiencia óptima, que el producto o servicio que da el sistema sea el mejor. Es medida por la velocidad e procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia.
- a. Rendimiento del tiempo de respuesta.
  - b. Velocidad de generación de páginas.
  - c. Velocidad de generación de gráficos.
5. Capacidad de mantenimiento. Esto es que el sistema no presente problemas al tratar de hacerle mejoras o adiciones, que sea fácil llevar un mantenimiento y que de ser posible el sistema proporcione la facilidad de hacerlo; se traduce en la capacidad de tener o contar con los puntos que a continuación se mencionan, aunque principalmente se manejan los tres primeros.

**Adaptabilidad.** Es el atributo de cuán cercanamente el sistema se ajusta a un solo modelo de trabajo y es probable que con el tiempo cambie el entorno original para el que se desarrolló el software. Es cómo se acomoda a los cambios de su entorno externo.

**Extensibilidad.** Este punto quiere decir qué tan fácil será añadir o modificar nuevos componentes al sistema, si esto representará un problema o no.

**Servicios.** Esto se refiere a cuales son las opciones que permite, qué prestaciones proporciona el sistema, con qué se cuenta.

Primarios

<sup>7</sup> Interfaz es el punto en el que se establece una conexión entre dos elementos, que les permite trabajar juntos. Las interfaces de usuario cuentan con el diseño gráfico, los comandos, mensajes y otros elementos que permiten a un usuario comunicarse con un programa.

- Capacidad de hacer pruebas.** El esfuerzo necesario para probar un programa para asegurarse de que realiza su función pretendida.
- Compatibilidad.** Que puede existir con otros sistemas, o entenderse con ellos, coexistir sin causar efectos o daños en otros.
- Capacidad de configuración.** Que la configuración del sistema sea lo suficiente buena para que al hacer modificaciones en ella no dañe sus funciones.

Alternos

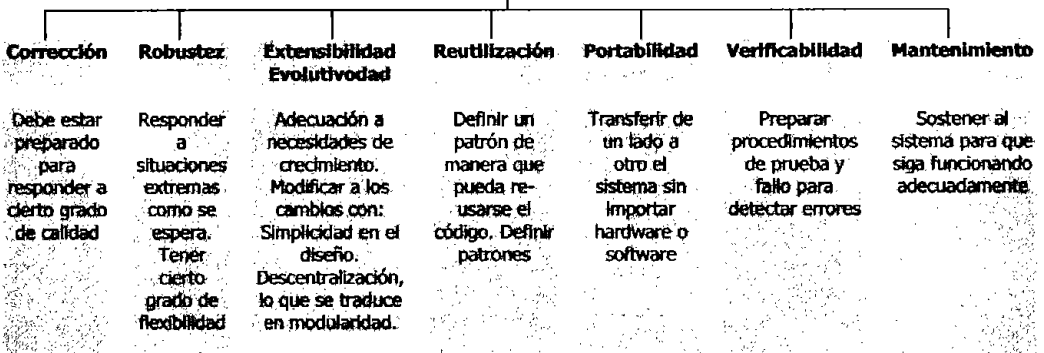
## 1.7. Factores de Calidad en el Software

Se desea que los sistemas de software sean rápidos, fiables, fáciles de usar, entendibles, modulares, estructurados y una infinidad de cosas más, que sean cada vez mejores; todos estos adjetivos describen dos tipos de cualidades diferentes llamados factores de calidad internos y externos que se describen a continuación.

### 1.7.1. Factores de Calidad Internos

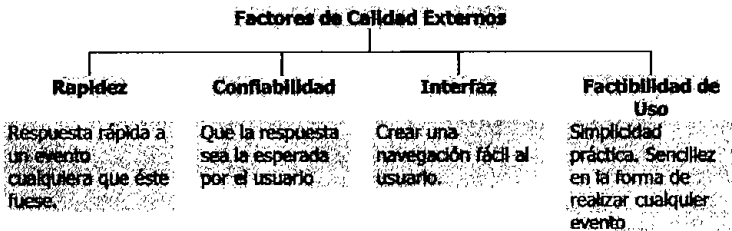
Son las características de las cuales solo un experto en el tema puede notar, ya que conoce a fondo estos conceptos.

#### Factores de Calidad Internos



## 1.7.2. Factores de Calidad Externos

Estos factores son los que detecta el usuario directamente, ya que puede percibirlos durante su interacción con el sistema.



## 1.8. Modelos de Calidad

En la actualidad son tres los principales modelos de calidad, dada su difusión, renombre y popularidad mundial son los que se ha posicionado entre los mejores y han permanecido, por estas razones haré mención de ellos. Éstos son: IEEE 1074: Estándar para el Desarrollo de los Procesos del Ciclo de Vida, CMM (Capability Maturity Model) e ISO9000.

### 1.8.1. IEEE 1074: Estándar Para el Desarrollo de los Procesos del Ciclo de Vida.

El IEEE Estándar para el desarrollo de los procesos del ciclo de vida describe un conjunto de actividades y procesos que rigen en el desarrollo y mantenimiento del software.

<b>Procesos del Software según IEEE 1074</b>	
<b>Grupo de Procesos</b>	<b>Procesos</b>
Modelado del Ciclo de Vida	Selección de un modelo del Ciclo de Vida
Administración del Proyecto	Iniciación del Proyecto Monitoreo y Control del Proyecto Administración de la Calidad del Software
Pre-desarrollo	Exploración del Concepto Asignación del Sistema
Desarrollo	Requerimientos Diseño Implementación
Post-desarrollo	Instalación Operación y Apoyo Mantenimiento Retiro
Procesos Integrales	Verificación y Validación Manejo de la configuración del software Documentación Entrenamiento

## 1.8.2. Capability Maturity Model (CMM)

Una de las metas del CMM (Modelo de Capacidad de Madurez) es proveer las guías para seleccionar las actividades del ciclo de vida del software. El CMM asume que el desarrollo de los sistemas de software se hace mucho más predecible cuando una organización utiliza un proceso de ciclo de vida bien definido, visible para todos los participantes y adaptable al cambio. El CMM utiliza los siguientes cinco niveles para caracterizar la madurez de una organización.

**Nivel 1: Inicial.** El éxito de un proyecto en este nivel de madurez usualmente depende del esfuerzo heroico y habilidades de individuos clave. Desde el punto de vista de los clientes, el modelo del ciclo de vida del software, si es que existe, es una caja negra: después de explicar el problema y hacer el acuerdo de negociación, el cliente deberá esperar hasta el final del proyecto para inspeccionar el producto.

**Nivel 2: Repetible.** Cada proyecto tiene un bien definido modelo de ciclo de vida del software. Los modelos, sin embargo, difieren entre proyectos, reduciendo la oportunidad del equipo y reutilizando el "ya sé cómo". Los nuevos proyectos se basan en la experiencia de la organización con antiguos proyectos similares y el éxito de éstos es predecible de acuerdo a aplicaciones en dominios similares. El cliente interactúa con la organización solamente en puntos muy específicos, permitiendo algunas correcciones antes de la entrega del producto final. Tiene los siguientes componentes:

- a) Enfoque: establecer controles básicos de la administración del proyecto.
- b) Administración de requerimientos: Los requerimientos se basan en el acuerdo de proyecto y se mantienen durante éste.
- c) Planeación y seguimiento del proyecto: Se establece un plan de administración del proyecto que es seguido durante la ejecución del mismo.
- d) Administración de subcontratistas: La organización selecciona y administra efectivamente calificados subcontratistas de software.
- e) Aseguramiento de la calidad: Todos los productos y actividades son revisados y analizados para verificar que cumplan con los estándares y patrones adoptados por la organización
- f) Gestión de la Configuración: Un conjunto de objetos de gestión de la configuración son definidos y mantenidos a lo largo del proyecto.

**Nivel 3: Definido.** Este nivel utiliza un modelo de ciclo de vida del software bien documentado para todas las actividades tanto administrativas como técnicas en la organización. El cliente tiene conocimiento del modelo estándar y el modelo seleccionado para un proyecto en específico. Cuenta con las etapas:

- a) **Enfoque:** Establecer una infraestructura que permita un modelo de ciclo de vida de software efectivo a lo largo del proyecto.
- b) **Enfoque de organización del proceso:** La organización posee un equipo para el mantenimiento y mejora constantes del ciclo de vida del software.
- c) **Definición de la organización del proceso:** Se utiliza un modelo estándar del ciclo de vida del software para todos los proyectos de la organización. Se hace uso de una base de datos para controlar toda la documentación e información relacionada con el ciclo de vida del software.

- d) **Programa de entrenamiento:** La organización define las necesidades para el entrenamiento para proyectos específicos y desarrolla programas de entrenamiento.
- e) **Administración de software integrado:** Cada proyecto tiene la posibilidad de adaptar sus procesos específicos a partir del proceso estándar.
- f) **Ingeniería del producto de software:** El software se construye de acuerdo con los métodos y herramientas definidos en el ciclo de vida del software.
- g) **Coordinación intergrupala:** Los equipos del proyecto interactúan con otros equipos para dirigir requerimientos y otros tópicos.
- h) **Revisiones por parejas:** Los desarrolladores examinan los productos por parejas con el objeto de identificar defectos potenciales y áreas que puedan requerir cambios.

**Nivel 4: Administrado.** Este nivel define métricas tanto para las actividades como para los productos. Hay una constante recopilación de información durante la duración del proyecto. Como resultado, el modelo de ciclo de vida del software puede ser cuantitativamente entendido y analizado. El cliente es informado acerca de los riesgos del proyecto antes de que éste inicie y conoce las medidas utilizadas por la organización. Sus fases son:

- a) **Enfoque:** Entendimiento cualitativo del proceso de ciclo de vida del software y los productos.
- b) **Administración cualitativa del proceso:** Se definen y miden constantemente las métricas de calidad y productividad a lo largo del proyecto. Esta información no es utilizada inmediatamente durante el proyecto, particularmente para mejorar la productividad de los desarrolladores, sino almacenada en una base de datos para permitir la comparación con otros proyectos.
- c) **Administración de la calidad:** La organización define un conjunto de metas para la calidad de los productos de software. Monitorear y ajustar dichas metas y productos a fin de entregar productos de alta calidad para el usuario.

**Nivel 5: Optimizado.** La información recolectada durante la duración del proyecto es utilizada como mecanismo de retroalimentación para mejorar el modelo del ciclo de vida del software a través del tiempo de vida de la organización. El cliente, administradores del proyecto y programadores se comunican y trabajan juntos a lo largo de la fase de desarrollo.

Para que sea posible medir la madurez de una organización, el SEI ha definido un conjunto de áreas claves de proceso. Para conseguir un nivel específico de madurez, la organización debe demostrar que cumple con todas las áreas claves del proceso definidas para ese nivel. Las áreas clave con las que cuenta son:

- a) **Enfoque:** Mantener rastro de los cambios tecnológicos y de proceso que puedan originar a su vez cambios en el modelo del sistema o los productos aún durante la realización del proyecto.
- b) **Prevención de defectos:** Se analizan fallas de proyectos anteriores, utilizando información de la base de datos. Si es necesario, se toman acciones específicas para prevenir que esos defectos ocurran de nuevo.
- c) **Administración del cambio de tecnología:** Las innovaciones tecnológicas son constantemente investigadas y compartidas en la organización.

d) **Administración del cambio en el proceso:** El proceso de software es constantemente redefinido y cambiado para tratar con ineficiencias identificadas por las métricas del proceso de software. Cambio constante es la norma, no la excepción.

### 1.8.3. ISO 9000

ISO 9000 es un conjunto escrito de estándares para la calidad, creados por la Organización Internacional Para La Estandarización (International Standards Organization). Una organización que ponga en práctica un sistema de gestión de calidad según esa norma puede ser auditada y recibir una certificación formal de su proceso de desarrollo, pero la certificación no garantiza el nivel de calidad de un producto, sin embargo comprueba que una compañía ha documentado prácticas y procedimientos de calidad.

Así mismo el registro con ISO 9000 no indica que el software esté libre de defectos, pero sí que el productor se compromete a corregir los errores o proveer la función defectuosa. Algunas características de calidad según ISO9126 se presentan a continuación:

#### ISO9126 Quality Characteristics

Functionality	Reliability	Usability	Efficiency	Maintainability	Portability
<ul style="list-style-type: none"> <li>• Suitability</li> <li>• Accuracy</li> <li>• Interoperability</li> <li>• Security</li> <li>• Functionality</li> <li>• Compliance</li> </ul>	<ul style="list-style-type: none"> <li>• Reliability</li> <li>• Maturity</li> <li>• Fault tolerance</li> <li>• Recoverability</li> <li>• Reliability compliance</li> </ul>	<ul style="list-style-type: none"> <li>• Usability</li> <li>• Understandability</li> <li>• Learnability</li> <li>• Operability</li> <li>• Attractiveness</li> <li>• Usability compliance</li> </ul>	<ul style="list-style-type: none"> <li>• Time behavior</li> <li>• Resource utilization</li> <li>• Efficiency compliance</li> </ul>	<ul style="list-style-type: none"> <li>• Analyzability</li> <li>• Changeability</li> <li>• Stability</li> <li>• Testability</li> <li>• Maintainability compliance</li> </ul>	<ul style="list-style-type: none"> <li>• Adaptability</li> <li>• Installability</li> <li>• Co-existence</li> <li>• Replaceability</li> <li>• Portability compliance</li> </ul>

Antes de entrar en materia es importante destacar que ISO contempla los Factores de Calidad Internos y Externos que cité anteriormente, por lo que resulta importante tenerlos presentes durante este proceso.

La manera en que podemos conseguir tener presentes los factores de calidad es a través de ISO 9000, ya que como se observa, éste los contempla. Esta norma nos señala los 4 pasos a seguir para lograrlo. Son herramientas que nos permiten analizar las diferentes áreas que intervienen dentro de la calidad. Estas herramientas son:

1. **Análisis de la voz del cliente.** Proceso de medición para conocer cómo exceder la satisfacción del cliente.
2. **Benchmarking.** Es la identificación, análisis corporativo e implantación de las mejores prácticas organizacionales mundiales para elevar la producción de áreas claves de la empresa excediendo las expectativas en calidad del cliente.
3. **Deficiencias y fortalezas productivas.** Esta tarea se lleva a cabo con la técnica de "lluvia de ideas" y permite detectar oportunidades de mejora continua.
4. **Análisis de mercado.** Es el análisis de diversos aspectos de las ventas de la empresa, de 5 años a la fecha y prospección a 3 años futuros.



### **1.8.3.1. Análisis de la Voz del Cliente**

Un cliente es alguien que tiene una expectativa hacia los productos, los servicios o la atención que se le proporciona, en este caso con esta palabra se hará referencia a los usuarios del sistema de este punto en adelante, esto debe tomarse muy en cuenta, ya que la notación de este estándar se usa la palabra cliente, pero la clientela del sistema son usuarios, y esto no debe perderse de vista.

La tarea a lograr será hacer que los usuarios tengan satisfechas y superadas sus expectativas, ya que la "Satisfacción al Cliente" es lo que gratifica y excede los requerimientos presentes y futuros del usuario actual y potencial sobre los productos, los servicios y la atención que la entidad le proporciona, sin importar la naturaleza de ésta. La "*Satisfacción al Cliente*" es un indicador que nos da este tipo de análisis y que nos va a decir si nuestro usuario está satisfecho con las opciones que nuestro sistema le proporciona.

El análisis de la voz del cliente es un proceso de medición para conocer cómo exceder la satisfacción del usuario, lo cual vamos a utilizar como una herramienta de Implantación de la calidad. Para lograr la satisfacción del cliente, ésta se debe contemplar como un proceso de mejora continua, y los usuarios del sistema son el objeto de estudio de medición a través de los requerimientos o necesidades que éste manifieste.

Satisfacción del cliente = incrementar la lealtad del cliente y las tareas del equipo.

#### **1.8.3.1.1. Objetivos Principales de la Investigación de la Satisfacción del Cliente (Usuario)**

Los objetivos que se persiguen en este punto son:

- Determinar los rasgos básicos de rendimiento que dan como resultado el satisfacer y exceder las expectativas en calidad del usuario.
- Evaluar el desempeño del equipo y la de la competencia (competidor principal).
- Establecer las prioridades y adoptar las medidas para corregir problemas.
- Controlar los progresos.

Bajo una investigación o medición exacta de la satisfacción del cliente podremos lograr los siguientes resultados:

- Conocer las demandas y expectativas de los clientes;
- Medir el grado de satisfacción;
- Desarrollar muestras de servicio;
- Identificar las tendencias;
- Establecer comparaciones con la competencia.

#### **1.8.3.1.2. Etapas**

El proceso de medición de satisfacción al cliente cuenta con 5 etapas, las cuales son:

- a) **Planeación del proyecto.** Esto es indicar las tareas, dar el rol de cada persona en qué tarea y lograr compromiso de cada persona con su tarea.
- b) **Mediciones.** Las mediciones se obtienen en base a los resultados de los cuestionarios realizados, de los internos y los de usuario. La medición planifica la puesta en práctica de las formas directas de investigación.
- c) **Diagnóstico.** Se obtiene a partir de los resultados de los cuestionarios analiza el rendimiento de la empresa para brindar satisfacción al cliente.
- d) **Evaluación de las actividades.** Este punto es posterior a la implementación de las mejoras tomadas de los cuestionarios.
- e) **Mejora productiva continua.** Esto significa una retroalimentación, no cerrar este ciclo, sino continuar con él en lo sucesivo.

### 1.8.3.1.3. Exceder la Satisfacción del Cliente

Es importante que los resultados de la investigación de la satisfacción del cliente, en este caso el usuario, se difundan a todo el equipo, o a los elementos que deban saber los resultados de ésta, para así lograr los cambios necesarios. También es bueno que se sepa que es lo que se persigue con estas medidas, lo que el equipo va a lograr con estas disposiciones y estar en constante comunicación para que esto se sepa y se logre.

La obtención de resultados sobre la satisfacción del cliente es una herramienta de diagnóstico que debe verse reflejada en cuatro objetivos generales y decisiones.

1. Mejorar el desempeño del equipo, en relación con el logrado hasta el momento en proyectos de software anteriores y similares.
2. Mejorar el desempeño del grupo, en relación a las expectativas futuras del usuario en los programas o servicios que espera los próximos años.
3. Identificar proyectos, claves de mejora continua para exceder las expectativas en calidad del usuario en programas, servicios o atención.
4. El equipo tiene que decidir en qué áreas es necesario ejecutar acciones de mejora continua en: análisis, diseño, políticas y procedimientos, estructura de la organización, comunicación con los usuarios, disponibilidad del software, tecnologías, servicio y calidad, entrenamiento, etcétera, según los resultados del análisis y los alcances del equipo.

Las claves para lograr mantener y exceder la satisfacción del usuario son:

- El mejoramiento productivo es continuo, no se detiene. La mejora productiva continua es un programa a largo plazo y permanente.
- Se logra a través de proporcionar las herramientas necesarias para un servicio óptimo en el ramo.
- El sobrepasar la satisfacción del cliente debe ser la prioridad del equipo a mediano y largo plazo.

Es de vital importancia que se siga revisando periódicamente el rendimiento de todos los puntos de importancia destacados por los usuarios, así como los errores o posibles mejoras que se pueden lograr como equipo de desarrollo, ya que estos factores cambian constantemente para permanecer vigentes.

### 1.8.3.2. Benchmarking

Es un proceso sistemático y continuo para evaluar o comparar las prácticas organizacionales, servicios y procesos de trabajo de la Organización, con respecto a aquellos de las compañías que son reconocidas como las mejores en su clase, con el propósito de realizar mejoras a las prácticas organizacionales que lo ameriten. Lo que es aprender de otros.

Es un proceso de análisis comparativo de capacidades organizativas y productivas que proporciona a las industrias una forma de mejorar.

#### 1.8.3.2.1. Los Indicadores y los Facilitadores del Benchmarking

- **Indicadores de desempeño.** Las medidas individuales de comparación del desempeño se denominan "*Benchmarks*" o indicadores del análisis comparativo del desempeño.
- **Facilitadores.** Se refieren a las acciones en que se traducen dichos indicadores, dentro de la organización, para adaptar la información obtenida a los requerimientos propios. Los "*facilitadores*" se aplican de acuerdo al ciclo Deming de Calidad (Planea, Actúa, Corrige y Estandariza: PACE).

#### 1.8.3.2.2. Benchmarking en Cinco Etapas

El proceso básico de Benchmarking sigue un Plan Básico en cinco etapas:

**1. Planificar.** Determinar a qué se le va aplicar el Benchmarking. Es necesario seleccionar y definir el proceso que va a ser analizado; Identificar los indicadores de desempeño, así como los facilitadores del Benchmarking en función de analizar los resultados.

Deberá definirse que áreas de la organización serán objeto de comparación, así como las empresas externas que deberían ser estudiadas. Definido esto, se procede a identificar y asegurar los recursos necesarios para llevar a cabo una exitosa investigación.

**2. Actuar.** Formar Un Equipo De Benchmarking. El proceso de elegir, orientar y dirigir un equipo es la segunda etapa del proceso de análisis comparativo de las capacidades organizativas y productivas de la organización. Se asignarán las responsabilidades específicas a los miembros del equipo.

Se introducen herramientas de manejo de proyectos, para garantizar que las tareas de Benchmarking sean claras para todas las personas involucradas.

**3. Corregir.** Identificar Los Socios Del Benchmarking. La tercera etapa del proceso consiste en identificar las fuentes de información de las industrias con las cuales se comparará la organización.

Estas fuentes son empleados de organizaciones en las que se practica el Benchmarking: asesores, analistas, fuentes gubernamentales, etc. También se incluye en esta etapa el proceso de identificación de las mejores prácticas industriales y organizativas.

**4. Estandarizar.** Recopilar Y Analizar La Información De Benchmarking. En la cuarta etapa se recopila la información de acuerdo con el protocolo establecido y luego se resume para hacer el análisis. El análisis consiste en dos aspectos: la determinación de la amplitud de las brechas de desempeño de la empresa con respecto al de las organizaciones analizadas, utilizando las medidas o indicadores de Benchmarking.

La identificación de los facilitadores de proceso que permitieron mejoramientos del desempeño en las organizaciones líderes contra las cuales se llevó a cabo el análisis comparativo.

**5. Retroalimentación.** Adaptar, mejorar y aplicar los facilitadores. La etapa final del proceso incluye la adaptación, el Mejoramiento y la implantación de los facilitadores. El objetivo del Benchmarking es introducir cambios en la organización de tal manera que mejore su desempeño.

Implica el llevar a cabo las acciones seleccionadas, priorizadas y jerarquizadas, las cuales contribuyan a una elevación segura en el desempeño de la empresa.

Los resultados del análisis del Benchmarking son además un elemento fundamental para la identificación de objetivos y metas de mejora de la calidad productiva y organizativa que conduzcan, en el mediano plazo de tres a cinco años futuros, al mejor desempeño de las prácticas de la organización.

#### **1.8.3.2.3. Tipos de Benchmarking**

**a) Interno.** Comparación de actividades similares entre diferentes sitios, departamentos o unidades operativas de la misma Empresa.

Ventajas:

- Los datos suelen ser fáciles de recopilar.
- Buenos resultados para compañías "excelentes" que están diversificadas.

**b) Competitivo.** Competidores directos que venden a la misma base de clientes.

Ventajas:

- Información concerniente a los resultados del negocio.
- Prácticas o tecnologías comparables.
- Recopilación de información.

**c) Funcional (genérico).** Entre organizaciones acreditadas por tener lo más avanzado en productos, servicios o procesos.

Ejemplos:

- Almacenamiento (L.L, Bean).
- Rastreo del estado de despachos (Federal Express).
- Servicio al cliente (American Express).
- Ventajas:
- Alto potencial para descubrir prácticas innovadoras.

- Tecnología o prácticas fácilmente transferibles.
- Desarrollo de redes profesionales
- Acceso a bases de datos pertinentes.
- Resultados estimulantes.

### 1.8.3.3. Deficiencias y Fortalezas Productivas

Esta herramienta consiste, como su nombre lo dice, en detectar las fortalezas y deficiencias dentro de una organización, empresa, entidad o grupo de trabajo y conduce hacia las oportunidades de mejora continua.

El diagnóstico se realiza a través de integrar un inventario de los problemas que afectan la calidad, rentabilidad, eficiencia, y seguridad de la organización.

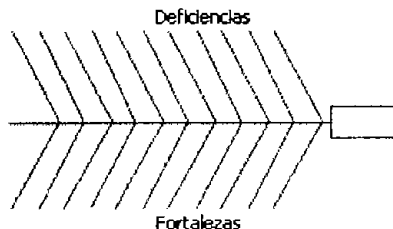
El inventario y el diagnóstico de oportunidades de mejora continua, se integra aplicando la técnica de "lluvia de ideas". La cual debe estar orientada a:

- Identificar y jerarquizar las deficiencias de calidad organizacional que pueden interferir con el buen desempeño y la productividad del grupo.
- Identificar y jerarquizar las fortalezas de calidad organizacional y productiva que proporcionan una ventaja competitiva al grupo.
- Identificar y jerarquizar las oportunidades de mejora continua más prácticas, rentables y factibles que conduzcan a elevar la productividad y la competitividad del grupo.
- Obtener el diagnóstico de debilidades y fortalezas de calidad organizacional y de mejora productiva continua.
- La identificación de acciones estratégicas de mejora continua.

La técnica de lluvia de ideas para la mejora productiva continua se aplica a través de 3 pasos sencillos, los cuales son:

1. Un grupo de trabajo o departamento elige un tema o proyecto específico.
2. Identifica el conjunto de deficiencias y fortalezas de calidad, productivas, administrativas y organizacionales.
3. Las deficiencias se agrupan en un "listado" para realizar el ejercicio de las "5M's"

#### 1.8.3.3.1. Forma de Aplicar la Técnica de las 5 M's



De manera preliminar se identifica el origen de las deficiencias de calidad productiva, administrativas y organizacionales que fueron determinadas en este estudio en:

- Método
- Materiales
- Equipo
- Mano de obra
- Medio ambiente

Para integrar el inventario y diagnóstico de deficiencias de calidad organizacional debe hacerse lo siguiente:

- Las deficiencias o problemas categorizados presentados son el instrumento de aplicación práctica de la estrategia, así como el soporte para fortalecer y consolidar la norma ISO 9000.
- Las deficiencias categorizadas sustentan el programa anual de calidad y mejora productiva continua, de la empresa en el marco de la norma ISO 9000.
- Junto con los resultados del análisis de Benchmarking, del análisis de la voz del cliente y del estudio de mercado. La identificación de las oportunidades de mejora productiva continua, jerarquizadas y priorizadas, serán la base para el diseño del plan de mejora continua de calidad productiva de la empresa en el mediano plazo de tres a cinco años, el cual se conducirá a través de la estrategia Hoshin-Kanri.
- También, la base de información obtenida, será relevante para sustentar la proyección de la empresa, hacia la clase mundial.

# CAPÍTULO 2. CARACTERÍSTICAS DE LOS PROGRAMAS EDUCATIVOS Y DE INFORMACIÓN

---

## 2.1. Software Educativo

Con el desarrollo de la tecnología en diversas áreas, la aplicación de nuevas herramientas, como son las computadoras, los diversos dispositivos de vídeo, el uso de compactos, grabadoras, proyectores y muchos más, los apoyos para el área educativa y para los docentes se incrementan. Cuando se menciona esta integración entre la tecnología relacionada con la computadora y la educación se hace referencia a lo que generalmente se llama Software Educativo, en donde, con estos soportes se abre la posibilidad de combinar textos con fotografías, ilustraciones, vídeos, audio y demás elementos multimedia.

Por lo anterior se puede decir que el software educativo surge de la integración de los elementos multimedia en el desarrollo de material destinado para la educación. Se denomina Software Educativo a los programas. Se denomina Software Educativo a los programas para computadora que han sido desarrollados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje.<sup>8</sup>

El propósito del presente capítulo es presentar los parámetros manejados para el software educativo y los que maneja la ingeniería de software y así hacer notorias las formas en las que éstas ayudan al proyecto.

### 2.1.1. Características Distintivas de los Programas Educativos

Los programas educativos pueden tratar de las mismas materias o valerse de formas similares a la enseñanza común del aula, pero este tipo de software tiene características distintivas que se desprenden de la misma definición que se le da al software educativo, las cuales son:

1. Son elaborados con una finalidad didáctica. Es decir con el propósito específico de ser utilizados para la enseñanza.
2. Usan como herramienta de desarrollo la computadora. Los profesores se valen de ella para desarrollar aplicaciones educativas y para propósitos pedagógicos.
3. En ocasiones son interactivos. Permiten un intercambio de información, siempre y cuando se haya planeado de esa manera, pues en ocasiones la información que contienen es meramente informativa y no interactiva, como las presentaciones electrónicas o multimedia.
4. Facilitan la integración de elementos multimedia. Permite lograr la combinación de fotografías, ilustraciones, vídeos, audio, etcétera.

---

<sup>8</sup> "Educación audiovisual." *Enciclopedia® Microsoft® Encarta 2001*. © 1993-2004 Microsoft Corporation.

5. Generalmente son fáciles de usar. Los conocimientos necesarios para poder operar estas aplicaciones son muy pocos, y, con la práctica se perfecciona su dominio.

### **2.1.2. Características de Calidad**

Generalmente los programas deben seguir ciertos parámetros para lograr un buen desarrollo y facilitar así su uso futuro.

### **2.1.3. Facilidad de Uso e Instalación**

Esto se traduce en que la forma en que el usuario interactúe con la aplicación sea fácil, agradable e intuitivo. Que sin mayores explicaciones se pueda intuir como se usa. De igual forma debe ser la instalación, que la configuración de la herramienta sea sencilla, y así el tiempo de capacitación será únicamente el necesario y en algunos casos nulo.

La ingeniería de software nos dice que podemos evaluar una interfaz<sup>9</sup> a través de los atributos de usabilidad<sup>10</sup>, los cuales sirven para que un sistema pueda ser considerado adecuado en su interfaz. Debe observar los siguientes atributos:

- **Aprendizaje.** El tiempo en el que el usuario logra obtener un beneficio del sistema.
- **Velocidad de la operación.** El tiempo que hay desde que el usuario está tramitando una operación hasta que ésta es completada o concluida.
- **Robustez.** Es la forma en que el sistema reacciona ante un error de usuario.
- **Recuperación.** Si el sistema es suficientemente bueno para recuperarse de errores causados por el usuario.
- **Adaptación.** Modelo de trabajo a adoptar de acuerdo al usuario.

#### **2.1.3.1. Compatibilidad**

Esto se refiere al desempeño de la aplicación junto con otras plataformas, software, entornos, usuarios, etcétera. Esto es, la adaptación a diversos contextos. Que el resultado deseado sea siempre el mismo, y que si se usó una aplicación no sea difícil de acceder a ella, así se podrá propagar su uso, sin importar que no se cuente con el software, pues éste se puede obtener fácilmente. También es importante considerar que nuestros datos en sí y los datos de salida, es decir, lo que arroja nuestra aplicación puedan ser trasladados a otros programas.

#### **2.1.3.2. Interfaz**

Generalmente lo que atrae más en esta área es el entorno gráfico, por lo que es importante cuidar los siguientes aspectos:

---

<sup>9</sup> Interfaz es el punto en el que se establece una conexión entre dos elementos, que les permite trabajar juntos. Las interfaces de usuario cuentan con el diseño gráfico, los comandos, mensajes y otros elementos que permiten a un usuario comunicarse con un programa.

<sup>10</sup> Sommerville, Ian. User Interface Design. En su: Software Engineering. USA, Addison Wesley, 2001. pp. 345.



- Se debe atender que el diseño de todos los elementos en conjunto sea claro, que las pantallas resulten atractivas, que no haya excesos o recargos y que las acciones a realizar salten a simple vista, en general, es mantener una pantalla sencilla.
- Uniformidad en todos sus elementos, ya sean títulos, menús, ventanas, iconos, botones, espacios de texto, imágenes, formularios, barras de navegación, barras de estado, hipervínculos, fondo, etcétera. Es decir, que se siga un patrón predeterminado para cada uno de estos componentes en tamaño, ubicación, colores, etcétera.
- Considerar el uso de elementos multimedia en las aplicaciones, pero no abusar de ellos, es decir, utilizarlos solo cuando sea necesario.
- Es importante ver que el estilo de redacción, y el lenguaje sean adecuados para el tipo de usuario o lector a quien se dirige, que no sea rebuscado o demasiado técnico para ellos, facilitar en lo posible la comprensión del punto que estamos tratando de describir.

En la ingeniería de Software se siguen los siguientes aspectos en el diseño de interfaz de usuario<sup>11</sup> que son importantes y complementarios de los ya citados:

- Principios de Diseño de Interfaz. En esta parte se debe atender que se cumplan los siguientes puntos:
  - Familiaridad del usuario. Es la familiaridad con otros sistemas ya utilizados.
  - Consistencia. Igualdad en los componentes.
  - Mínima sorpresa. Que no exista un cambio radical en los componentes o versiones. Que el comportamiento del sistema debe sea predecible
  - Recuperabilidad. Capacidad de recuperación ante posibles daños o pérdidas. A su vez debe contar con:
    - ◎ Confirmación de acciones destructivas
    - ◎ Proveer un recurso de deshacer
  - Guiar al usuario. Proporcionar ayudas. Guías de usuario. Ayuda, arreglos y su manejo.
  - Diversidad de usuarios. Observar las características de todos los posibles usuarios y tratar de incorporarlas. Esto es proporcionar características adecuadas a cada tipo de usuario.
  - Facilidad de uso.
    - ◎ Selección de menús, listas, etcétera.
    - ◎ Llenado de formularios
- Lineamientos Para Diseño de Pantallas
  - Mantener una pantalla sencilla
  - Manejar y mantener una presentación consistente
  - Facilitar los movimientos de usuario entre pantallas, vigilando:
    - ◎ Desplazamiento
    - ◎ Solicitud de mayor detalle

---

<sup>11</sup> Sommerville, Ian. User Interface Design. Software Engineering. USA, Addison – Wesley, 2001. pp. 327-349.

- ⊙ Dialogo de pantalla
- Crear pantallas atractivas y en ellas vigilar:
  - ⊙ Tipo de letra
  - ⊙ Imágenes
  - ⊙ Color
- En las transacciones que se hagan en el sistema se debe cumplir con las siguientes indicaciones:
  - ⊙ El sistema aceptó la entrada
  - ⊙ La entrada se encuentra en forma correcta
  - ⊙ La entrada no se encuentra en forma correcta
  - ⊙ Habrá un retraso en el procesamiento
  - ⊙ La solicitud ha sido concluida
  - ⊙ El sistema es incapaz de concluir la petición
- Lineamientos de Interfaz Web. En este punto es importante:
  - Establecer mapas de sitio
  - Evitar símbolos de construcción
  - Evitar ligas rotas o inservibles
  - Evitar que el usuario recorra la pantalla
  - El diseño no deberá depender de las funciones del navegador
  - Documentación de errores, para posterior corrección antes de la liberación del sistema.
  - Las opciones de navegación deben ser obvias
- Lineamientos Para la Utilización de Color en las Interfaces del Usuario. Ya que el color en las pantallas es un factor de peso al desarrollar se debe:
  - Limitar el número de colores utilizados y ser conservador al momento de utilizarlos.
  - Utilizar un cambio de color para mostrar el cambio en el estado del sistema.
  - Ser cuidadosos al utilizar juegos de colores.
  - Utilizar el código de colores predefinido en una forma consistente.
  - Manejar un grado razonable de retroalimentación para el usuario en dos sentidos:
    - ⊙ Para incrementar el grado de confianza
    - ⊙ Para saber la manera en que incrementa el trabajo

### 2.1.3.3. Contenidos

Además de seleccionar la información que contendrá cada aplicación y su estructura deben considerarse los siguientes puntos<sup>12</sup>:

---

<sup>12</sup> PRESSMAN, Roger S. Métodos De Diseño. Ingeniería del software: Un enfoque práctico. Cuarta Edición. México. McGraw-Hill. 1998. pp. 249-283

- Que la información que se presenta es actual y adecuada. Se debe diferenciar apropiadamente:
  - Los datos y que éstos sean objetivos.
  - Opiniones.
  - Elementos imaginarios.
- No cargar la pantalla con exceso de datos o una interfaz molesta.
- Eliminar faltas de ortografía y cuidar que la redacción sea adecuada.
- Contenidos objetivos e imparciales. Que los textos no contengan información negativa, sin discriminaciones raciales, sociales, sexuales o de credo, o que pudiese agredir a alguien, aún sin ser pretendido.
- Documentar adecuadamente el material presentado.

#### **2.1.3.4. Interacción General**

La forma en que el usuario interactúa con el sistema es la que determina si éste será amigable y de fácil uso, es por ello que conviene observar los siguientes puntos:

- Consistencia. Que todos los componentes conserven un formato en todas las transacciones.
- Ofrecer respuestas significativas. Esto es garantizar que existe una comunicación entre el usuario y la aplicación.
- Pedir confirmación de acciones destructivas.
- Proveer un recurso de deshacer.
- Tratar de no pedir al usuario memorizar información entre acciones.
- Buscar la eficiencia en el diálogo, el movimiento y el pensamiento. Esto se refiere a hacer solo los movimientos estrictamente necesarios, ya sean "click 's", mover barras de desplazamiento, así mismo las acciones que se hagan deben ser de acuerdo a la lógica o sentido común.
- Manejo de errores. El sistema deberá poder tolerar los posibles errores.
- Agrupar las actividades por función y organizar la pantalla de acuerdo a esto.
- Proporcionar ayudas acordes al contexto.
- Usar tiempos verbales y verbos sencillos, así como frases cortas para facilitar la comprensión.

#### **2.1.3.5. Navegación**

Otro punto crucial para que el usuario se sienta "bien" al usar la aplicación que se desarrolle es la forma en que se traslada entre las diversas ventanas de la misma. Para lograr una buena navegación se deben observar los siguientes puntos:

- Tener un sistema de navegación que permita al usuario poseer el control. A deferencia del punto anterior éste no se visualiza, es simplemente el mecanismo por medio del cual se va a poder avanzar, retroceder, o "navegar" a través de los contenidos.

- Que el tiempo de respuesta sea mínimo. Según estudios realizados en la evaluación de interfaces basadas en usuarios, se observó que si una página tarda en descargar su información más de 8 segundos el usuario pierde el interés y decide cancelar la transacción, por lo que es importante no perder mucho tiempo en cargar animaciones, imágenes u otros elementos multimedia que conformen nuestra aplicación.
- Contar con un mapa de navegación, no rebuscado, que permita un desplazamiento a través de las diversas páginas lo más sencilla posible.
- Que cuando el usuario ingrese una petición se le facilite en la mayor medida posible que la respuesta que reciba sea satisfactoria.
- Proporcionar indicaciones (si es necesario) de cómo deberán ser hechas las peticiones, preguntas, acciones o todo lo necesario para que el usuario no tenga dudas a cerca de cómo funciona la aplicación.

La ingeniería de software también contempla los siguientes puntos para llevar a cabo:

- Diseño de Navegación. Se definen las rutas de navegación que permitan al usuario acceder al contenido y a los servicios de las aplicaciones. Para que el diseñador pueda llevarlo a cabo se debe:
  - Identificar la semántica de la navegación para los diferentes usuarios del sitio, lo que se traduce en controles de acceso.
  - Definir la mecánica de cada enlace de navegación, esto se refiere a los enlaces basados en texto, íconos, botones, etcétera.
    - Ⓞ Se deben elegir los enlaces de navegación adecuados para el contenido.
    - Ⓞ Se deben establecer las conexiones y ayudas adecuadas, esto es, íconos y enlaces gráficos con aspecto "CLICKEABLE".
    - Ⓞ Para navegación basada en texto se deberá utilizar el color que indica los enlaces de navegación.
    - Ⓞ Proporcionar una indicación de que se ha elegido una opción de navegación.
    - Ⓞ Proporcionar una indicación de los enlaces por los que se ha navegado.

### **2.1.3.6. Adaptación**

La adaptación es la adecuación a los usuarios y a su trabajo. Es importante tomar en cuenta al tipo de usuarios que se va a dirigir lo que desarrollemos, ya que no se puede hablar de términos demasiado técnicos a personas que apenas saben usar la computadora por ello debemos observar:

- Los contenidos deben ser de interés para el usuario y así mismo debemos atender aspectos como el vocabulario, la estructura, la extensión, profundidad, y contar con ejemplos o gráficos que sean acordes a los contenidos.
- Que la actividad que se realice sea apropiada para el nivel en el que deberá trabajarse con la aplicación.
- Que la interfaz de usuario facilite la comunicación o transmisión de ideas con el usuario.

### **2.1.3.7. Documentación**

Este concepto resulta muy importante desde el punto de vista pedagógico y también para el de la ingeniería de Software, ya que en ambas se destaca la importancia de que se documente todo lo concerniente a la creación de software o aplicaciones, sean manuales de usuario o documentos que pudieran servir de apoyo para modificaciones, mejoras o cambios posteriores.

### **2.1.3.8. Otros Factores de Importancia**

- Originalidad. Debemos velar que nuestra interfaz, la forma en que se realizan las transacciones, el diseño, así como todas y cada una de las partes que conforman nuestra aplicación sean radicalmente (en lo posible) deferentes a las otras o a las anteriormente realizadas.
- Motivación y fomento de la iniciativa. Las aplicaciones que desarrollamos, producen y mantienen curiosidad o interés en el usuario, es cuando se está logrando una motivación, la cual puede contribuir a incrementar el interés y usarlo a favor de la iniciativa y el aprendizaje autónomo de los usuarios así se lograrán mejores avances.
- Uso de tecnología avanzada. Si se cuenta con recursos que permiten tener tecnología de punta, al usar estas herramientas favorecemos a que se conozcan nuevas formas de transmitir el conocimiento, también la forma de organizar la información y usar los diversos medios de comunicación.

## **2.2. Sistemas De Información Orientados a Objetos**

Dado que el sistema que se está llevando a cabo está basado en el desarrollo Orientado a Objetos, en la presente sección se presentará una definición de las principales características y tareas que se llevan a cabo en cada etapa de "*fabricación*" del proyecto. Además se enlistarán las principales labores que se llevan a cabo en cada una de estas fases, tal y como se había expresado en secciones anteriores en cuanto a organización.

Un sistema de información Orientado a Objetos es el conjunto de dispositivos que colaboran en la realización de una tarea, se refiere a la colección o combinación de programas, procedimientos, datos y equipamiento utilizado en el procesamiento de información, pero de forma que el enfoque que se le dé no sea el tradicional, el estructurado, sino la visión orientada a objetos, la cual es un conjunto de objetos limitados que, a su vez, son colecciones independientes de estructuras de datos y rutinas que interactúan con otros objetos. Es decir, todas las tareas, entidades y posibles operaciones se ven en términos de objetos, clases, instancias, métodos, y mensajes.

En contraste con el enfoque estructurado es radicalmente diferente, ya que en éste se dividen operaciones, y tiene un tipo de programación que produce código con un flujo limpio, un diseño claro y un cierto grado de modularidad, que se puede entender mejor como una estructura jerárquica, en cambio en el punto de vista OO las operaciones se pueden concentrar en un solo objeto, aunque no sean exactamente lo mismo.

## 2.2.1. Conceptos del enfoque OO

Como en el capítulo 1 se definieron conceptos importantes es primordial describir una serie de conceptos para que pueda comprenderse el concepto OO, para ello se describen algunos en los siguientes párrafos.

**Clase.** Es un concepto OO que encapsula las abstracciones de datos y procedimientos que se requieren para describir el contenido y comportamiento de alguna entidad del mundo real. Contiene características generales, llamados atributos. Encapsula datos que son atributos y métodos, entre otras cosas. Una clase es una descripción generalizada que describe una colección de objetos similares. Todos los objetos que existen dentro de una clase heredan sus atributos y las operaciones disponibles para la manipulación de los atributos. Las clases OO y objetos derivados de ellas encapsulan los datos y las operaciones que trabajan sobre estos en un único paquete.

**Superclase.** Una superclase es una colección de clases.

**Subclase.** Es la instancia de una clase.

**Objeto.** Es una entidad que encapsula datos representados como una colección de atributos y los algoritmos que procesan estos datos, además contienen operaciones, otros objetos, y otra información relacionada con la entidad.

**Atributo.** Describe la clase. Son las características que pertenecen a una clase. Los atributos describen la clase o el objeto de alguna manera. Puede tomar un valor definido por un dominio determinado. Los atributos describen un objeto que ha sido seleccionado para ser incluido en el modelo de análisis, los atributos definen al objeto, clarifican lo que se representa con el objeto en el contexto del espacio del problema.

**Métodos, operaciones o servicios.** Son abstracciones procedimentales capaces de manipular los datos. La única forma de alcanzar los atributos y operar sobre ellos es ir a través de alguno de los métodos. Son procesos que manipulan los atributos de la clase y datos. También se les llama operaciones, y son algoritmos que procesan a los atributos. Las operaciones definen el comportamiento de un objeto y cambian, de alguna manera, los atributos de éste.

**Mensajes.** El paso de mensajes mantiene comunicados a los integrantes de un sistema OO. Los mensajes son el medio a través del cual los objetos interactúan, un mensaje estimula la ocurrencia de cierto comportamiento en el objeto receptor, el comportamiento se realiza cuando se ejecuta una operación. Tiene la siguiente sintaxis: "*mensaje = [destino, operación, parámetros]*". Aquí *destino* es el objeto receptor que es estimulado por el mensaje, *operación* se refiere al método que recibe el mensaje y *parámetros* proporciona información requerida para el éxito de la operación.

**Instancia.** Es el miembro de una clase.

**Encapsulamiento.** Significa que toda la información se encuentra empaquetada bajo un nombre y puede reutilizarse como una especificación o componente de un programa.

**Dominio.** Es un conjunto o rango de valores específicos.

**Jerarquía de clases.** Es la forma en que se organizan las clases, subclases y superclases. En ella los atributos y operaciones de la superclase son heredados por subclases que pueden añadir, cada una de ellas, atributos privados y métodos.

**Herencia.** La herencia es una de las diferencias clave entre sistemas convencionales y sistemas OO. Una subclase hereda todos los atributos y operaciones asociadas con su superclase. Esto significa que todas las estructuras de datos y algoritmos originalmente diseñados e implementados para la superclase están inmediatamente disponibles para la subclase (no es necesario más trabajo extra). La reutilización se realiza directamente. Cualquier cambio en los datos u operaciones contenidas dentro de una superclase se hereda inmediatamente por todas las subclases que se derivan de la superclase.

Debido a esto, la jerarquía de clases se convierte en un mecanismo a través del cual los cambios (a altos niveles) pueden propagarse irremediablemente a través de todo el sistema.

**Anulación.** La anulación ocurre cuando los atributos y operaciones se heredan de manera normal, pero después son modificados según las necesidades específicas de la nueva clase.

**Herencia múltiple.** A veces parece más fácil heredar algunos atributos y operaciones de una clase y otros de otra clase, a esto se le llama herencia múltiple. A veces puede generar impactos no deseados en las clases inferiores a ella.

**Polimorfismo.** Es una característica que reduce en gran medida el esfuerzo necesario para extender un sistema OO. Permite que un número de operaciones diferentes tengan el mismo nombre. Esto reduce el acoplamiento entre objetos, haciéndolos (a cada uno) más independientes.

## 2.2.2. Características de Los Sistemas OO

Hay 3 características de sistemas OO que los hacen únicos, las cuales son:

1. **Ocultamiento de la información.** Los detalles de implementación interna de datos y procedimientos están ocultos al mundo exterior. Esto reduce la propagación de efectos colaterales cuando ocurren cambios.
2. **Reutilización de componentes.** Las estructuras de datos y las operaciones que las manipulan están mezcladas en una entidad sencilla: la clase. Esto facilita la reutilización de componentes.
3. **Interfaces simplificadas.** Un objeto que envía un mensaje no se tiene que preocupar de los detalles de estructuras de datos internas en el objeto receptor. Por tanto, se simplifica la interacción y el acoplamiento del sistema tiende a reducirse. Las interfaces entre objetos encapsulados están simplificadas.

### 2.2.3. Ventajas del Enfoque OO

Cada libro describe diferentes métodos y enfoques para hacer sistemas, pero tras estudiar las diferentes maneras de hacerlo decidí hacer una descripción de las características del usado en el *"Sistema de Comunidades de Aprendizaje Iztacalá"* con el enfoque OO, y a continuación describo las ventajas de éste para que se observe porqué se eligió esta perspectiva. Las ventajas de la visión Orientada a objetos son:

- ⇒ Las tecnologías OO llevan a reutilizar y esto lleva a un desarrollo de software más rápido y programas de mejor calidad.
- ⇒ El software OO es más fácil de mantener, por que su estructura es relativamente descompuesta.
- ⇒ Los cambios que vayan a hacerse provocan menos efectos colaterales gracias a la estructura de jerarquías que maneja.
- ⇒ Los sistemas OO son fáciles de adaptar y escalar, ya que se pueden crear nuevos sistemas a partir de componentes existentes.
- ⇒ Gracias al encapsulamiento de datos se posibilita el ocultamiento de información y reduce el impacto de modificaciones mayores o efectos colaterales asociados a cambios.
- ⇒ Las características anteriores hacen referencia a las características de diseño descritas en el inicio del presente capítulo, lo cual hace hincapié en la visión OO, que resulta conveniente para el sistema en cuestión.

### 2.2.4. Gestión de Proyectos de Software Orientado a Objetos

Como se sabe, respecto a OO todo cambia, en este punto no se hace excepción, los principios de gestión fundamentales serán los mismos, lo que cambia es la técnica que debe ser adaptada de tal manera que un proyecto OO sea dirigido correctamente. La gestión de proyectos de software es la dedicada a dar tiempos y medidas a los proyectos de software.

#### 2.2.4.1. Estimaciones y Planificación OO

Las estimaciones pueden realizarse usando valoraciones del esfuerzo y la duración para el desarrollo de software convencional pero debido a la naturaleza del proyecto vale la pena usar un enfoque diseñado para software OO. Lorenz y Kidd sugieren el siguiente enfoque:

- a. Sacar estimaciones usando la descomposición de esfuerzos, análisis de Punto de función (PF) u otros métodos convencionales.
- b. Desarrollar guiones de escenario usando AOO (Análisis Orientado a Objetos) y contarlos. El número de guiones de escenarios puede cambiar con el proyecto.
- c. Determinar la cantidad de clases clave usando AOO.
- d. Clasificar el tipo de interfaz para la aplicación según la siguiente tabla:



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

Una vez hecho esto, desarrollar un multiplicador para las clases de soporte. Multiplicar el número de clases clave por el multiplicador para obtener una estimación del número de clases de soporte.

- e. Multiplicar la cantidad total de clases clave + soporte por el número de unidades de trabajo por clase. Se sugiere entre 15 y 20 días persona por clase.
- f. Comprobar la estimación respecto a clases multiplicando el número promedio de unidades de trabajo por guión de acción.

### **2.2.4.2. Seguimiento de Proyectos OO**

Para el seguimiento de un proyecto se debe ver que se cumpla con ciertos puntos por cada etapa, los cuales son:

#### **1. Para el análisis OO**

- Todas las clases y la jerarquía de clases están definidas y revisadas.
- Los atributos de clases y las operaciones asociadas a una clase se han definido y revisado.
- Las relaciones entre clases se han establecido y revisado.
- Se ha creado y revisado un modelo de comportamiento.
- Se han marcado clases reutilizables.

#### **2. En el diseño OO**

- El conjunto de subclases se ha definido y revisado.
- Las clases se han asignado a subclases y han sido revisadas.
- Se ha establecido y revisado la asignación de tareas.
- Se han identificado responsabilidades y colaboraciones.
- Se han diseñado y revisado los atributos y métodos.
- El paso de mensajes se ha creado y revisado.

#### **3. Programación OO**

- Cada nueva clase ha sido codificada a partir del modelo de diseño.
- Las clases extraídas de una biblioteca de reutilización se han integrado.
- Se ha construido un prototipo.

#### **4. Pruebas OO**

- La corrección y completación del AOO y el modelo de diseño han sido revisados.
- Se ha desarrollado y revisado las clases, responsabilidades y colaboraciones.
- Se han diseñado casos de prueba y ejecutado pruebas a nivel de clases para cada clase y las clases se han integrado.
- Las pruebas a nivel de sistema se han terminado.

## 2.2.5. Análisis

El análisis de un sistema sirve para detectar los requerimientos para desarrollar una aplicación. La tarea del análisis es descubrir qué se necesita, refinar lo que se vaya obteniendo en base a la detección de necesidades, pulir a detalle, se especifican cómo se va a ocupar qué cosa, y en qué condiciones; por último se modelan partes de la aplicación en base al comportamiento total del sistema.

Es importante saber que el análisis se define como la investigación que separa un problema en partes, el proceso de clasificación e interpretación de un problema determinado, es el estudio del problema y el establecimiento de las necesidades.

Por lo anterior podemos concluir que el proceso de análisis es la actividad que permite la fragmentación de un todo complejo en sus componentes para realizar un estudio individual. Es el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física.

El AOO es la parte que se encarga de definir los resultados esperados, o la información que deberá obtenerse, esto es que se entienda el problema en términos de datos disponibles y definir el proceso necesario para convertirlos en la información requerida

### 2.2.5.1. Tareas del Análisis

El análisis de un sistema OO puede resultar algo complejo, lo cual depende de la complejidad y el tamaño del mismo, entre otras cosas, pero sin importar el tamaño de la aplicación habrá actividades específicas a realizar, dentro de ellas se realizan las siguientes.

El objetivo del AOO es desarrollar una serie de modelos que describan el software de computadora al trabajar para satisfacer un conjunto de requisitos definidos por el cliente. El modelo de análisis nos sirve para ilustrar información, funcionamiento y comportamiento dentro del contexto de los elementos del modelo de objetos.

#### 2.2.5.1.1. Definición de Requerimientos.

La definición de requerimientos no es propiamente una parte del análisis, pero por la naturaleza del proyecto SCAI<sup>13</sup> este punto formará parte del análisis. Los requerimientos de un sistema muestran lo que un sistema debe hacer y define restricciones en su operación e implementación. Hay diferentes tipos de requerimientos, los cuales son:

1. **Requerimientos de usuario.** Se describen en lenguaje natural y diagramas, y lo que describen es qué servicios se espera que el sistema provea y las restricciones bajo las que debe operar.

---

<sup>13</sup> Sistema de Comunidades de Aprendizaje Iztacala

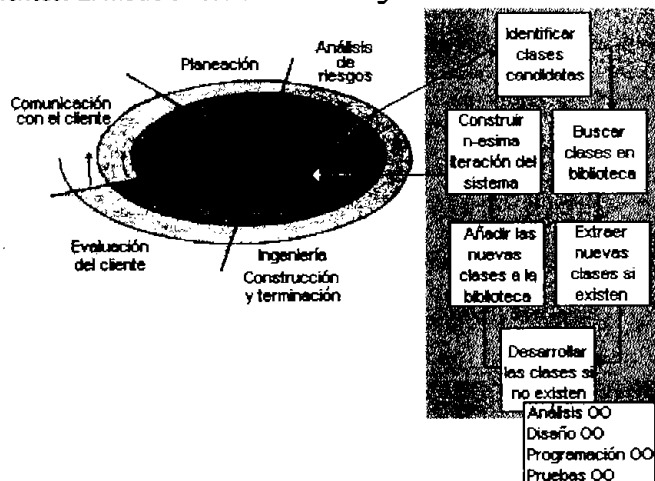
2. **Requerimientos de sistema.** Muestran los requerimientos y servicios del sistema en detalle. Aquí se encuentra contenido el documento de requerimientos de sistema, también llamado especificaciones funcionales, y describe con gran precisión las funciones del sistema. Es generalmente el contrato entre el desarrollador y el comprador. Los requerimientos de sistema se clasifican frecuentemente en los siguientes puntos:
  - a. Requerimientos funcionales.
  - b. Requerimientos no funcionales.
  - c. Requerimientos de dominio.
3. **Especificación de diseño del software.** Es una descripción abstracta del diseño del software, es la base detallada del diseño e implementación y añade más detalle a la especificación de requerimientos de sistema.

Sin duda hay mucho más detalle en estas tareas, pero para efectos del desarrollo de este proyecto, no se considera relevante la descripción detallada de estas características, ya que solo se desea dar una breve descripción de los pasos que se seguirán para el logro de este proyecto.

### 2.2.5.1.2. Definición de un Modelo de Procesos OO.

Un modelo de procesos es la utilización de herramientas, métodos y estrategias que nos permite desarrollar un sistema en partes. Hay varios modelos, pero para la visión OO se recurre a los dos siguientes:

1. **Modelo de Procesos OO.** La IS desarrolló un modelo de procesos para describir el proceso orientación a objetos, está basado en la inmersión del Modelo de Proceso de Ensamblaje de Componentes en el Modelo Evolutivo de Proceso, y está acoplado de forma que se pueda manejar la reutilización de componentes. El modelo resultante es el siguiente:



OO = objetos + clasificación + herencia + comunicación

La figura nos muestra el proceso OO que se mueve a través de una espiral evolutiva que comienza con la comunicación con el usuario, se define el dominio del problema y se identifican las clases básicas del problema. La planificación y el análisis de riesgos establecen una base para el plan del proyecto OO. Luego se siguen las tareas especificadas en el recuadro sombreado de forma repetida que forman parte de la etapa de Ingeniería, Construcción y Terminación.

2. **Modelo recursivo/paralelo.** Otro modelo de procesos OO es el "Modelo recursivo/paralelo". Generalmente se usa el modelo de proceso OO mencionado en párrafos anteriores, pero Ed Berard<sup>14</sup> y Grady Booch<sup>15</sup>, entre otros sugieren el uso de un Modelo recursivo/paralelo para el desarrollo de software orientado a objetos.

El modelo debe hacerse descomponiendo sistemáticamente el problema en componentes altamente independientes, aplicar de nuevo el proceso de descomposición a cada una de los componentes independientes para, a su vez, descomponerlos, que es la parte recursiva, conducir este proceso de reaplicar la descomposición de forma recurrente sobre todos los componentes, y finalmente continuar este proceso hasta completar el sistema al cien por ciento.

Es importante observar que si se detecta que un componente o subcomponente está disponible en una biblioteca de reutilización ya no se hará el proceso y solo se extraerá de ella. Es decir, que cada repetición o iteración del proceso recursivo paralelo requiere planificación, ingeniería análisis, diseño, extracción de clases, prototipado y pruebas.

### 2.2.5.1.3. Elección de un Método de AOO

Hoy en día hay muchos métodos de AOO, pero el propósito de esta sección es definir solo las características principales de algunos de estos métodos, serán los más utilizados y decir cuál ha de ser aplicado para el SCAI<sup>16</sup>. La diferencia de un método y un modelo es que el modelo nos sirve para controlar el proceso, y llevar un control sobre las actividades a desempeñar durante cada etapa. El método nos permite desarrollar las actividades propias de cada etapa en la creación de un sistema, como son el análisis, diseño, etcétera. Dicho de otra forma, el método nos ayuda a identificar partes funcionales dentro del sistema.

1. **El método de Coad y Yourdon**<sup>17</sup>. Este es el método que se considera con frecuencia, como uno de los más sencillos de aprender, ya que la notación del modelado es simple y las reglas para desarrollar el modelo de análisis son claras.

---

<sup>14</sup> Berard, E. V., *Essays on Object Oriented Software Engineering*, Addison – Wesley, 1993.

<sup>15</sup> Booch, G., *Object Oriented Design*, Benjamins Cummings, 1991

<sup>16</sup> Sistema de Comunidades de Aprendizaje Iztacala

<sup>17</sup> Coad, P. y E. Yourdon, *Object Oriented Analysis*, 2a edición, Prentice Hall, 1991

2. **El método de Jacobson**<sup>18</sup>. También llamado Ingeniería del Software OO (ISOO). Este método se diferencia de los otros por la importancia que da al "caso de uso"<sup>19</sup>.
3. **-El método de Rumbaugh**<sup>20</sup>. Rumbaugh y sus colegas desarrollaron la Técnica del modelado de Objetos<sup>21</sup> para el análisis, diseño del sistema y diseño del nivel de objetos.
4. **El método de Wirfs-Brock**<sup>22</sup>. Propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño.
5. **El método de Booch**. El método de Booch<sup>23</sup> abarca un microproceso de desarrollo que define un conjunto de tareas de análisis que se reaplican en cada etapa en el macroproceso, lo que mantiene un enfoque evolutivo.

El método Booch está soportado por una variedad de herramientas automatizadas. El microproceso de desarrollo consiste en:

- Identificar clases y objetos.
  - Proponer objetos candidatos.
  - Conducir el análisis de comportamiento.
  - Identificar escenarios relevantes (significativos).
  - Definir atributos y operaciones para cada clase.
- Identificar la semántica de clases y objetos.
  - Seleccionar y analizar escenarios.
  - Asignar responsabilidades para alcanzar el comportamiento deseado.
  - Dividir las responsabilidades para equilibrar el comportamiento.
  - Seleccionar un objeto y enumerar sus papeles y responsabilidades.
  - Definir operaciones para satisfacer las responsabilidades.
  - Buscar colaboraciones entre objetos.
- Identificar relaciones entre clases y objetos
  - Definir las dependencias que existen entre objetos.
  - Describir el papel (rol) de cada objeto participante
  - Validar los escenarios por revisión completa
- Realizar una serie de refinamientos.
  - Producir diagramas apropiados para el trabajo realizado en los puntos anteriores.
  - Definir jerarquías de clases apropiadas.
  - Crear agrupamientos basados en clases comunes.

---

<sup>18</sup> Jacobson, I., Object Oriented Software Engineering, Addison – Wesley, 1992.

<sup>19</sup> El Caso de uso es una descripción o escenario que describe cómo el usuario interactúa con el producto o sistema.

<sup>20</sup> Rumbaugh et al., Object Oriented Modeling and Design, Prentice Hall, 1991.

<sup>21</sup> Object Modeling Techniques.

<sup>22</sup> Wirfs-Brock, R., B. Willkerson y L. Weiner, Designing Object Oriented Software, Prentice Hall, 1990.

<sup>23</sup> Booch, Grady., Análisis y Diseño Orientado a Objetos, 2ª edición, Addison Wesley, 1996.

- Implementar clases y objetos (en el contexto del AOO esto implica complementar el modelo de análisis).

Este último método, el de Booch, es el elegido para realizar el Análisis de este sistema, ya que el analista se identificaba con él y conocía la metodología para ello. Es por esto que a continuación se hace una descripción más detallada del modelo.

### a) Etapas

**Análisis de requerimientos.** En esta etapa se define qué quiere el usuario del sistema. Es una etapa de alto nivel que identifica las funciones principales del sistema, el alcance del modelado del mundo y documenta los procesos principales y las políticas que el sistema va a soportar. En este punto se llevan a cabo las siguientes actividades:

- Funciones primarias del sistema. Principales entradas y salidas del sistema, referencias a políticas, sistemas existentes o procedimientos, etc.
- Conjunto de mecanismos claves que el sistema debe proveer. Estado de entrada, estado de salida y estados esperados.

**Análisis de Dominio.** El análisis del dominio del software<sup>24</sup> es la identificación, análisis y especificación de requisitos comunes de un dominio de aplicación específico, normalmente para su reusabilidad en múltiples proyectos dentro del mismo dominio de aplicación. El análisis orientado a objetos del dominio es la identificación, análisis y especificación de capacidades comunes y reusables dentro de un dominio de aplicación específico, en términos de objetos, clases, y marcos de trabajo comunes. Es el proceso de definir de una manera concisa, precisa y OO la parte del modelo del mundo del sistema. Las siguientes actividades son parte de esta etapa:

- Definir Clases. En este paso se debe empezar a identificar objetos y clases a través del planteamiento del problema o especificación de requerimientos. Los objetos se determinan subrayando cada nombre o requisito representativo e introduciéndola en una tabla con dos columnas, una del objeto o clase potencial y otra con la clasificación general.

Destacando sinónimos, para que no se preste a confusiones. Habrá objetos de dos tipos: el espacio de solución, que debe resolver algún requerimiento, o implementar alguna solución, y el espacio de problema, que es cuando un objeto se necesita solamente para describir una solución. Los objetos pueden ser:

- Entidades externas. Otros sistemas, dispositivos, o personas que producen o consumen información a usar por un sistema.
- Cosas. Son parte del dominio de información del problema: informes, presentaciones. cartas, señales, etc.

---

<sup>24</sup> Firesmith, D. G., Object Oriented Requirements Analysis and Logical Design, Wiley, 1993.

- Ocurrencias o eventos. Suceden dentro del ámbito de operación del sistema: transferencia de propiedad o la terminación de una serie de movimientos de una máquina.
- Papeles o roles. Son desempeñados por personas que interactúan con el sistema, como el director, ingeniero, vendedor, etcétera.
- Unidades organizacionales. Que sean relevantes en una aplicación (división, grupo, equipo).
- Lugares. Solo que establezcan el contexto del problema y la función general del sistema, como puede ser una planta de producción.
- Estructuras. Definen una clase de objetos o, clases relacionadas de objetos. Los ejemplos pueden ser sensores, vehículos de cuatro ruedas o computadoras.
- Definir relaciones. Las clases colaboran unas con otras de diversas maneras, las relaciones básicas son asociación, herencia, posesión, y uso.
- Encontrar atributos. Al desarrollar un conjunto significativo de atributos para un objeto, se debe estudiar de nuevo la narrativa de proceso o la descripción del sistema para el problema y seleccionar aquellos elementos que razonablemente pertenecen al objeto. Se debe ver qué elementos compuestos y/o simples definen completamente al objeto en el contexto del problema actual. De manera que los atributos de un objeto podrían quedar de la siguiente forma:

Objeto = atributo1 + atributo2 + atributo3...

- Definir herencia. Es una relación entre clases en la que una clase comparte la estructura y / o el comportamiento definidos en una o mas clases, lo que da origen a herencia múltiple o simple, la clase de la que otras heredan se llama superclase y las que heredan de ella se les llama subclase. A través de la herencia se define una jerarquía entre clases.
- Definir operaciones. Una operación (método) cambia valores de uno o más atributos contenidos en el objeto y deben ser implementados de manera tal que permita manipular las estructuras de datos que han sido derivadas de dichos atributos. Los tipos de operaciones pueden clasificarse en tres grandes categorías:

- Operaciones que manipulan de alguna manera datos.
- Operaciones que realizan algún cálculo
- Operaciones que monitorizan un objeto frente a la ocurrencia de un suceso de control.

Se debe estudiar la descripción del sistema y seleccionar aquellas operaciones que razonablemente pertenecen al objeto. También se analizan los mensajes entre objetos, ya que implican un conjunto de operaciones adicionales que deben estar presentes.

- Validar e iterar sobre el modelo. Es importante recordar que todo proceso OO implica revisiones continuas para evolucionar con los requerimientos y en base a los cambios que del diseño se reflejarán en el análisis.

**Recopilación de información.** La metodología de Booch (sección 2.5.5.1.3. punto 5) usa los siguientes tipos de diagramas para describir las decisiones de análisis y diseño, tácticas y estratégicas. Para ello se cuenta con diversas herramientas, las cuales se describen brevemente a continuación:

- Diagrama de Clases. Consisten en un conjunto de clases y relaciones entre ellas. Puede contener clases, clases paramétricas, utilidades y metaclasses. Los tipos de relaciones son asociaciones, contenedora, herencia, uso, instanciación y metaclasses.
- Especificación de Clases. Es usado para capturar toda la información importante acerca de una clase en formato texto.
- Diagrama de Categorías. Muestra clases agrupadas lógicamente bajo varias categorías
- Diagramas de transición de estados. Son diagramas que muestran los diferentes estados que puede tener un objeto.
- Diagramas de Objetos. Muestra objetos en el sistema y su relación lógica. Pueden ser diagramas de escenario, donde se muestra cómo colaboran los objetos en cierta operación, o diagramas de instancia, que muestra la existencia de los objetos y las relaciones estructurales entre ellos.
- Diagramas de Tiempo. Aumenta un diagrama de objetos con información acerca de eventos externos y tiempo de llegada de los mensajes.
- Subsistemas. Un subsistema es una agrupación de módulos, útil en modelos de gran escala.
- Diagramas de módulos. Muestra la localización de objetos y clases en módulos del diseño físico de un sistema. Un diagrama de módulos representa parte o la totalidad de la arquitectura de módulos del sistema.
- Diagramas de procesos. Muestra la localización de los procesos en los distintos procesadores de un ambiente distribuido.

**Documentación a entregar.** Al finalizar este proceso se debe entregar la siguiente documentación:

- Diagrama de clases con las abstracciones clave, identificando las clases del dominio claves y sus relaciones.
- Especificación de las clases.
- Vistas de herencia. Diagramas de clases con este tipo de relaciones.
- Diagramas de escenarios de objetos.
- Especificación de objetos. Relacionan objetos y sus clases.



## 2.2.6. Diseño

El diseño orientado a objetos (DOO) constituye un tipo de diseño que logra un cierto número de niveles de modularidad. Los componentes principales del sistema están organizados en módulos denominados subsistemas. Los datos y las operaciones que manipulan los datos están encapsulados en objetos, una forma modular que es el bloque de construcción de un sistema OO. El DOO debe describir la organización de datos específicos, de atributos y los detalles procedimentales de las operaciones individuales.

El DOO debe encontrar objetos, distribuirlos en clases, definir interfaces de clases y jerarquías de herencia, y establecer relaciones clave entre ellas. El diseño debe ser específico al problema actual pero también lo suficientemente general para resolver problemas y requisitos futuros. Se debe evitar también el rediseño, o al menos minimizarlo.

El DOO da la posibilidad de indicar los objetos que se derivan de cada clase y cómo estos objetos se interrelacionan unos con otros. El DOO ilustra cómo se desarrollan las relaciones entre objetos, cómo se debe implementar el comportamiento y cómo implementar la comunicación entre objetos. El DOO se basa en cuatro importantes conceptos de diseño del software, los cuales son:

1. **Abstracción.** Proceso mediante el cual se identifican los aspectos importantes de un fenómeno y se ignoran los detalles. Dicho proceso debe ser lo más genérico posible, es decir, formar patrones que puedan ser repetibles durante el proceso que nos permita obtener el modelo.
2. **Ocultación de la información.** El principio de ocultamiento de información sugiere que los módulos se caractericen por decisiones de diseño que haga que cada uno se oculte de los demás. Se deberán especificar y diseñar los módulos para que la información, procedimiento y datos contenidos dentro de un módulo sea inaccesible a otros módulos que no necesiten esa información. Los módulos independientes comparten entre sí, sólo la información necesaria para conseguir la función del software.
3. **Independencia funcional.** Se consigue desarrollando módulos con una función única y una «aversión» a excesiva interacción con otros módulos. Se trata de diseñar software de manera que cada módulo trate una subfunción específica de los requisitos y tenga una sencilla interfaz cuando se vea desde otras partes de la estructura del programa. La independencia se mide usando dos criterios cualitativos: cohesión y acoplamiento.
  - La cohesión es una medida de la fuerza relativa funcional de un módulo.
  - El acoplamiento es una medida de la interdependencia relativa entre los módulos.
4. **Modularidad.** Acto de dividir el software en componentes identificables y tratables por separado, a estas partes se les llama módulos y están integrados para satisfacer los requisitos del sistema.

Para DOO podemos esquematizarlo en pirámide, las capas del DOO son:

- **Capa del subsistema.** Contiene una representación de cada uno de los subsistemas. Permiten conseguir los requisitos definidos por el cliente e implementar la infraestructura técnica que los soporte.

- **Capa de clases y objetos.** Contiene las jerarquías de clases que permiten crear el sistema usando generalizaciones y especializaciones definidas incrementalmente. Determina representaciones de diseño para cada objeto.
- **Capa de mensajes.** Contiene los detalles que le permiten a cada objeto comunicarse con sus colaboradores. Establece las interfaces externas e internas para el sistema.
- **Capa de responsabilidades.** Contiene estructuras de datos y el diseño algorítmico para todos los atributos y operaciones de cada objeto.
- **Capa de diseño.** Se centra en el diseño de los objetos del dominio, llamados también patrones de diseño, aportan soporte a las actividades de la interfaz hombre - computadora, gestión de tareas y datos, también pueden usarse para desarrollar el diseño de la aplicación.



Meyer<sup>25</sup> sugiere cinco principios de diseño básicos que pueden utilizarse para arquitecturas modulares:

1. **Unidades modulares lingüísticas.** El lenguaje de programación a usar debe ser capaz de soportar directamente la modularidad definida.
2. **Pocas interfaces.** Debe minimizarse el número de interfaces entre módulos.
3. **Pequeñas Interfaces o bajo acoplamiento.** Debe minimizarse la cantidad de información que se mueva a través de una interfaz.
4. **Interfaces explícitas.** Cada vez que se comuniquen los módulos, deben realizarlo de una manera obvia y directa.
5. **Ocultación de la información.** Se logra cuando toda la información sobre un módulo está oculta al acceso exterior, a menos que la información se defina explícitamente como información pública.

<sup>25</sup> Meyer, Bertrand, Object Oriented Software Construction, 2ª edición, Prentice-Halls. 1998.

## 2.2.6.1. Tareas del Diseño

El DOO es de las partes que demanda mucho esfuerzo y tiempo, ya que además de tomar los datos obtenidos en el análisis, puede llegar a modificar el mismo. Las tareas que se pueden llevar a cabo dentro del diseño se describen en los párrafos siguientes.

### 2.2.6.1.1. Elección de un Método de Doo

Al igual que en el AOO, también hay muchos métodos de DOO, y los mismos descritos en la sección 2.2.5.1.3. serán especificados en esta. Cada uno de los métodos tiene un conjunto de representaciones de diseño y una notación que permite crear un diseño de forma consistente. En los párrafos que siguen, se presentan los métodos de DOO de forma resumida, y así permitir que se elija el que más convenga, en este caso ya se explicó que se usó el de Booch, al cual se le pondrá más empeño al explicarlo. Se mencionarán las características principales de estos métodos.

1. **Método de Coad y Yourdon.** Este método fue desarrollado observando como realizan su trabajo diseñadores especializados del software OO.
2. **Método de Jacobson.** El modelo de diseño hace hincapié en el seguimiento al modelo de análisis ISOO<sup>26</sup>.
3. **Método de Wirfs-Brock.** Este método define una secuencia de tareas técnicas en el cual el análisis conduce ineludiblemente al diseño.
4. **Método de Booch.** Para Booch el diseño es el proceso de determinar una implementación efectiva y eficiente que realice las funciones y tenga la información del análisis de dominio. Las siguientes actividades se plantean en esta etapa:
  1. Determinar la arquitectura inicial. Son decisiones acerca de recursos de implementación, categorías y prototipos a desarrollar
  2. Determinar el diseño lógico. Es el detalle al diagrama de clases
  3. Implementación física. Interfaz a dispositivos o características propias de la implementación
  4. Refinamiento del diseño. Es incorporar el aprendizaje debido a los prototipos y cumplir con requerimientos de desempeño.

El método de Booch abarca un proceso de micro-desarrollo y un proceso de macro-desarrollo. El nivel de micro-desarrollo define un conjunto de tareas de diseño que se reaplican en cada etapa del proceso de macro-desarrollo. Debido a esto, se mantiene un enfoque evolutivo. El proceso de micro-desarrollo es el siguiente:

- Planificación arquitectónica:
  - Agrupar objetos similares en particiones arquitectónicas separadas.
  - Distribuir objetos en capas por niveles de abstracción.
  - Identificar escenarios relevantes.

---

<sup>26</sup> Ingeniería del Software Orientada a Objetos

- Crear un prototipo de diseño.
- Validar el prototipo de diseño aplicándolo en escenarios de uso.
- **Diseño táctico:**
  - Definir políticas independientes del dominio, las reglas que gobiernan el uso de operaciones y atributos.
  - Definir políticas específicas al dominio para la administración de memoria, la gestión de errores y otras funciones de la infraestructura.
  - Desarrollar un escenario que describa la semántica de cada política.
  - Crear un prototipo para cada política.
  - Instrumentar y refinar el prototipo.
  - Revisar cada política para asegurar que transmite su visión arquitectónica.
- **Planificación de la realización:**
  - Organizar escenarios desarrollados durante el AOO por prioridad.
  - Asignar las correspondientes realizaciones arquitectónicas a los escenarios.
  - Diseñar y construir cada realización arquitectónica de manera incremental. Ajustar los objetivos y el plan de la realización incremental como se requiera.

En esta etapa debe entregarse la siguiente documentación:

- Descripción de arquitectura. Describe las decisiones más importantes de diseño como son el conjunto de procesos, manejadores de bases de datos, sistemas operativos, lenguajes, etc.
- Descripciones de prototipo. Describen las metas y contenido de las implementaciones sucesivas de prototipos, su proceso de desarrollo y la forma de probar requerimientos.
- Diagramas de Categorías.
- Diagramas de clases en diseño. Detallan las abstracciones de análisis con características de implementación.
- Diagramas de objetos en diseño. Muestran las operaciones necesarias para desarrollar una operación.
- Nuevas especificaciones.
- Especificaciones de clases corregidas. Muestra la especificación completa de los métodos con algoritmos complicados, la implementación de relaciones y el tipo de atributos.

#### **2.2.6.1.2. Descripción de Subsistemas**

El diseño del sistema se realiza a través de la descripción de los subsistemas necesarios para implementar los requisitos del usuario y el entorno de soporte requerido para su realización. Durante el diseño de los subsistemas es necesario definir cuatro componentes del diseño:

1. Dominio del problema. Los subsistemas responsables de la implementación de los requisitos del cliente directamente.
2. Interacción humana. Los subsistema, que implementan la interfaz de usuario (esto incluye subsistemas reutilizables de Interfaz Gráfica Usuario).
3. Gestión de tareas. Los subsistemas responsables de control y coordinación de tareas concurrentes que pueden empaquetarse dentro de uno o varios subsistemas; y
4. Gestión de datos. El subsistema que es responsable del almacenamiento y recuperación de objetos.

Ya que se han definido los subsistemas y comienza el diseño de cada uno de los componentes mencionados, se inicia con el diseño de objetos. Los elementos del modelo CRC se llevan a cabo en el diseño

Se divide el modelo de análisis para definir colecciones cohesivas de clases, relaciones y comportamientos. Estos elementos del diseño se empaquetan como un subsistema, generalmente los elementos del subsistema comparten alguna propiedad en común. Los subsistemas, se caracterizan por sus responsabilidades; esto es, un subsistema puede identificarse por los servicios que realiza. Al usarse dentro del contexto del diseño de un sistema OO, un servicio es una colección de operaciones que realizan una función específica. Al definir y diseñar subsistemas, éstos deben cumplir con los siguientes criterios de diseño:

- El subsistema debe poseer una interfaz bien definida a través de la cual ocurre toda la comunicación con el resto del sistema.
- Las clases dentro de un subsistema deben colaborar únicamente con otras clases dentro del subsistema a excepción de un pequeño número de clases de comunicación.
- El número de subsistemas debe mantenerse pequeño.
- Los subsistemas pueden dividirse internamente para ayudar a reducir la complejidad.

### **Características a Destacar en la Descripción de Subsistemas**

Hay ciertos puntos que deben observarse al tratar de hacer los subsistemas y éstas son el objeto de los siguientes párrafos.

Concurrencia. Si los objetos o subsistemas deben actuar sobre eventos de forma asíncrona y al mismo tiempo, se toman como concurrentes. Las tareas concurrentes se definen a través el examen del diagrama de estados para cada objeto.

Gestión de tareas. Coad y Yourdon sugieren una estrategia para el diseño de los objetos que manejan tareas concurrentes:

- Se determinan las características de la tarea.
- Se definen una tarea coordinadora y los objetos asociados.
- El coordinador y las otras tareas se integran.

Las características de una tarea se determinan a través de la comprensión del cómo se inicia la tarea. Se debe determinar también la prioridad y sentido crítico de la tarea. Tareas con una alta prioridad deben tener un acceso inmediato a los recursos del sistema. Aquellas con un alto grado de sentido crítico deben continuar su operación aún cuando la disponibilidad de recursos está reducida o el sistema esté operando con cierto nivel de degradación. Se definen los atributos y operaciones de los objetos requeridos para lograr comunicación y coordinación con otras tareas. La plantilla básica de la tarea es:

- Nombre de tarea. El nombre del objeto.
- Descripción. Una descripción narrativa acerca del propósito del objeto.
- Prioridad. Prioridad de la tarea.
- Servicios. Una lista de operaciones que constituyen las responsabilidades del objeto.
- Coordinado por: la manera según la cual se invoca el comportamiento del objeto.
- Comunicar vía. Valores de datos de entrada y salida relevantes a la tarea.

Gestión de datos. La gestión de datos abarca dos áreas distintas:

1. La gestión de datos críticos para la propia aplicación
2. La creación de una infraestructura para el almacenamiento y recuperación de objetos.

En general, la gestión de datos se diseña por capas. La idea es aislar los requisitos de bajo nivel para la gestión de datos de los de alto nivel para la gestión de los atributos del sistema.

El diseño del componente para la gestión de datos incluye el diseño de los atributos y operaciones necesarias para la gestión de los datos. Los atributos relevantes se añaden a cada objeto en el dominio del problema y brindan servicios de:

- a) Informar a cada objeto que se almacene.
- b) Recuperar objetos almacenados para ser usados por otros componentes del diseño.

En este punto es donde se decide usar o no una base de datos, o si se pueda llevar a cabo el almacenamiento en archivos planos, o la opción que mejor le convenga para los intereses del sistema.

Interfaz hombre-máquina. Una vez que se han definido el actor y su escenario de uso, a través de los casos de uso, se identifica una jerarquía de órdenes. La jerarquía de órdenes define las principales categorías de menús del sistema y debe ser refinada repetidamente hasta que cada caso de uso pueda implementarse navegando a través de la jerarquía de funciones. Se debe realizar la detección de clases reusables, sólo se necesita identificar los objetos que poseen las características apropiadas para el dominio de problema.

Gestión de recursos. Se debe diseñar un mecanismo de control sobre una variedad de recursos diferentes disponibles para un sistema o producto OO, ya que los subsistemas compiten por estos recursos al mismo tiempo.

Comunicación entre subsistemas. Una vez que se ha especificado cada subsistema, es necesario definir las colaboraciones que existen entre ellos. La comunicación puede ocurrir al establecer un enlace cliente servidor. Se debe especificar qué operaciones o intercambios hay entre los subsistemas. Un contrato brinda indicaciones acerca de las maneras en que un subsistema puede interactuar con otro y los siguientes pasos del diseño pueden aplicarse para especificar un contrato para un subsistema:

1. Listar las solicitudes que puedan realizar los colaboradores del subsistema, organizarlas según el subsistema y definir las dentro del marco de uno o más contratos apropiados. Se deben contemplar los contratos heredados desde las superclases.
2. Para cada contrato se destacan las operaciones heredadas y privadas, requeridas para implementar las responsabilidades que implica el contrato. Se deben asociar las operaciones con clases, específicas que residan dentro del subsistema.
3. Se debe considerar un contrato a la vez y crear una tabla con los siguientes datos:
  - Contrato.
  - Tipo. Determinar si el tipo del contrato es cliente servidor, punto a punto, etcétera.
  - Colaboradores. Los nombres de los subsistemas que son partes del contrato.
  - Clases. Los nombres de las clases contenidas dentro del subsistema que soportan servicios implicados por el contrato.
  - Operaciones. Los nombres de las operaciones dentro de la clase que implican los servicios.
  - Formato del mensaje. El formato de mensaje requerido para implementar la interacción entre colaboradores, es una descripción apropiada del mensaje para cada interacción entre subsistemas.
  - Si los modos de interacción entre los subsistemas son complejos, se creará un grafo de interacción entre subsistemas, se representa cada subsistema y las interacciones con otros subsistemas. Los contratos que se invocan durante una interacción se señalan de la forma mostrada por la figura.

### **2.2.6.1.3. Diseño de Objetos**

Se debe desarrollar un diseño detallado de los atributos y operaciones que incluye cada clase, y una especificación completa de los mensajes que conectan la clase con sus colaboradores.

Descripciones de objeto. Una descripción de un objeto, una instancia de una clase o subclase, puede tomar una de estas dos formas:

1. Una descripción del protocolo que establece la interfaz de un objeto definiendo cada mensaje que el objeto puede recibir y la correspondiente operación que el mismo ejecuta al recibir el mensaje. Es un conjunto de mensajes y del comentario correspondiente para cada uno de ellos.
2. Una descripción de la implementación que muestra detalles de ella para cada operación implicada por un mensaje que se pasa al objeto. Los detalles de implementación incluyen información acerca de la parte privada del objeto, esto es, detalles internos acerca de las estructuras de datos que describen los atributos del objeto y detalles procedimentales que describen las operaciones.

Para un gran sistema con muchos mensajes, es posible a menudo crear categorías de mensajes. Una descripción de la implementación de un objeto aporta los detalles internos ocultos necesarios para la implementación pero que no son necesarios para su invocación. Una descripción consta de la siguiente información:

- Especificación del nombre del objeto y referencia a su clase.
- Especificación de las estructuras de datos privadas indicando los elementos de datos y tipos.
- Descripción procedimental de cada operación o alternativamente, punteros a tales descripciones procedimentales.

Diseño de algoritmos y estructuras de datos. Los algoritmos y estructuras de datos se diseñan creando una notación para implementar la especificación de cada operación. A veces los algoritmos son solo secuencias de comando para cumplir ciertas operaciones, pero otras veces, si la operación es compleja, pudiera ser necesaria la modularización de la operación. Las operaciones pueden dividirse en tres amplias categorías:

1. Operaciones que manipulan datos de alguna manera.
2. Operaciones que realizan cálculos.
3. Operaciones que monitorizan un objeto debido a la ocurrencia de un evento controlador.

Los verbos se asocian a acciones u ocurrencias. En el contexto de la formalización del diseño de objetos, se deben considerar verbos y frases verbales descriptivas como operaciones potenciales. Después de haber hecho lo anterior sigue la optimización y hay tres etapas para la optimización del DOO:

- Asegurarse que el diseño lleva a una utilización eficiente de los recursos y a una facilidad de implementación.
- Revisar las estructuras de datos atributos y las operaciones para aumentar el rendimiento.
- Crear nuevos atributos para preservar información derivada, evitando la repetición de computaciones previas.



Componentes de programas e interfaces. Ahora se deben identificar las interfaces que existen entre objetos y la estructura general de los objetos. Se deberá representar en el contexto del lenguaje de programación con el cual se debe implementar el diseño. El primer paso es definir las interfaces para cada una de las operaciones asociadas y el siguiente requiere un refinamiento paso a paso para cada operación asociada al paquete.

#### **2.2.6.1.4. Patrones de Diseño**

La tarea a desempeñar es detectar patrones que caractericen un problema, y los patrones correspondientes que puedan combinarse para crear una solución. Los patrones resuelven problemas de diseño específicos y hacen el diseño orientado a objetos más flexible, elegante, y a reutilizar patrones de diseño existentes y crear nuevos si no se puede aplicar la reusabilidad.

Descripción de un patrón de diseño. Los patrones de diseño pueden describirse a través de:

1. Nombre del patrón. Es una abstracción que aporta un significado acerca de su aplicabilidad y objetivos.
2. Problema al cual se aplica generalmente el patrón. Indica el entorno y las condiciones que deben existir para hacer aplicable el patrón de diseño.
3. Características del patrón de diseño. Muestran los atributos del diseño que deben ajustarse para permitirle al patrón acomodarse a una variedad de problemas. Representan características del diseño según las cuales puede buscarse el patrón apropiado.
4. Consecuencias de la aplicación del patrón de diseño. Aportan una indicación de las ramificaciones de las decisiones de diseño.

Uso de patrones en el diseño. Los patrones de diseño pueden usarse aplicando:

- Herencia. A través del uso de la herencia un patrón de diseño existente se convierte en una plantilla para una subclase nueva. Los atributos y operaciones que existen en el patrón pasan a ser parte de la clase.
- Composición. Un objeto complejo puede ensamblarse a partir de la selección de un conjunto de patrones de diseño y la composición del objeto seleccionado. Cada patrón de diseño se trata como una caja negra y la comunicación entre ellos ocurre solamente a través de interfaces bien definidas. La composición usa patrones de diseño existentes, componentes reusables, de forma inalterada

### **2.2.7. Programación OO**

Cualquier lenguaje que no es natural, del hablar del hombre, es decir, artificial, puede utilizarse para definir una secuencia de instrucciones para el procesamiento por una computadora, es decir, la traducción de las instrucciones comprensibles al humano a un código que comprende la computadora es a lo que se le llama programación.

Los lenguajes de programación son sumamente importantes, ya que con ellos es posible integrar una aplicación por completo, permiten hacer conexiones, traer o llevar datos, hacer consultas, operaciones, y muchas, muchas otras cosas más. Hay varios tipos de programación, pero para los propósitos de esta tesis, solo se hará referencia a lo que es la Programación Orientada a Objetos (POO).

POO<sup>27</sup> es un estilo de programación en el que un programa se contempla como un conjunto de objetos limitados que, a su vez, son colecciones independientes de estructuras de datos y rutinas que interactúan con otros objetos. Una clase define las estructuras de datos y rutinas de un objeto. Un objeto es una instancia de una clase, que se puede usar como una variable en un programa. En algunos lenguajes orientados a objetos, éste responde a mensajes, que son el principal medio de comunicación, en otros lenguajes orientados a objeto se conserva el mecanismo tradicional de llamadas a procedimientos.

### **2.2.7.1. Estándares de codificación**

Hay actividades que se realizarán posteriormente a la creación del código, como son las revisiones de código, evaluación, consultas, lectura, reemplazar, modificar o reutilizar el código, por ello es importante crear, desarrollar y seguir estándares de programación.

Por todo lo anterior es importante que se utilicen estándares en el desarrollo y documentación del sistema, con la finalidad de disminuir el tiempo de análisis en estas etapas. Es necesario que el código y la documentación asociada al sistema resulten claros para cualquier persona que las lea. Estos estándares deben conocerse antes de comenzar el desarrollo del sistema y sirven para:

- Organizar los pensamientos del programador
- Evitar errores.
- Determinar la forma de generar la documentación del código
- Para que lo que se pretende estandarizar resulte claro y fácil de leer
- Suspender el trabajo y regresar a él sin perder la pista acerca de lo ya realizado.
- Ayudar en la localización de errores
- La realización de cambios
- Clarifica qué secciones de un programa realizan determinada función.
- Mantener la correspondencia entre componentes de diseño y componentes de código.
- Los cambios en el diseño serán fáciles de implementar en el nivel de código.
- Que las modificaciones al código sean directas y la posibilidad de error se disminuya
- Permitir que otros puedan comprender que hace y como opera, con facilidad.

Se sugiere que cada librería, componente o archivo del sistema tenga un encabezado en el cual se muestre la siguiente información:

- Archivo
- Versión
- Fecha

---

<sup>27</sup>Programación orientada a objetos. *Enciclopedia® Microsoft® Encarta 2001*. © 1993-2000.

- Autor
- Descripción

### 2.2.7.2. Modelos para la programación

Cuando se desarrolla un sistema se suele implementar en el lenguaje que el desarrollador domina, el que la empresa utiliza o simplemente, el que permite resolver de manera mas eficiente el problema. Cada componente de un programa implica al menos tres aspectos principales:

- Estructuras de control. Algunos modelos y estándares sugieren que el código debe estar escrito de manera tal que un componente pueda leerse fácilmente desde lo general a lo particular. Una buena reestructuración puede mejorar la comprensión.
- Algoritmos. Regularmente el desarrollador es, de acuerdo a su experiencia, el encargado de decidir el algoritmo a emplear en cada parte del desarrollo. Por lo que, para otra persona puede resultar difícil comprenderlo. Uno de los aspectos que requiere mayor cuidado es el rendimiento en la implementación. La experiencia del programador puede hacer que el código corra tan rápido como sea posible o consuma mas recursos del sistema.
- Estructuras de datos. Al escribir programas se debería fijar el formato y almacenar datos de modo que la gestión y manipulación de datos sea más directa. Existen varias técnicas que utilizan la estructura de datos para indicar como debería organizarse el programa.
  - Mantener la simplicidad del programa.
  - Localización de entrada y salida.
  - Revisión y reescritura.
  - Reutilización. Productiva y consumidora
  - Documentación interna
    - Encabezado
    - Comentarios del programa
    - Nombres de variables
    - Dar formato para mejorar la comprensión
    - Documentación de los datos

### 2.2.8. Pruebas del Software OO

EL objetivo de la realización de Pruebas Orientadas a Objetos (POO) es encontrar el mayor número posible de errores con una cantidad de esfuerzo racional a lo largo de un espacio de tiempo realista. Hay tres puntos que deben hacerse para probar adecuadamente los sistemas OO, las cuales son:

- a) La definición de las pruebas debe ampliarse para incluir técnicas de detección de errores aplicados a los modelos de DOO y AOO.
- b) Estrategia para las pruebas de unidad e integración deben cambiar significativamente.

- c) El diseño de casos de prueba debe tener en cuenta las características propias del software orientado a objetos.

### 2.2.8.1. Aspectos Importantes

Es muy importante considerar el encapsulamiento y la herencia al desarrollar pruebas, puesto que de no hacerlo puede verse afectado el rendimiento de la aplicación, al no detectar errores por no diseñar casos de prueba que consideren estos puntos, se recomienda ver la sección 2.2.1. que maneja estos conceptos.

Las mejores pruebas se obtienen cuando el diseñador observa al sistema de manera nueva o no convencional. Cualquiera que sea el estilo de la interfaz, en el diseño de casos de prueba que ejerciten la estructura superficial se deberán usar objetos y operaciones como guías que lleven a tareas olvidadas.

La estructura profunda son los detalles técnicos internos de un programa OO. Esto es, la estructura que aparece a través del examen del diseño y/o el código. La prueba de la estructura en profundidad se diseña para ejercitar dependencias, comportamientos, y mecanismos de comunicación, que se han establecido como parte del diseño de subsistemas y objetos del software OO. Los modelos de análisis y diseño se usan como base para la prueba de la estructura profunda.

### 2.2.8.2. Modelos de Pruebas de AOO Y DOO

Los modelos de análisis y diseño OO no pueden probarse convencionalmente, pues no pueden ejecutarse, pero sí se puede examinar la corrección (exactitud) y consistencia, de esto se hablará en párrafos subsecuentes.

- **Corrección.** La corrección sintáctica se juzga según el uso apropiado de la simbología, y cada modelo se revisa para asegurar que se han mantenido la notación y sintaxis del método específico de análisis y diseño elegidos. Se debe juzgar la corrección del modelo con el dominio del problema del mundo real, si el modelo refleja el mundo real de a un nivel de detalle apropiado, entonces está semánticamente correcto.
- **Consistencia.** La consistencia de los modelos de AOO y DOO pueden juzgarse a través de una consideración de las relaciones entre entidades en el modelo. Para valorar la consistencia, deberán examinarse cada clase y sus conexiones a otras clases. Para ello puede usarse el modelo clase-responsabilidad-colaboración<sup>28</sup> (CRC) y un diagrama de objeto-relación<sup>29</sup>.

### 2.2.8.3. Diseño de Casos de Prueba

Berard<sup>14</sup> ha definido un método de diseño de casos de prueba muy útil, el cual se explica a continuación:

1. Cada paso de prueba debe estar identificado unívocamente y asociado explícitamente con la clase a probar.

---

<sup>28</sup> El modelo CRC se compone de tarjetas índice CRC, cada tarjeta lista el nombre de la clase, sus responsabilidades (operaciones) y colaboradores (otras clases a las que se envían mensajes y de las que depende para el cumplimiento de sus responsabilidades). Las colaboraciones implican una serie de relaciones entre clases del sistema OO.

<sup>29</sup> Aporta una representación gráfica de las conexiones entre clases.

2. Debe establecerse el propósito de la prueba.
3. Desarrollar una lista de pasos de prueba para cada prueba, la cual deberá contener lo siguiente:
  - a) Una lista de estados especificados para el objeto a probar.
  - b) Una lista de mensajes y operaciones a ejercitar como consecuencia de la prueba.
  - c) Una lista de excepciones que pueden ocurrir al probar el objeto.
  - d) Una lista de condiciones externas.
  - e) Información suplementaria que ayudará en la implementación de la prueba.

#### **2.2.8.4. Tipos de Pruebas OO**

La estrategia clásica para ejecutar pruebas comienza con la prueba a pequeña escala (prueba de unidad) y trabaja moviéndose hacia la prueba a gran escala (validación y prueba del sistema). Estas se explican a continuación:

- Prueba de unidad. La prueba de clases para software OO está dirigida por las operaciones encapsuladas por la clase y el estado del comportamiento de la clase. Esta prueba se centra en la clase, ya que es la menor unidad de programa compilable, puede contener operaciones, y, a su vez, una operación puede existir como parte de un número de clases diferentes, ya que las clases han sido probadas individualmente, se integran y se ejecuta una serie de pruebas de regresión para descubrir errores ocasionados al integrarse y posibles efectos colaterales causados por añadir nuevos módulos, finalmente, el sistema se prueba globalmente para asegurar que se descubren los errores en los requisitos.
- Prueba de integración. Buscan errores posibles en comunicación de mensajes. Para determinar errores posibles al invocar funciones (operaciones), debe examinarse el comportamiento de la operación. Las técnicas de integración son aplicables tanto a atributos como a operaciones. Existen dos estrategias diferentes para pruebas de integración en sistemas OO<sup>30</sup>, las cuales son las siguientes:
  - a) Pruebas basadas en hilos. Integra el conjunto de clases necesario para responder a una entrada o evento del sistema. Cada hilo se integra y prueba individualmente. Se aplica la prueba de regresión para asegurar que no ocurren efectos colaterales.
  - b) Pruebas basadas en uso. Comienza la construcción del sistema probando las clases independientes que usan muy pocas o ninguna de las clases servidor, después se comprueban las clases dependientes, que usan las clases independientes, esto continúa hasta construir el sistema por completo.
  - c) Prueba de agrupamiento. Se ejercita un conjunto de clases colaboradoras determinadas a través del examen de los modelos CRC y objeto relación a través del diseño de casos de prueba que intentan descubrir errores en las colaboraciones.

---

<sup>30</sup> Brinder, R. V., Object Oriented Software Testing, Communications of the ACM, Septiembre 1994.

- Prueba de validación. La validación del software orientado a objetos se centra en las acciones visibles del usuario y las salidas del sistema reconocibles por éste. Para llevar a cabo pruebas de validación se debe basar en los casos de uso<sup>31</sup> que forman parte del modelo de análisis.
- Pruebas convencionales. Hay también pruebas usadas convencionalmente que pueden usarse para evaluar el software OO, las cuales son:
  - Prueba de bucles. Los bucles son muy usados en la mayoría de los algoritmos implementados, es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. Hay cuatro clases diferentes de bucles:
    1. Simples
    2. Concatenados
    3. Anidados
    4. No estructurados.
  - Prueba de Caja Negra. Permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra ignora intencionadamente la estructura de control, centra su concentración en el campo de la información. Hay varios métodos de prueba de caja negra, los cuales son:
    1. Métodos de prueba basados en grafos
    2. Partición equivalente
    3. Análisis de valores límite
    4. prueba de comparación
- Pruebas basadas en fallos. El objetivo de la prueba basada en fallos dentro de sistemas OO es el diseñar pruebas que posean una alta probabilidad en la detección de errores posibles, lo cual comienza con el modelo de análisis, buscando la aparición de errores posibles. Se diseñan casos de prueba para ejercitar el diseño o el código y así determinar si hay fallos.
- Pruebas basadas en escenarios. Las pruebas basadas en escenarios se concentran en lo que el usuario hace, no en lo que hace el producto. Esto significa capturar las tareas (a través de casos de uso) que el usuario debe realizar, y después aplicarlas a sus variantes como pruebas. Los escenarios descubren errores de interacción y tienden a utilizar varios subsistemas en una prueba simple. Los resultados de pruebas basadas en escenarios detectan: 1. Especificaciones incorrectas y 2. Interacciones entre subsistemas.
- Pruebas aplicables a clases. La realización de pruebas para sistemas OO se centran en una clase simple y los métodos encapsulados por ella. Las pruebas aleatorias y el particionamiento son métodos que pueden usarse para ejercitar una clase durante las pruebas OO. Las siguientes son pruebas para aplicar directamente a las clases:

---

<sup>31</sup> El caso de uso brinda un escenario que posee una alta probabilidad con errores encubiertos en los requisitos de interacción del cliente.

- Pruebas aleatorias. El mínimo comportamiento de la historia de vida de una instancia incluye varias operaciones, las cuales representan la secuencia mínima de prueba. Sin embargo, pudieran ocurrir una amplia variedad de comportamientos dentro de una secuencia. De esto puede generarse aleatoriamente una amplia variedad de secuencias de operaciones. Las pruebas generadas se ejecutarán junto con otras pruebas de orden aleatorio para ejercitar las historias de vida de diferentes instancias de clase.
- Pruebas de partición al nivel de clase. Las pruebas de partición reducen el número de casos de prueba necesarios para ejercitar la clase. Las entradas y salidas están categorizadas, y los casos de prueba están diseñados para ejercitar cada categoría. Las particiones basadas en estados categorizan las operaciones de clase basándose en su habilidad para cambiar el estado de la clase. Las pruebas se diseñan de manera tal que se ejercitan por separado las operaciones que cambian de estado y las que no lo hacen. La partición basada en atributos categoriza las operaciones de clase basándose en los atributos que usan. Se diseñan entonces secuencias de pruebas para cada partición. La partición basada en categorías categoriza las operaciones de clase basándose en las funciones genéricas que cada una realiza.

#### **2.2.8.5. Diseño de Casos de Prueba Interclases**

Es en esta etapa cuando deben comenzar las pruebas de las colaboraciones entre clases. Las pruebas de colaboraciones entre clases pueden realizarse a través de la aplicación de métodos aleatorios y de particionamiento, así como pruebas basadas en escenarios y pruebas de comportamiento.

- Pruebas de clases múltiples. Kirani y Tsai<sup>32</sup> sugieren una serie de pasos para hacer casos de pruebas aleatorios:
  1. Para cada clase cliente, usar la lista de operadores de clase para generar una serie de secuencias de prueba aleatorias. Los operadores enviarán mensajes a otras clases servidores.
  2. Para cada mensaje que se genera, determinar la clase colaboradora y el operador correspondiente en el objeto servidor.
  3. Para cada operador en el objeto servidor (que haya sido invocado por los mensajes enviados desde el objeto cliente), determinar los mensajes que este transmite.
  4. Para cada uno de los mensajes, determinar el próximo nivel de operadores que se invocan e incorporarlos en la secuencia de prueba.

---

<sup>32</sup> Kirani, S. y W. T. Tsai. Specification and Verification of Object Oriented Programs. Computer Science Department, University of Minnesota. Diciembre 1994.

- Pruebas derivadas de modelos de comportamiento. Se debe derivar una secuencia de pruebas que ejercitan el comportamiento dinámico de la clase y aquellas clases que colaboran con ella. Para llevar esto a cabo se vale del Diagrama de Transición de Estados (DTE) para registrar el flujo de comportamiento del sistema, ya sea uno o varios. Las pruebas a diseñar deberán ser capaces de abarcar todos los estados, esto es, las secuencias de operaciones deberán provocar que las clases ejecuten transiciones a través de todos los estados permitidos. Se podrán derivar aún más casos de prueba para asegurar que se han ejercitado de manera adecuada todos los comportamientos de la clase.



# PARTE II. SEGURIDAD

---

---

## CAPÍTULO 3. IMPLICACIONES DE SEGURIDAD

---

---

### 3.1. Conceptos Básicos

La Seguridad Informática surge para cubrir la necesidad básica de la protección de las ideas. Para ello se puede hacer una restricción en los individuos que tienen acceso a la información, otra opción es codificar, de alguna manera disfrazando, u ocultando la información.

Con el tiempo cambió esta situación, la forma en cómo comunicar las ideas se transformó, la comunicación oral, pasó a ser escrita, de ahí, transmitida en libros, periódicos, etcétera, luego fue codificada, para la protección de ideas.

Se sabe que la información contenida en una computadora no está protegida, puesto que si la máquina sufre algún daño se perdería toda la información, es por ello que se necesita recurrir a otras tácticas para lograr la conservación de la información, es de aquí de donde surge la necesidad de proteger la información y los medios que la contienen, transmiten o colaboran con ella de alguna manera. Ahora es necesario definir puntos más concretos para lograr la comprensión total del tema. Pero para poder comprender bien este tema, se deben definir los siguientes conceptos:

**Informática** es la disciplina encargada del tratamiento o manejo sistematizado de la información con el fin de ayudar a la toma de decisiones, valiéndose primordialmente de medios computarizados. Conjunto de conocimientos científicos y de técnicas que hacen posible el tratamiento automático y racional de la información considerada como soporte de los conocimientos y las comunicaciones. Combina los aspectos teóricos y prácticos de la ingeniería, electrónica, teoría de la información, matemáticas, lógica y comportamiento humano.

**Información** es el conjunto de datos ordenados y evaluados en la perspectiva necesaria, de tal manera que proporcionen conocimiento relevante en una unidad de tiempo determinada. Es un conjunto de conocimientos basados en los datos que mediante un procesamiento, se les da significado, propósito y utilidad.

**Seguridad.** Proviene del vocablo latino "*securitis*", el cual quiere decir ausencia o idea de que no hay peligro. Garantía, tranquilidad, protección, certidumbre, confianza.

**Seguridad Informática.** Es el conjunto de elementos que garantizan el comportamiento de un sistema de información según lo esperado por un usuario. Conjunto de técnicas desarrolladas para proteger los equipos informáticos individuales y conectados en una red frente a daños accidentales o intencionados, como pueden ser mal funcionamiento del hardware, la pérdida física de datos, el acceso a bases de datos por personas no autorizadas, fallo en la energía, virus, etcétera.

**Sistema.** Conjunto de elementos que interactúan entre sí para el logro de un objetivo común.

**Comunicar.** Transmitir una idea al grado que puedo afectar el conocimiento de una persona.

**Comprometer.** Hacer que el sistema se vea comprometido es verlo amenazado, que tenga o pueda tener pérdida, alteración, modificación, robo, interceptación o interrupción. Cualquier debilidad que pueda explotarse.

**Vulnerabilidad.** La vulnerabilidad en un sistema es cualquier debilidad que pueda explotarse, o en su caso el punto más débil que pueda ser detectado.

**Amenazas.** Quebrantar la seguridad de mi sistema. Cualquier circunstancia con el poder suficiente para lograr tener pérdida, alteración, modificación, robo, interceptación o interrupción

**Ataque.** Todo aquél acto deliberado que con él lleva el objeto de evadir o violar las políticas de la seguridad informática o de un sistema. Pueden darse en dos formas: Activo, es donde se alteran los recursos o la operación del sistema, y Pasivo sólo se conoce y / o se hace uso de la información.

La Seguridad en Informática (SI) se ha convertido en parte fundamental del mundo computacional ya que cada vez se propaga más la delincuencia, también día con día más gente se incorpora a la red mundial, internet, este medio conlleva una serie riesgos inherentes, y a través de ello se abre una puerta de entrada para las personas que están buscando atacar, ya sean "hackers" o "crackers"<sup>33</sup>.

Todas estas circunstancias hacen necesario que se tomar medidas para brindar seguridad a los usuarios de esta herramienta y para proteger sistemas y redes de datos.

También es de suma importancia estar informados de como actúan los delincuentes cibernéticos y de tratar de prevenir los movimientos o acciones que éstos puedan tomar para realizar sus ataques. Debido a que las computadoras no están seguras en su ambiente, resulta evidente que la única forma de garantizar la integridad física de los datos es mediante copias de seguridad, comúnmente llamados respaldos.

Un sistema de cómputo es seguro si se puede confiar en él, si su "software" se comporta como se espera que lo haga, y la información almacenada en él se mantiene inalterada y accesible durante tanto tiempo como su dueño lo desee, dicho esto en términos de Ingeniería de Software, es la habilidad de un sistema para protegerse así mismo contra intrusiones accidentales o deliberadas.

Es importante saber que no se va a poder proteger la información al 100%, para ejemplificar esta situación consideremos la frase de Gene Spafford:

*"El único sistema seguro es uno que está apagado, desconectado, guardado en una caja fuerte de titanio, encerrado en un bunker de concreto, rodeado por gas venenoso y guardias armados muy bien pagados. Aun así, no apostaría mi vida por él."*

---

<sup>33</sup> Aficionado a las computadoras, la programación y la tecnología. Denomina a quien trata de invadir en secreto computadoras, y consultar o alterar los programas o los datos almacenados en las mismas, disfruta desmenuzando sistemas operativos y programas para ver cómo funcionan. Los crackers hacen casi lo mismo, con la diferencia que su objetivo es dañar.

## 3.2. Principios de seguridad

La **misión de la SI** es evitar cualquier forma posible pérdida o daño al sistema de información, evitar que el sistema de información se comporte de manera inesperada para el usuario.

Las **Principales Preocupaciones de la SI** son:

- Que el comportamiento del sistema sea según lo esperado por el usuario.
- Resguardo de material específico, así como de los recursos.
- Material no crítico, pero importante o útil.
- Material clasificado.
- Material sensible.
- Resguardo de información
- Resguardo de instalaciones
- Resguardo de los equipos
- Resguardo de los medios de almacenamiento
- Resguardo de respaldos
- Resguardo del personal

Existen varias **Razones Para** querer **Irrumpir la Seguridad** de un sistema, las principales son:

- Espionaje comercial
  - Inteligencia (espionaje)
  - Vandalismo
  - Terrorismo
  - Curiosidad
- } Medios por los que se vulnera o daña un sistema

**Atributos de un Sistema Seguro.** Se dice que un sistema debe contar con los siguientes puntos para poder ser seguro:

- **Confidencialidad.** Que sólo conozcan la información los que tienen derecho a ella. Un sistema de cómputo no debe permitir que la información contenida en él sea accesible a nadie que no tenga la autorización adecuada.
- **Integridad.** La información no se altera sin autorización. Un sistema de cómputo no debe permitir modificaciones no autorizadas a los datos o a la información contenida en él. Esto comprende cualquier tipo de modificaciones:
  - Por errores de hardware y/o "software".
  - Causados por alguna persona de forma intencional.
  - Causados por alguna persona de forma accidental.

- **Autenticidad.** Que la información venga de las fuentes autorizadas o fidedignas. Se maneja en cuestión de telecomunicaciones, y se refiere a contar con un medio para verificar quién envía la información, así como poder comprobar que los datos no fueron modificados durante su transferencia.
- **Disponibilidad.** Que los usuarios legítimos puedan usar la información cuando la requieran. Los usuarios podrán utilizar los recursos en el momento en que los necesiten. También significa que el sistema sea capaz de recuperarse rápidamente en caso de ocurrir un problema de cualquier especie.

### 3.3. Objetivos de Seguridad

Por su parte la Organización Internacional de Normalización sugiere una serie de objetivos de seguridad, los cuales son:

- **Confidencialidad.** Garantiza que la información sólo pueda ser accesada por los usuarios autorizados.
- **Autenticidad.** Demostrar ser quien digo ser por:
  - Algo que se sabe (contraseña).
  - Algo que se tenga identificado (identificación personal).
  - Algo que se es (ADN).

La autenticidad se puede dar de las siguientes maneras:

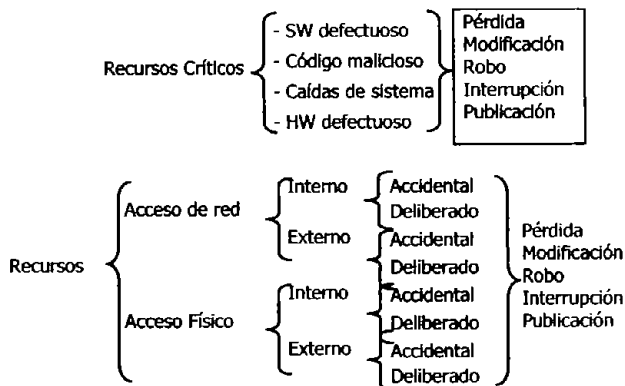
- **Directa:** sólo intervienen las partes autorizadas (banco –cajera- usuario).
- **Indirecta:** participa un intermediario confiable que funge como juez.
- **Unilateral:** sólo una de las partes se autentica (MSN usuario se identifica).
- **Mutua:** ambos se autentican (e-comercio, banca electrónica).
- **Integridad.** Proteger los activos del sistema contra publicación, robo, alteración, destrucción, etc.
- **Control de acceso.** Nos permite proteger los activos de sistema contra accesos no autorizados. No usa técnicas criptográficas.
- **No repudio.** Identificar todas las actividades que hubo en el momento de la transacción. Protege de que alguna de las partes niegue haber recibido un mensaje. Los mecanismos son:
  - Acuse de recibo.
  - Acuse de salida (firmas digitales, cifrado, llave pública o secreta etcétera parte confiable).

### 3.4. Amenazas y Riesgos

**Amenazas.** Esto es, cómo puedo identificar el riesgo, para ello se deben llevar a cabo las siguientes tareas:

- 1) Identificar las amenazas.
- 2) Cuantificar las amenazas.
  - a) Costo de prevenir pérdida, modificación, robo, interrupción o publicación.
  - b) Costo de pérdida. Si el daño que sufrimos es irreparable, irremplazable o irreversible. Cuántas horas, días, o semanas no va a estar disponible el servicio. Tiempo = \$.
- 3) Distribución probabilística. Evaluar posibilidad de que suceda algo.
  - a) Evaluar el costo y probabilidad.
  - b) Analizar si se pueden costear mecanismos de seguridad.
- 4) Revisión constante.
  - a) Cierto tiempo.
  - b) Cambios.

**Tipos De Amenazas.** Existen diferentes tipos de amenazas para lo cual debo identificar mis recursos y en base a ello definir qué deseo o quiero proteger y cómo. Lo anterior se esquematiza en los siguientes cuadros:



**Riesgo.** Posibilidad de que ocurra algo desfavorable. Posibilidad de pérdida, modificación, robo, interrupción o publicación.

$$\text{Riesgo} = \text{Amenazas} + \text{Impacto}$$

**Disminución de Riesgos.** Debemos tomar en cuenta que la inseguridad es permanente, es decir, no se acaba, ya que no podemos controlar todos los factores que pueden afectar a nuestro sistema. Lo único que podemos hacer ante esto es disminuir riesgos. Por ello es importante:

- Aseguramos de que el costo de romper la seguridad no excede el de salvaguardar la información.

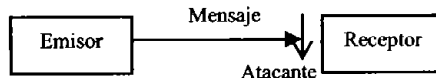
- Asegurar que el tiempo empleado para romper la seguridad exceda el tiempo de vida útil de la información.

**Posturas Ante Riesgos.** Hay diferentes estrategias que como organización se pueden tomar ante los riesgos, las cuales son:

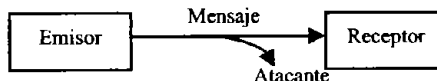
- **Paranoica.** Exagero la seguridad. Nada está permitido. Todo está cerrado. No permito nada.
- **Prudente.** Abro lo necesario, únicamente para que funcione mi sistema al 100%. "Lo que no está expresamente permitido está prohibido".
- **Permisiva.** Abro varios servicios aunque no sea necesario. "Lo que no está expresamente prohibido está permitido".
- **Promiscua.** Todo esta permitido. Abro todo.

**Tipos de afectaciones a sistemas.** En un modelo de comunicación nuestros sistemas pueden tener las siguientes afectaciones:

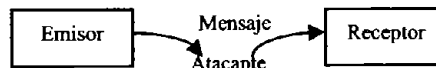
1. Interrupción. Existe intercepción del mensaje que es enviado del emisor al receptor antes de que llegue a éste último en el medio de transmisión.



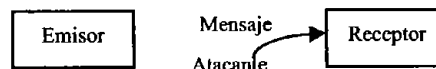
2. Intercepción. El mensaje es interceptado antes de llegar al receptor y de esta manera al igual que el receptor el atacante tendrá la información, como sigue:



3. Modificación. El mensaje es modificado por el atacante antes de llegar al receptor y de esta manera el receptor recibe información errónea.



4. Falsificación o suplantación. El mensaje no es enviado por el emisor sino por el atacante que lo suplanta como:



### 3.4.1. Qué se debe de proteger

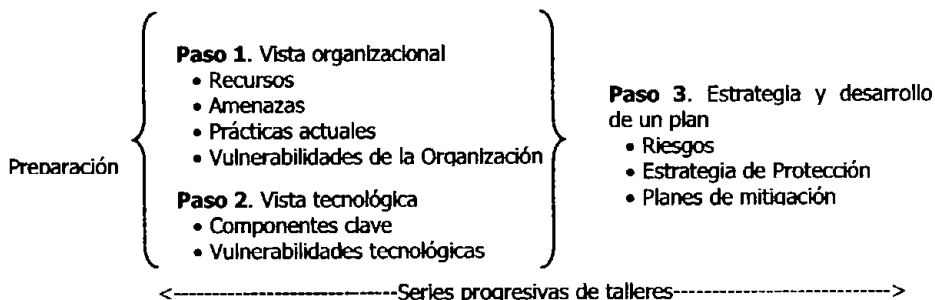
Lo que se trata de proteger son los recursos con los que contamos, a continuación se presenta un cuadro de los posibles recursos dentro de una organización:



Dentro de ellos también comprende el "software" (todos aquellos programas y aplicaciones informáticas que se ejecutan en una computadora o en sistemas), hardware (conjunto de elementos físicos, mecánicos, eléctricos y electrónicos que forman parte de una computadora o un medio para transmitir la información, como es la infraestructura de la red), recursos humanos, información, instalaciones y consumibles entre otros.

Para detectar riesgos y amenazas se recurre al método llamado OCTAVE por sus siglas en inglés Operationally Critical Threat, Asset, and Vulnerability Evaluation, desarrollado por el Software Engineering Institute de la Universidad Carnegie Mellon en Pittsburgh, EE.UU, cuya patente está registrada ante el CERT. Se traduce como Amenaza Crítica, Recursos, y Evaluación de Vulnerabilidad Operacionalmente, consta de una serie de pasos que se describen a continuación. La aplicación de sus árboles de amenazas se encuentran en el Anexo 5.

### 3.4.1.1. Proceso OCTAVE



#### Método OCTAVE

- Enfocado en organizaciones grandes
- Es un método sistemático, sensible al contexto por evaluar los riesgos
  - Unión de talleres
  - Dirigido por el equipo de análisis
- Definido por:
  - Método de aplicación guía (procedimientos, guías, catálogos de información)
  - Método entrenamiento
  - Información del manejo de riesgos de seguridad

### 3.4.1.2. OCTAVE-S Método para Pequeñas Organizaciones

Actualmente en prueba piloto, este método define un método más estructurado para evaluación de riesgos en las organizaciones pequeñas. Las características principales de éste método son:

- Requiere menos especialización de seguridad en el equipo del análisis
- Se requiere que el equipo de análisis completo, o casi completo, comprenda la organización y qué es importante.

Se definirá por:

- Procedimientos detallados para cada proceso
- Hoja de trabajo y plantillas para cada proceso
- Catálogos de información

#### PASOS

##### Paso 1. Vista Organizacional

- Recursos
- Amenazas
- Prácticas actuales
- Vulnerabilidades de la Organización
- Requisitos de Seguridad

##### *Catálogo de prácticas OCTAVE*

Conocimientos de Seguridad y entrenamiento	Estrategia de Seguridad	Administrar la Seguridad	Políticas de Seguridad y Regulaciones	Colaboración en Dirección de Seguridad	Contingencia Planeación de Recuperación de Desastres

Seguridad Física	Información de Seguridad Tecnológica	Seguridad del Personal
Planes y procedimientos de Seguridad Física Control de Acceso Físico Monitoreo y Auditoría Seguridad Física	Administración de Sistemas y Redes Herramientas de Administración de sistema Monitoreo y Auditoría de Seguridad Autenticación y Autorización Administración de vulnerabilidades Cifrado Diseño y Arquitectura de Seguridad	Manejo de Incidentes Personal general Prácticas



### ***Recursos críticos***

Los recursos más importantes para la organización son:

- Información
- Sistemas
- Servicios y aplicaciones
- Personas

Habrá un impacto adverso grande a la organización si:

- Se descubren los recursos a personas desautorizadas.
- Se modifican los recursos sin autorización.
- Se pierden o destruyen los recursos.
- Se interrumpe el acceso a los recursos.

### ***Perfil de la amenaza***

El perfil de la amenaza contiene un rango de escenarios de amenaza para un recurso crítico usando las siguientes fuentes de amenazas:

- Actores humanos con acceso vía red
- Actores humanos con acceso físico
- Problemas del sistema
- Otros problemas

El perfil de la amenaza se representa visualmente usando árboles de amenaza para cada recurso, uno para cada uno de las cuatro fuentes de amenazas.

### ***Propiedades de la amenaza***

- Recurso
- Acceso (optativo)
- Actor
- Motivo (optativo)
- Resultado


## **Paso 2. Vista tecnológica**

- Componentes clave
- Vulnerabilidades tecnológicas

### ***Estrategia de Evaluación de vulnerabilidad***

- Dirigir una evaluación de vulnerabilidad que se enfoque en dónde radiquen los activos críticos
- Identificar componentes clave y revisar la evaluación previa de los resultados para la evaluación de vulnerabilidad de esos componentes
- Hacer una recomendación a largo plazo para construir una dirección de capacidad de dirección de vulnerabilidad

## Herramienta de vulnerabilidad y prácticas

Seguridad Física	Información de Seguridad Tecnológica	Seguridad del Personal
Planes y procedimientos de Seguridad Física Control de Acceso Físico Monitoreo y Auditoría Seguridad Física	 Herramientas de Administración de sistema	Manejo de Incidentes Personal general Prácticas

### Paso 3. Estrategia y desarrollo de un plan

- Riesgos

#### *Riesgo*

El riesgo comprende:

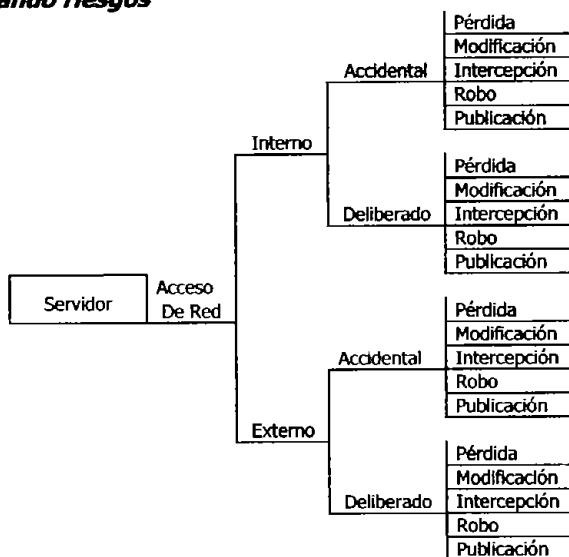
- Un evento (una posibilidad de amenaza)
- Consecuencia (impacto en la organización)
- Incertidumbre (que la posibilidad amenaza ocurra)

Los riesgos se evalúan para determinar:

- Prioridad relativa
- Qué riesgos puede realmente mitigar

La evaluación de impacto se requiere en OCTAVE; la probabilidad cualitativa se está probando en OCTAVE-S.

## Evaluando riesgos



Árbol de evaluación de riesgos<sup>34</sup>

- Estrategia de Protección
- Planes de mitigación

## 3.5. Políticas de Seguridad

Según el diccionario enciclopédico ESPASA<sup>35</sup> una política es el arte o traza con que se conduce un asunto o se emplean los medios para alcanzar un fin determinado, orientaciones o directrices que rigen la actuación de una persona o entidad en un asunto o campo determinado.

Según el RFC 2196 una PS es un documento formal de reglas para todos los usuarios que tienen acceso a los recursos, tecnologías e información de la organización, es una descripción de lo que deseamos proteger y el por qué de ello, este RFC es el que describe las políticas de seguridad y de él se extrajo la sección de políticas de seguridad entre otra bibliografía.

**Políticas Seguridad Informática.** Por todo lo anterior podemos decir que una PS persigue definir las expectativas de la organización en lo que se refiere al buen uso de recursos y contemplar la prevención y respuesta de incidentes, cuidar recursos, dar prevención a incidentes, y si ocurren, cómo los voy a manejar.

<sup>34</sup> OCTAVE. Operationally Critical Threat, Asset, and Vulnerability Evaluation, Software Engineering Institute Carnegie Mellon University. Pittsburgh, EE.UU,

<sup>35</sup> Diccionario Enciclopédico Espasa. México. Espasa Calpe. 1990. Tomo 19.

### **3.5.1. Características de una Buena PS**

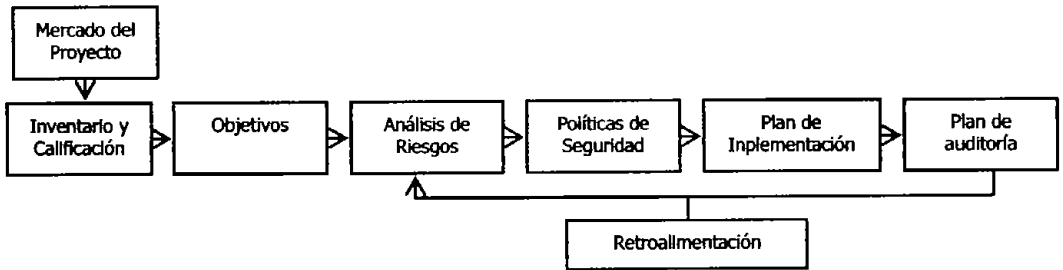
- Uso de lenguaje sencillo y claro evitando lenguaje técnico en lo posible
- Escribir la política como si se fuera a usar siempre
- Aprobada por la alta dirección
- Fácil de mantener
- Implementar a través de procedimientos administrativos
- Lograr la libre actuación de los usuarios
- Hacerla pública
- Enfocarla a problemas reales y grandes
- Apoyarla con herramientas de seguridad
- Acciones rápidas
- Definir claramente las responsabilidades
- Sanciones donde la prevención no sea técnicamente posible.
- Debe hacerse de tal forma que pueda escribirlo cualquier miembro del equipo.
- Evitar describir técnicas o métodos particulares que definan una sola forma de hacer las cosas.
- Hacer referencia explícita y clara a otras dependencias cuando se requiera.
- Utilizar la guía para la presentación de documentos escritos.
- Actualización constante

### **3.5.2. Preguntas para realizar políticas**

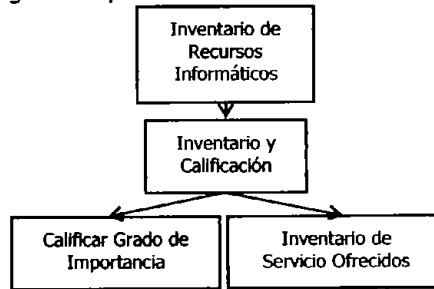
- ¿Cómo será mi mecanismo único y personal?
- Características del mecanismo
- Llevar las características del mecanismo a nivel de políticas
- ¿Qué debemos incluir en una política?
- Los objetivos de la seguridad en informática
- Reglas operativas
  - Definir quien, cómo, cuando y por qué.
  - Detalles para identificar infractores
  - Asignación de responsabilidades

### **3.5.3. Etapas en el Diseño e Implementación de una PS**

El diseñar una PS para su posterior implementación no puede ser un acto caprichoso, de momento, de moda; es un proyecto completo que debe ser asumido con la mayor responsabilidad si se quiere lograr el efecto buscado. Cada una de estas etapas cumple papel importante en el exitoso final del proyecto.



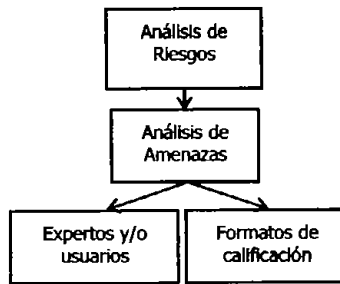
1. Mercadeo del Proyecto. Para facilitar el trabajo de concientización es bueno apoyarse en casos de fallos de seguridad ocurridos en negocios similares y los efectos generados; hacer notar la responsabilidad que cabe a las empresas que no han realizado los esfuerzos necesarios para minimizar los riesgos y que debido a ello pueden infringir normas legales o afectar a otros; para finalizar se puede evaluar la razón costo/beneficio de las medidas tendientes a enfrentar con seriedad los aspectos relativos a seguridad y que el no hacerlo podría llevar a grandes problemas.
2. Inventario y Calificación. Para saber que hay que proteger es necesario hacer un juicioso inventario de recursos informáticos ("hardware", "software" y "liveware"), y de los servicios ofrecidos, donde se determine la importancia para la organización y el grado de criticidad de cada uno. En la siguiente figura se aprecian éstos elementos.



3. Determinar los Objetivos. Están asociados a los objetivos de la Seguridad Informática, orientados a proteger la organización contra amenazas que atenten contra:
  - La continuidad de las operaciones.
  - La confidencialidad y privacidad de la información manejada.
  - La confiabilidad y exactitud el sistema
  - La seguridad física.

Esto se logra observando los objetivos de la seguridad los cuales son:

- Continuidad en las operaciones
  - Confidencialidad y privacidad
  - Confiabilidad y exactitud
  - Seguridad física
4. Análisis de Amenazas. En la siguiente figura se aprecian algunas de las posibilidades que se tienen de conocer los riesgos que se presentan para los diferentes recursos de la organización. Los métodos de análisis de amenazas son:



5. Plan de implementación. Terminado el diseño, se procede a la fase de construcción de las políticas donde se procede a:
- Elegir un escenario de riesgo (ER, casos que pudieran originar debilidades).
  - Determinar las actividades sujetas a control (ASC) asociadas al ER.
  - Elegir una ASC.
  - Identificar amenazas relacionadas a la ASC elegida.
  - Tomar una amenaza.
  - Definir cuales serán los controles para protegerse de la amenaza elegida.
  - Escoger otro ER hasta que se hayan agotado y repetir desde paso 2.
  - Repetir los pasos 3 a 6.
  - Seleccionar los controles a recomendar.

También se puede recurrir a la opinión de expertos y a las vivencias de los usuarios del sistema, para recolectar información de los probables riesgos y los efectos a que están expuestos los procesos de información.

6. Plan de Auditoría. La auditoría es la evaluación y análisis del entorno que cambia con facilidad (y más bruscamente en la informática), de forma crítica, objetiva e independiente, con el objeto de evaluar el grado de protección ante las amenazas. Además de crear buenas políticas, se debe vigilar que se lleven a cabo adecuadamente, y monitorear que estén siempre al día.
7. Retroalimentación. Una vez implantadas las políticas es necesario ver que los cambios generados por la implementación de las PS, sean vigilados y correctamente integrados, para poder llevar un mantenimiento constante, estos cambios se integran con un nuevo análisis de riesgos.

### **3.5.4. Creación del Manual de Políticas de Seguridad en Informática (PSI)**

Después de los pasos previos y se procede a la construcción del documento formal, esta etapa es la del diseño de la PS y deben quedar especificados los diferentes componentes que cubrirá y la forma como se aplicará la misma. Es el documento que contendrá las políticas de seguridad, su aplicación y todo lo respectivo a ellas. A partir de los puntos anteriores y los de esta sección se desarrolló el MPS para el proyecto mencionado en la presente tesis.

Un MPS se forma de los siguientes puntos:

- Índice
- Introducción (Breve historia-empresa- función manual)
- Objetivos
- Misión (SI optativo)
- Visión (SI optativo)
- Áreas
- Organigrama
- Apartado
- Políticas (programas específicos y sistemas específicos). Por:
  - Área
  - Proyecto
- Glosario
- Anexo

Los MP pueden ser a cerca de:

- Políticas sobre programas.
- Políticas sobre temas específicos (uso de la sala de cómputo).
- Políticas sobre sistemas específicos (sistema de inscripción).

### **3.5.4.1. Contenido del documento de PS**

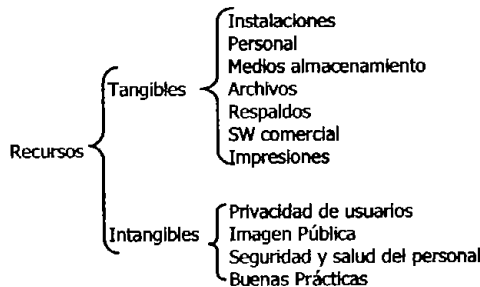
- Elementos asociados la adquisición de hardware, "*software*" y contratación de servicios externos teniendo presente los aspectos referentes a seguridad ("*outsourcing*", mantenimiento a hardware y/o "*software*", desarrollo de aplicaciones, administración de proyectos, etc.).
- Las disposiciones sobre privacidad y control de acceso incluidas las políticas de autenticación.
- Las responsabilidades asociadas a los métodos de actuación y el manejo de incidentes.
- Declaración de la política. Debe contener lo que se desea regular y la posición de la administración.
- Nombre y cargo de quién aprueba o autoriza la política, dependencia, del grupo o de la persona que es el autor o el proponente de la política.
- Debe especificarse a quien está dirigida la política y quién es el encargado de garantizar su cumplimiento.
- Indicadores para saber si se cumple o no la política.
- Referencias a otras políticas y regulaciones con las cuales tenga relación.
- Enunciar el proceso para solicitar excepciones.
- Describir los pasos para solicitar cambios o actualizaciones a la política.
- Explicar que acciones se seguirán en caso de contravenir la política.

- Fechar a partir de la cual tiene vigencia la política, cuando se revisará la conveniencia y la obsolescencia de la política.
- Incluir la dirección de correo electrónico, la página web y el teléfono de la persona (s) que se pueden contactar en caso de preguntas o sugerencias.



# CAPÍTULO 4. SEGURIDAD FÍSICA Y LÓGICA

La Seguridad Física (SF) se refiere a todo lo relacionado con el sistema a proteger que sea tangible, todo lo que pueda tocarse, eso es SF, y la Seguridad Lógica (SL) es todo lo relacionado con lo intangible de un sistema de cómputo, programas, servicios que proporciona el sistema, bases de datos, información y todo la información necesaria para soportar el sistema. Para comprender mejor esto recordemos el cuadro de los diferentes tipos de recursos con que se cuenta mostrados en la sección 3.4.1.



## 4.1. Seguridad Física

La SF es el acceso físico hacia los recursos que manejan la información; es la primera capa de seguridad que se necesita tener en cuenta, para ello es importante plantearse las siguientes preguntas:

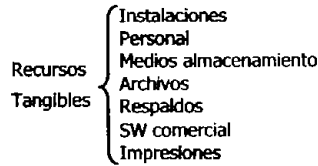
- ¿Quién tiene acceso físico directo a su máquina?
- ¿Deben tener estas personas acceso?
- ¿Se puede proteger el equipo de acceso franco?
- ¿Debe hacerlo?

El determinar qué cantidad de SF habrá depende de los recursos que deben protegerse y con cuánto presupuesto se cuenta para ello, y también de esto depende cuánto tiempo y esfuerzos se invertirán en esa tarea.

Los métodos de seguridad físicos son tan sencillos como las cerraduras en las puertas, protección a los cables, cerrar con llave armarios, o vigilancia pueden ser opciones viables, pero dado que el servidor con que se cuenta para el proyecto se encuentra en la oficina de los directores de proyectos, resultan innecesarias estas medidas, por lo que sólo se tomarán en cuenta aspectos como candados al CPU, seguridad en el BIOS, seguridad en el cargador del boot, proteger con contraseña el inicio de sesión y la interfaz de texto<sup>36</sup> entre otras cosas.

<sup>36</sup> En LINUX existen dos tipos de interfaz: la gráfica y la de texto, la grafica es la que se puede ver en un ambiente de

Debe tomarse en cuenta lo siguiente:



### 4.1.1. Medidas a tomar en cuanto a SF

- Ubicación física y disposición del centro de cómputo
- Instalaciones físicas del centro de cómputo.
- Control de acceso físico
- Suministro de energía
- Aire acondicionado
- Protección, detección y extinción de incendios
- Protección contra inundaciones
- Mantenimiento

## 4.2. Seguridad lógica

La SL es todo lo relacionado con lo intangible de un sistema de cómputo, programas, servicios que proporciona el sistema, bases de datos, información y toda la información necesaria para soportar el sistema y comprende lo siguiente:

	<b>Privacidad de usuarios</b>
Recursos Intangibles	Imagen Pública
	Seguridad y salud del personal
	Buenas Prácticas

En este punto el objetivo es proteger los activos de información del sistema de cómputo para que sean utilizados siempre de forma autorizada, sólo por necesidad, y evitar acciones que puedan provocar su pérdida, modificación, robo, interrupción o publicación o divulgación no autorizados, de forma accidental o intencionada.

Según la DGSCA (Dirección General de Servicios de Cómputo Académico), la seguridad lógica comprende las siguientes tareas:

- **Integridad.** Un sistema debe hacer solamente lo que esta supuesto que hará y nada más. Debe funcionar de acuerdo a las especificaciones planteadas (aún cuando falle).

---

ventanas y paneles despleables, y la de texto son aquellas comúnmente llamadas consolas, que presentan la línea de comandos para poder dar instrucciones.

- **Aislamiento.** La protección se logra aislando el objeto valioso, y controlando el acceso a él. Algunos métodos para poder hacerlo son:
  - Seguridad por oscuridad
  - Criptografía
- **Control de Acceso.** Es detectar quién está entrando al sistema, si está o no autorizado, para qué, y con que privilegios. Algunos mecanismos para lograr esto son:
  - Passwords
  - Sistemas dedicados
- **Monitoreo.** La operación exitosa de un sistema seguro depende no solamente de las técnicas de seguridad utilizadas para construirlo, sino del monitoreo continuo de estas técnicas para detectar cualquier vulnerabilidad. Esto se puede lograr a través de verificar:
  - Comandos
  - Herramientas
  - Bitácoras
- **Respaldos.** Estar llevando a cabo constantemente la tarea de respaldar toda la información de la máquina que tienen la función de ser el servidor.

#### **4.2.1. Seguridad Local**

Se debe poner atención en la seguridad del sistema contra ataques de usuarios locales. Conseguir acceso a través de cuentas de auténticos usuarios es una de las cosas que los intrusos hacen mientras intentan apoderarse de la cuenta de "root", pueden hacerlo con la ayuda de mala seguridad y mala administración de los servicios locales, usando una variedad de herramientas.

Es importante proporcionar cuentas sólo a usuarios conocidos, de quienes se tiene información, por todo lo anterior es importante observar los siguientes puntos:

- Seguridad en cuentas
- Seguridad de "root"
- Seguridad de contraseñas.

#### **4.2.2. Seguridad en el sistema de archivos y archivos individuales.**

El sistema de archivos es la forma en que están organizados jerárquicamente los archivos en linux, donde se encuentran los componentes del sistema operativo, y sistema, y donde están los de los usuarios y del administrador de sistema. Es importante planear y prepararse antes de poner un sistema en línea ya que esto puede ayudar proteger los datos guardados en él. Para poder vigilar adecuadamente el sistema de archivos deben vigilarse los siguientes puntos:

- Permisos en archivos
  - SUID y SGID
  - STICKY BIT
- Chequeos de integridad

### 4.2.3. Seguridad con criptografía

La criptografía es la ciencia matemática que estudia el almacenamiento y transmisión de la información de manera privada y segura, se intenta hacer inaccesible la información a personas no autorizadas. El proceso es tomar un mensaje normal y pasarlo a través de algoritmos de cifrado, esto es, cifrar un mensaje, después de ello sólo será legible para quienes posean la clave, o método para averiguar el significado oculto.



El sistema operativo Unix proporciona un mecanismo de cifrado llamado DES, Norma de Cifrado de Datos (acrónimo de Data Encryption Standard), es un algoritmo desarrollado por el Instituto Nacional de Estándares y Tecnologías (NIST National Institute of Standards and Technologies) para codificar y decodificar datos y contraseñas. Aunque originalmente DES es un algoritmo bidireccional, es decir, se puede cifrar y después descifrar un mensaje, dando las llaves (forma parte del proceso de cifrado, es una clave para poder cifrar y descifrar) correctas, en Unix DES es unidireccional, es decir, no debe ser posible invertir la criptografía para recibir la contraseña de `/etc/passwd` o `/etc/shadow`, donde se encuentran las contraseñas cifradas, cuando se intenta entrar al sistema la contraseña requerida es cifrada de nuevo y comparada con la entrada registrada en éstos archivos, si son iguales, se permite el acceso, de esta forma se asegura que no habrá forma de revertir el cifrado y llegar a obtener la llave o el mensaje en texto claro.

#### 4.2.3.1. Mecanismos de cifrado

Después de ver en párrafos anteriores, se puede decir que la criptografía abarca el uso de mensajes encubiertos, códigos y cifras, pero, para poder llevar a cabo este proceso se vale de diversos mecanismos, los cuales son:

##### 4.2.3.1.1. Criptografía de Llave pública

La criptografía de llave pública, como el usado para PGP (algoritmo de cifrado "Pretty Good Privacy", Privacidad Bastante Buena), usa una llave para cifrar, y otra para descifrar.

Para transmitir llave de forma segura la criptografía de llave pública usa dos llaves separadas: una llave pública y una llave privada. La llave pública está disponible para hacer el cifrado, mientras al mismo tiempo se guarda la llave privada para descifrar los mensajes cifrados con la llave pública correcta. Linux tiene soporte para PGP si desea usarse. Permite las siguientes posibilidades:

- Firmar digitalmente un texto, un documento o un archivo, autenticándolo y evitando que pueda ser alterado o modificado.
- Cifrar y firmar Simultáneamente.
- Es fácil de usar
- Es Seguro pues está sometido a constantes pruebas.
- La Firma digital permite asegurar la integridad de un documento o un archivo, así como conocer quien nos lo envía.

#### **4.2.3.1.2. SSL y S-HTTP**

Estos son protocolos criptográficos y a continuación se da una breve explicación de c/u de ellos.

- **SSL.** Secure Socket Layer es un método criptográfico desarrollado por Netscape para proporcionar seguridad en Internet. Soporta varios protocolos criptográficos diferentes, y proporciona autenticación del cliente y servidor. SSL trabaja en la capa de transporte, y crea un canal cifrado seguro de datos, cuando el cliente pide al servidor seguro una comunicación segura, el servidor abre un puerto cifrado, gestionado por un software llamado Protocolo SSL.

Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, y cifrando la clave de sesión mediante un algoritmo de cifrado de clave pública. La clave de sesión es la que se utiliza para cifrar los datos que vienen del y van al servidor seguro. Se genera una clave de sesión distinta para cada transacción, lo cual permite que aunque sea capturada por un atacante en una transacción dada, no sirva para descifrar futuras transacciones.

- **S-http.** Secure HyperText Transfer Protocol fue desarrollado por Enterprise Integration Technologies (EIT) y es otro protocolo que proporciona los servicios de seguridad para Internet. Permite tanto el cifrado como la autenticación digital, pero es un protocolo de nivel de aplicación, es decir, que extiende el protocolo HTTP por debajo. Fue diseñado para proporcionar confidencialidad, autenticación, integridad, y no repudio (que no puede confundirse con otra persona). Hace la negociación entre las partes involucradas en cada transacción. Cifra cada mensaje individualmente.

#### **4.2.3.1.3. SSH**

Permite realizar transferencia de datos de manera segura. Una vez que se instala en el servidor o la maquina que va a funcionar como servidor el software de servidor de SSH (Secure Shell) y en la Pcs clientes el software respectivo, se pueden realizar conexiones seguras del cliente al servidor, de tal manera que la transferencia de datos no pueda ser robada o vista mientras se hace la comunicación. SSH reemplaza a comunicaciones inseguras como telnet, ftp, rlogin y rsh entre otras y manda toda la información cifrada para que no sea legible.

#### **4.2.3.1.4. STELNET**

SSLeay es una aplicación libre del protocolo de Netscape capa conexiones seguras, incluye varias aplicaciones, como el telnet seguro, entre otras, usando esta biblioteca, se crea un reemplazo de telnet seguro y cifra sobre una conexión del telnet.

#### **4.2.3.1.5. PAM - los Módulos de la Autenticación Conectables**

PAM permite cambiar sobre la marcha los métodos de verificación, requisitos y encapsular todos los métodos de verificación sin recompilar ninguno de los binarios. Es un mecanismo flexible para la autenticación de usuarios, utiliza una arquitectura conectable y modular, que otorga al administrador del sistema de una gran flexibilidad en establecer las políticas de autenticación para el sistema. PAM permite el desarrollo de programas independientes del mecanismo de autenticación a utilizar.

#### **4.2.3.1.6. Cryptographic IP Encapsulation (CIPE)**

CIPE cifra los datos a nivel de red. El viaje de los paquetes entre hosts se hace cifrado. A diferencia de SSH que cifra los datos por conexión, lo hace a nivel de socket. Así un conexión lógica entre programas que se ejecutan en hosts diferentes está cifrada.

CIPE se puede usar para crear una Red Virtual Privada. El cifrado a bajo nivel tiene la ventaja de poder hacer trabajar la red de forma transparente entre las dos redes conectadas en la RVP sin ningún cambio en el software de aplicación.

#### **4.2.3.1.7. Kerberos**

Kerberos fue desarrollado en el MIT por el proyecto Athena. Es un protocolo de autenticación que utiliza una tercera parte confiable, diseñada para redes TCP/IP. Un servicio Kerberos, sobre una red, actúa como un árbitro confiable, provee autenticación segura en la red, permitiendo a una persona a acceder a diferentes personas sobre la red. Está basada en criptografía simétrica.

En el modelo Kerberos hay entidades (clientes y servidores) sobre una red, los clientes pueden ser usuarios, programas independientes de software que necesiten hacer algo (bajar archivos, enviar mensajes, acceso a Bases de Datos, obtener privilegios de administrador).

Este protocolo mantiene una base de datos de clientes y sus llaves secretas. Dado que conoce todas sus llaves secretas, puede crear mensajes para convencer a una entidad que la otra entidad es quien dice ser. También puede crear llaves de sesión para que cliente y servidor se comuniquen con seguridad.

#### **4.2.3.1.8. Contraseñas de Shadow.**

Las contraseñas de Shadow son un medio de mantener seguro la información de las contraseñas cifradas de los usuarios normales. Red Hat usa contraseñas de Shadow por defecto, pero de no ser así se guardan sin cifrar en `/etc/passwd` que todos pueden leer.

Las contraseñas de Shadow son almacenadas en `/etc/shadow` que sólo usuarios privilegiados pueden leer. Para usar las contraseñas de Shadow, es necesario asegurarse que todas las utilidades que necesitan acceso a la información de la contraseña sean recompilados.

#### 4.2.4. Seguridad del Núcleo (Núcleo).

Como el núcleo controla la red de la computadora, es importante que esté muy seguro, no comprometido, y al día para evitar ataques nuevos. Para ello se pueden hacer las siguientes tareas:

- Configurar las opciones de compilación del Núcleo (sección 6.4.5.1.)
- Configurar los dispositivos del Núcleo (sección 6.4.5.2.)

### 4.3. Seguridad de Red

La seguridad de red es tratar de proteger al sistema de cualquier tipo de ataque por esta vía, y es muy importante proteger el sistema cuando éste presta servicios a usuarios a través de ella, para poder protegerla es importante observar los puntos subsecuentes.

#### 4.3.1. Sniffers

Comúnmente llamados "*Packet Sniffers*" (paquetes espías), es una de las formas más frecuentes que tienen los intrusos para obtener acceso a sistemas de la red, usándolos con una máquina que ya ha sido comprometida, lo que se hace es escuchar en los puertos Ethernet buscando "*Password*", "*Login*" y "*su*" en el flujo de paquetes y registra el tráfico posterior a éstas cadenas de caracteres, y obtienen claves de sistemas que pueden ser o no de los que tratan de atacar, pues por el flujo es de la red completa, por ello la importancia del cifrado.

El uso de ssh u otros métodos de claves cifradas impide este ataque. Aplicaciones como APOP para cuentas de correo también previene este ataque.

#### 4.3.2. Servicios del sistema

En cuanto se ponga el sistema en red, se debe verificar qué servicios se necesita ofrecer, y desactivar los que no sean necesarios descartando posibles ataques por medio de esos servicios. La forma de protegerse de esta vulnerabilidad es bloqueando lo servicios que no se necesiten y protegiendo los que se deben dar. Para mas detalle ver la sección 6.5.2.

Algunos servicios que se podrán necesitar (de acuerdo con los requerimientos de sistema) activos son:

- ftp
- telnet
- mail, como pop-3 o imap
- identd
- time

### 4.3.3. Identificación de hosts vía DNS

Esto es muy importante, ya que se debe mantener actualizada la información DNS (Domain Name Server) de todas las máquinas de la red puede ayudar a aumentar la seguridad. En el caso de un equipo no autorizado se conecte a su red, se puede reconocer por la ausencia de la entrada DNS. Muchos servicios se pueden configurar para no aceptar conexiones de equipos que carecen de entradas DNS válidas.

### 4.3.4. Provisión de seguridad en correo

Uno de los servicios más importantes que se pueden proporcionar es el servidor de correo, pero es uno de los servicios más vulnerable a los ataques debido al número de tareas que debe realizar y los privilegios que necesita.

Generalmente se usa *sendmail*, que tiene muchos fallos de seguridad, por ello es muy importante mantener la versión actualizada (<http://www.sendmail.org>). Otra opción para proporcionar este servicio es *qmail*, que se diseñó contemplando la seguridad de principio a fin, es más rápido, estable y seguro (<http://www.qmail.org>).

### 4.3.5. Escaneo de puertos

Hay algunas herramientas diseñadas para alertarle de pruebas con Satan, ISS y otro software de exploración. De todas formas el uso de *tcp\_wrappers* y estando seguro de buscar en los ficheros regulares de registro debería notar tales pruebas. Incluso en el más bajo nivel, Satan aun deja trazas en los registros en un sistema Red Hat.

Hay varios paquetes diferentes de software que efectúan una exploración basada en puertos y servicios de máquinas o de redes, como lo son SATAN y ISS, este software se conecta con la máquina destino (o todas las máquinas de la red) en todos los puertos que puede e intenta determinar qué servicio se están ejecutando. Basándose en esta información, podría descubrir si la máquina es vulnerable a un ataque específico en ese servidor.

Hay varios paquetes diferentes de software que efectúan una exploración basada en puertos y servicios de máquinas o de redes. SATAN y ISS son dos de los mejores conocidos. Este software se conecta con la máquina destino (o todas las máquinas de la red) en todos sus puertos que puede e intenta determinar qué servicio se está ejecutando allí. Basándose en esta información, podría descubrir si la máquina es vulnerable a un exploit específico en ese servidor.



### 4.3.5.1. Detección de intrusos

Como se explicó en párrafos anteriores, el *escaneo* de puertos es muy peligroso, ya que puede detectar vulnerabilidades en el sistema y explotarlos para dañar el servidor, para ello es importante hacer un monitoreo de los usuarios detectados en el sistema y vigilarlos. Para ello hay diversas herramientas:

**Identd.** Es un pequeño programa que se ejecuta desde *inetd*. Mantiene la pista de qué usuario está ejecutando un determinado servicio, e informa de ello cuando se le solicita.

Aunque no hay forma de conocer si los datos que obtiene del *identd* remoto son correctos o no y no hay verificación de identidad en las solicitudes, *identd* ayuda y es otra fuente de información cuando *identd* no está alterado, entonces sabe que le está diciendo el usuario o *uid* (Identificador de usuario) de la gente de los sitios remotos que están usando los servicios *tcp*, y así se puede tomar acciones contra ese usuario. Si no se usa *identd*, se pueden recurrir a los registros de sistema para encontrar quien fué a qué hora, etcétera, para saber algo a cerca del intruso. *Identd* viene con la mayoría de las distribuciones.

### 4.3.6. Negación de Servicios

Un ataque de negación de servicios es aquél en que el atacante intenta hacer que algún recurso esté demasiado ocupado para responder solicitudes legítimas o para denegar a los usuarios legítimos acceso a su máquina. A continuación se listan algunos de ellos:

- SYN Flooding. Es un ataque de negación de servicio de red. Se aprovecha de un hueco de seguridad en la forma en que se crean las conexiones TCP. Los núcleos de Linux 2.0.30 y posteriores tienen varias opciones configurables para prevenir ataques de este tipo. Ver sección 4.2.4.
- Pentium "FOOF" Bug. Recientemente se descubrió que una serie de código ensamblador enviado a un Pentium genuino reiniciaba la máquina. Esto afecta a todas las máquinas con un procesador Pentium solamente, sin importar que sistema soporte. Los núcleos Linux 2.0.32 y superiores tienen un trabajo sobre este bug, previniendo que bloquee la máquina. El núcleo 2.0.33 tiene una versión mejorada de la corrección, sugerida sobre 2.0.32.
- Ping Flooding. Es un ataque de negación de servicio por fuerza bruta. La forma en que se da es que el atacante envía una "inundación" (flood) de paquetes a la máquina, si se hace desde un equipo con mayor ancho de banda que su máquina, será incapaz de enviar algo a la red. Otro similar es *smurfing*, que envía paquetes a un equipo con la dirección IP de retorno de la máquina, permitiéndoles que la inundación sea menos detectable.

Para evitar estos ataques se puede usar *tcpdump* y determinar de donde vienen los paquetes (o parece que vienen), entonces contactar al proveedor con esta información, también pueden ser evitados con *tuteadotes* o *cortafuegos*.

- Ping o Death. Es el resultado de paquetes entrantes que son más grandes de lo que pueden almacenar las estructuras de datos del núcleo que recogen esta información, y por ello, los cuelga o los rompe.
- Teardrop / New Tear. Usan un hueco presente en el código de fragmentación IP en plataformas Linux y Windows.

Algunos de estos problemas ya se resolvieron en distribuciones recientes de Linux, pero es importante verificar cuales son y corregir los que no han sido corregidos.

## 4.4. Seguridad preventiva

Hay algunas acciones que se pueden tomar para adelantarse a cualquier incidente o ataque, y así poder evitarlo, y es de estas cuestiones de las que tratará la presente sección de esta tesis, con el fin de proporcionar herramientas para lograr la preservación del sistema a proteger.

### 4.4.1. Firewalls

Los Firewalls (cortafuegos) son medios de restringir qué información se permite que entre y salga de la red local, normalmente el equipo cortafuegos está conectado a Internet y a la red local, y sólo se accede desde la red a Internet a través de él, de esta forma se puede controlar qué entra y sale de internet y la red local.

Hay diversas formas y métodos de activar un Firewall, en Linux la herramienta *ipfwadm* permite cambiar los tipos de tráfico de red que se necesite sobre la marcha. También puede registrar tipos de tráfico particulares de la red. Es muy importante saber que no es suficiente contar con un Firewall para protegerse de posibles ataques, ya que se necesitan asegurar las máquinas que están detrás de él.

### 4.4.2. Respaldos

Los respaldos son copias a documentos, archivos, ficheros, e incluso sistemas de archivos, críticos, es decir, que información de mi sistema es indispensable tener en algún tipo de dispositivo externo para evitar su pérdida total.

#### 4.4.2.1. Políticas de respaldos

Cuando se decide hacer respaldos es importantísimo decidir cómo es que se van a llevar a cabo, cada cuándo, en qué medio, y otros puntos que se tratan a continuación:

- Centralizar la máquina de respaldo para una mejor organización.
- Etiquetar los respaldos. Tomando en cuenta los siguientes aspectos:
  - Sintaxis. Instrucción o en todo caso la(s) utilería(s) que se utiliza; además de especificarse los archivos y/o files system respaldados.
  - Hostname. (Nombre de la máquina servidor).
  - Definir el intervalo de tiempo para respaldar, en función de la importancia y dinamismo de la información. Y a su vez eligiendo los sistemas de archivos o archivos necesarios a respaldar.

		INTERVALO
<input type="checkbox"/>	www Internet	1 semana
<input type="checkbox"/>	@ mail	Diario
<input type="checkbox"/>	Base de Datos	Diario
<input type="checkbox"/>	Desarrollo	1 semana
<input type="checkbox"/>	Sistema Operativo	1 mes

La tabla anterior es un ejemplo de cómo distribuir los intervalos de tiempos aproximados, para hacer los respaldos de determinada información, según la importancia que se les otorgue o se crea necesario, por la cantidad de información que se encuentra en constante "movimiento".

- Tamaño de lo respaldado. Es necesario adecuar los files system al medio donde se va a almacenar.
- Guardar los respaldos fuera del sitio. Recomendable de que también se tenga un respaldo dentro.
- Reducir las actividades al máximo cuando se está haciendo un respaldo.
- Verificar que el respaldo esté bien hecho.
- Que el tiempo que se demora en restaurar mi respaldo sea el adecuado.

Todas las consideraciones anteriores deben de ir especificadas en el manual de políticas de seguridad.

Es muy importante decidir cuándo se harán las copias de los siguientes puntos:

- Del sistema
- De la base de datos
- De programas como rutinas

A continuación se presenta una lista de los diversos medios en los que se pueden guardar los respaldos y algunas de sus características.

Medio	Capacidad	Velocidad	\$Dispositivo	\$Media
Floppy	1.4 Mb	< 100 kb/s	15.0	25
Zip	1.20 Mb, 1.50 Mb	900 kb/s	200.0	8
CD-R	650-800 Mb	2.4 Mb/s	200.0	1.18
CD-RW	650-800 Mb	2.4 Mb/s	250.0	3
DVD	7-10 cd's	?	300.0	5
LX EXABYTE (8mm)	7-8 Gb	1 Mb/s	1200.0	8
DDS2, DDS3, DDS4 (4mm)	20 Gb	2.5 Mb/s	1000	30
DLT	40-80 GB	6 Mb/s	4000	60

Algunas herramientas para hacer respaldos en Linux son:

- TAR. Su nombre viene de "Tape Archiver". Permite empaquetar archivos en un empaquetador TAR, es decir, almacena otros archivos en uno solo. El contenedor puede ser otro archivo, una cinta, o una tubería, los archivos deben ser ordinarios, FS completos.
- DD. Comando que permite copiar de dispositivo a dispositivo.
- CPIO. Permite empaquetar archivos en un contenedor CPIO, almacena otros archivos en uno solo, ya sea otro archivo, una cinta, o una tubería, los archivos deben ser ordinarios, FS completos.

### 4.4.3. Proteger las bitácoras

Las bitácoras son registros de sistema en donde se guardan eventos importantes para el mismo, como lo son comandos, accesos, mensajes del sistema, etc. Se pueden ubicar en */var/adm*, */usr/adm*, o en */var/log*. A continuación se detallan sus registros:

- **acct o pacct.** Graban todos los comandos.
- **lastlog.** Ultimos accesos.
- **messages.** Mensajes del sistema.
- **Sulog.** Uso del comando su.
- **utMp.** Cada usuario en el sistema.
- **wtMP.** Registro de conexiones de usuarios.
- **xferlog.** Accesos a *ftp*.

*Syslog* es la bitácora del sistema, y es un sistema centralizado de bitácora. Corre el demonio *syslogd* y cada mensaje que guarda se compone de 4 partes:

1. Nombre del programa
2. Característica
3. Prioridad
4. Mensaje

Cada mensaje maneja categorías, las cuales son:

- **Kern.** Mensajes del *kernel*.
- **user.** Procesos de usuario.
- **mail.** Correo electrónico.
- **lpr.** Sistema de impresión.
- **auth.** Accesos.
- **daemon.** Demonios.
- local0 a local7 Reservados.

Las diversas prioridades que tiene un mensaje son:

- **emerg.** Error muy grave, caída inminente del sistema.
- **alert.** Condición que debe ser corregida de inmediato.
- **crit.** Condición crítica.
- **err.** Error ordinario.
- **warning.** Aviso.
- **notíce.** No es un error pero debe ser revisado.
- **info.** Mensaje informativo.
- **debug.** Incluye todo tipo de mensajes.
- **none.** No manda mensajes.

## 4.5. Herramientas de seguridad

Red Hat tiene herramientas de seguridad, la gran mayoría de ellas trabajan para proteger activamente el sistema. Algunas de las más comunes y útiles herramientas son:

- **Utilidades Shadow.** Es una colección de herramientas para administrar usuarios locales y grupos en un sistema que utiliza contraseñas cifradas.
- **Kerberos 5.** es un sistema seguro que proporciona servicios de autenticación<sup>37</sup> de redes. No permite que contraseñas de texto común pasen sobre una red para obtener acceso a los servicios.
- **OpenSSL.** Ayuda a proteger una amplia gama de servicios que soportan operaciones sobre una capa de cifrado.
- **OpenSSH.** Son un conjunto de utilidades que fácilmente pueden sustituir herramientas presentes pero inseguras como "telnet" y "ftp" para transmisión de archivos, con las potentes y seguras "ssh" y "scp" en caso de ser necesario este tipo de comunicación.
- **Tripwire.** Es una aplicación que sirve para alertar modificaciones en directorios y ficheros de sistema específicos, así se sabrá si hay usuarios no autorizados que están teniendo acceso a su sistema o si usuarios autorizados están haciendo cambios indeseados a ficheros importantes. Se encarga de verificar la integridad del sistema. Detecta cambios en los archivos a través de la comparación con firmas digitales, además monitorea cambios en permisos, ligas, tamaños y directorios.
- **COPS.** Son varias herramientas de seguridad que permiten desde monitorear el control de puertos abiertos en un determinado equipo a estar atento a contraseñas de usuario incorrectas. Son un juego de scripts de shell para probar el sistema que verifican archivos buscando cambios sospechosos.
- **Tcp Wrappers.** Programas que proporcionan filtros a la mayoría de los servicios de red de UNIX tales como ftp, tftp, exec, rsh, telnet, rlogin, finger etc. Pueden ser instalados sin ningún cambio al software instalado.
- **Secure Shell.** Es un programa mediante el cual se realizan conexiones entre máquinas a través de una red abierta de forma segura.

Hay algunas actividades generales que se pueden hacer para proteger el sistema:

- Limitar el número de usuarios que pueden ejecutar comandos como "root". Ya sea intencionalmente que accidentalmente de alguien que conoce la contraseña de "root" o a quien ha sido dado permiso por medio de "sudo" para ejecutar un comando en el ámbito de "root".
- Saber qué paquetes de software se han instalado en el sistema y permanecer alerta para detectar huecos de seguridad, así saber qué paquetes hay que supervisar y cuándo hay que actualizarlos.

---

<sup>37</sup> Autenticar es verificar la identidad de ser, en verdad, quien se dice ser.

- Limitar los servicios que se ejecutan en el sistema sólo a aquellos estrictamente necesarios. Es importante conservar los recursos de sistema y desinstalar paquetes que se usan. Para ello se puede ejecutar una herramienta como `ntsysv` para evitar que servicios innecesarios inicien con el sistema a la hora del arranque.
- Requerir que los usuarios creen contraseñas seguras y cambiarlas a menudo o manejar las cuentas de usuario a través del administrador de sistema y dar contraseñas por defecto.
- Asegurarse que los permisos de ficheros no estén abiertos sin ser necesario, ya que la mayoría de los ficheros no deberían ser escribibles.
- Convertir en una rutina la actividad de supervisión de los registros de sistema. Red Hat guarda datos útiles en los registros de sistema situados en el directorio `/var/log`, especialmente en el fichero `messages`. Una tarea sencilla ejecutada como `root`, como el comando `grep "session opened for user root" /var/log/messages | less`, permite desempeñar una revisión parcial del sistema y supervisar quién está en él como `root`. Es importante considerar que este no es un método a prueba de fallos, pues alguien con el permiso de escritura en un fichero de sistema importante podría modificar el fichero `/var/log/messages` para borrar su rastro.

# **PARTE III. ESTUDIO DE CASO: APLICACIÓN DE CALIDAD Y SEGURIDAD EN EL SISTEMA WEB "COMUNIDADES DE APRENDIZAJE IZTACALA"**

---

---

## **CAPÍTULO 5. APLICACIÓN DE CALIDAD**

---

---

### **5.1. Cuestiones Generales**

En esta parte de la tesis se hará la presentación de los estándares, principios y reglas que se tomaron en cuenta empleadas en el proyecto del cual ya se ha hablado ampliamente en la parte de Contexto de Desarrollo del Proyecto. A grandes rasgos se puede decir que en el Laboratorio de Interacción Humano - Máquina del CCADET de la UNAM se tiene el proyecto de desarrollar un sistema llamado "Sistema de Comunidades de Aprendizaje Iztacala", para los docentes de la carrera de Odontología del Módulo de Instrumentación de la FES Iztacala que permitirá tener una comunicación y retroalimentación entre los usuarios del sistema y sus diversas comunidades. Los desarrolladores este sistema son algunos de los integrantes del Laboratorio de Interacción Humano - Máquina del CCADET de la UNAM.

Comúnmente para cada proyecto se cuenta con partes definidas como lo son la Definición, Desarrollo y el mantenimiento, pero en este caso se tomaron etapas más precisas, las cuales quedaron de la siguiente forma:

- Análisis
- Diseño
- Codificación
- Pruebas

Se hizo de esta manera por que el equipo de trabajo acordó dividirlo así por los roles que desempeñaría cada uno, así que por lo práctico resulta mejor, y este modelo se apega los descritos en las diversas bibliografías que consulté. Y es el Modelo Lineal Secuencial, pero en su segunda versión (Ver sección 1.3.4.).

## 5.2. Modelo Elegido: Evolutivo

El modelo fue elegido por que así fue desarrollado el sistema, pero en vez de contar con etapas como Especificación, Desarrollo, y Validación se dividieron en las siguientes:

Análisis	⇒	Versión inicial
Diseño	⇒	Versión intermedia
Pruebas	⇒	Versión final

El propósito de que se escogiera este tipo de desarrollo, es por que se adapta perfectamente a la forma en que se desarrollan los proyectos en el LIHMYM y por ello se escogió el Desarrollo Evolutivo Exploratorio, ya que primeramente se tuvo conocimiento de los requerimientos del usuario y en base a ello se desarrolla.

## 5.3. Ciclo de Vida de Nuestro Sistema

9. **Viabilidad.** A este tipo de estudios también se les llama estudios de factibilidad y suelen encontrarse indistintamente, a continuación se presentan los estudios del sistema:
- a) **Factibilidad Técnica.** Tanto los dispositivos físicos como lógicos están disponibles en el laboratorio del CCADET y hay gran diversidad de ellos, por lo tanto el proyecto tiene recursos suficientes. El riesgo de desarrollo es bajo, ya que el equipo de trabajo ya ha realizado trabajos de este tipo, inclusive sin todo el equipo. Se requiere lo siguiente:
- Una computadora con espacio libre en disco duro de 4.5 GB, como mínimo. Se recomiendan equipos similares o superiores a Pentium III, que tengan mínimo en memoria RAM 128 MB. Sólo se necesitará una terminal dedicada a ser servidor.
  - Para Linux debe existir, forzosamente, al menos dos particiones raíz (/) y swap<sup>38</sup>. La cual debe ser dos veces lo que se tenga como memoria RAM, si se tiene 128 en RAM, swap deberá ser de 256 MB. La partición Raíz, ocupa por sí sola el espacio del sistema operativo, pero es mejor asignar particiones exactas en espacio, para proteger al mismo, por lo que se recomienda una partición de 384 MB.
  - Se requiere una unidad de respaldo de disco compacto.
  - Es necesario tener acceso a Internet con el fin de proporcionar el servicio a los usuarios; el ancho de banda con el que se cuenta es de la UNAM, suministrado por DGSCA, lo que garantiza un servicio fluido y eficiente.
- b) **Viabilidad Económica.** El presupuesto planeado no se ve muy afectado por los gastos en recursos materiales, por lo que respecta a recursos humanos el presupuesto asignado fue distribuido con becas para tres estudiantes y un diseñador que participan en el proyecto. Los recursos se distribuyeron de la siguiente forma:

---

<sup>38</sup> Swap es el área también conocida con el nombre de espacio de intercambio o memoria virtual, ésta área se utiliza cuando la memoria RAM es limitada, es donde se colocan los procesos que no caben en memoria.



RECURSOS MATERIALES			
Oportunidad	Costo \$	Deseable	
		Si	No
Equipo Servidor	Ya se Tenía	✓	
Linux	Freeware	✓	
PHP	Freeware	✓	
Apache	Freeware	✓	
MySQL	Freeware	✓	
Java	Freeware	✓	

- El mobiliario no representa un gasto más, ya que se cuenta con todo el material necesario para alojar el sistema, escritorio, computadora, regulador, CD writer, etcétera.
- La instalación eléctrica ya se tiene, ya que es un una oficina habilitada con suministros de energía y todo lo necesario.

c) **Viabilidad Operacional.** Los cambios que se tendrán en cuanto a organización son muy pocos, la administración e instalación del servidor estará a cargo de un técnico en informática o persona (s) capacitada (s) para atender cualquier inconveniente, quién deberá estar familiarizados con el uso de las herramientas empleadas. El usuario no verá reflejado ningún cambio mayor, ya que las transacciones dentro del sistema se realizan como las que comúnmente se hacen en internet. Se requerirá el desarrollo de un manual de políticas, el cual será parte de la presente tesis y se presenta en la sección 6.3.

10. **Detección de requerimientos de información.** Las propuestas se hicieron entre todo el equipo de trabajo, principalmente los directores de proyecto, el analista, y el diseñador del sistema en base a la información que se obtuvo de los usuarios, y del sistema anterior a este que contaba con funciones similares, pero no tan explícitas.
11. **Análisis.** Esta parte se está realizando paralelamente a este trabajo, para ello se están valiendo del análisis orientado a objetos.
12. **Diseño, Desarrollo e Implementación.** Este sistema fue hecho de manera que se llevaran a cabo estas tareas paralelamente y está siendo diseñado desde antes que la presente tesis se concretara, y se vale de las herramientas de diseño orientado a objetos, igual que el Análisis, por las necesidades del sistema. El sistema requiere de ser una aplicación Web, para poder prestar servicio. El diseño del sistema fue hecho, en parte por diseñadores profesionales, que llevaron a cabo estudios de "*Benchmarking*" y se diseñó de forma acorde a la naturaleza del sistema. El desarrollo fue hecho por un programador que hizo las conexiones valiéndose de lenguajes como Java, PHP, SQL y otros más, siguiendo en cada una de las etapas las indicaciones de calidad que se le dieron al principio del proyecto.
13. **Pruebas.** El sistema deber ser probado por el equipo de trabajo, además de que se llevarán a cabo pruebas con el usuario para ver si cumplía con los requerimientos, de las cuales se presentarán los resultados en capítulos posteriores.

14. **Documentación.** La documentación debe ser hecha por los encargados del análisis.
15. **Mantenimiento.** Esta etapa queda en manos de los administradores del sistema, y a ellos les corresponde persistir en la preservación de la salud del sistema.

#### **5.4. Aplicación de los Principios y Factores de Calidad en el Software. Asignación de Tareas.**

Para lograr el seguimiento de los factores y principios tan importantes de calidad, se proporcionó a cada uno de los integrantes del equipo del proyecto el documento de Asignación De Actividades por etapa, es decir, a los integrantes del Análisis, Diseño y Programación (ver Anexo 1), para que observaran estas pautas antes y durante y después de llevar a cabo las tareas correspondientes a su etapa. Con el fin de detectar si se habían seguido estos parámetros, se llevaron a cabo la aplicación de dos cuestionarios de evaluación interna, uno al inicio del proyecto y otro casi al final (ver Anexo 2), los resultados que arrojó son los siguientes:

### 5.4.1. Primera Revisión

#### 5.4.1.1. Análisis

Pregunta	Ponderación	Si	No	N/A	Calificación
1	5		✓		3
2	5	✓			3
3	5	✓			4
4	5			✓	0
5	5			✓	0
6	5			✓	0
7	5	✓			5
8	5	✓			5
9	5	✓			3
10	5		✓		3
11	5	✓			2
12	5		✓		0
13	5		✓		1
14	5	✓			4
15	5	✓			4
16	5	✓			5
17	5	✓			5
18	5	✓			5
19	5	✓			5
20	5	✓			4
21	5		✓		4
22	5	✓			2
23	5	✓			1
24	5		✓		2
25	5			✓	0
26	5		✓		1

27	5	✓			4
28	5	✓			4
29	5	✓			3
30	5	✓			3
31	5	✓			3
32	5	✓			3
33	5	✓			4
34	5	✓			4
35	5	✓			4
36	5	✓			4
37	5	✓			4
38	5		✓		0
39	5	✓			3
39	195	27	8	4	114

		Subtotal	Total
SI	27		
No	8	35	
N/A	4		39

	Ponderación	Estimado	Real
Preguntas	39	35	35
Valores	195	175	114
Calificación	10	10	6.514

### 5.4.1.2. Diseño

Pregunta	Ponderación	SI	No	N/A	Calificación
1	5	✓			3
2	5	✓			2
3	5	✓			5
4	5		✓		1
5	5		✓		1
6	5	✓			2
7	5	✓			3
8	5			✓	0
9	5	✓			5
10	5			✓	0
11	5	✓			3
12	5		✓		1
13	5		✓		1
14	5		✓		0
15	5	✓			4
16	5			✓	0
17	5	✓			5
18	5			✓	1
19	5			✓	1
20	5	✓			3
21	5	✓			3
22	5			✓	4
23	5	✓			4
24	5		✓		1
25	5	✓			3
26	5		✓		5
27	5		✓		5
28	5		✓		5
29	5	✓			5
30	5	✓			5
31	5	✓			0
32	5			✓	0
33	5	✓			3
34	5		✓		3
34	170	17	10	7	87

	Subtotal	Total
Si	17	
No	10	27
N/A	7	34

	Ponderación	Estimado	Real
Preguntas	34	27	27
Valores	170	135	87
Calificación	10	10	6.444

### 5.4.1.3. Diseño de sistema y programación

Pregunta	Ponderación	Si	No	N/A	Calificación
1	5	✓			4
2	5		✓		2
3	5			✓	0
4	5			✓	0
5	5			✓	0
6	5	✓			4
7	5			✓	0
8	5			✓	0
9	5	✓			4
10	5	✓			5
11	5	✓			2
12	5	✓			5
13	5	✓			5
14	5	✓			5
15	5	✓			5
16	5		✓		5
17	5	✓			5
18	5	✓			5
19	5	✓			5
20	5	✓			5
21	5	✓			5
22	5	✓			1
23	5	✓			5
24	5	✓			2
25	5		✓		5
26	5	✓			5
27.1	5	✓			5
27.2	5	✓			5
27.3	5	✓			5
27.4	5		✓		1
27.5	5	✓			4
27.6	5	✓			4
28	5		✓		5
29	5		✓		3
30	5	✓			4
31	5	✓			4
32	5	✓			5

33	5	✓			5
34	5	✓			5
35	5	✓			4
36	5	✓			5
37	5	✓			5
38	5	✓			5
39	5	✓			4
40	5	✓			5
41	5	✓			5
42	5	✓			5
43	5	✓			4
44	5		✓		5
45.1	5	✓			5
45.2	5	✓			5
45.3	5	✓			5
45.4	5	✓			5
45.5	5	✓			5
45.6	5	✓			5
45.7	5	✓			3
45.8.1	5	✓			5
45.8.2	5	✓			3
46	5	✓			3
47	5	✓			5
48	5	✓			5
49	5	✓			5
50	5	✓			5
51	5	✓			5
52	5	✓			5
66	325	53	7	5	265

	Subtotal	Total
Si	53	
No	7	60
N/A	5	65

	Ponderación	Estimado	Real
Preguntas	65	60	60
Valores	325	300	265
Calificación	10	10	8.833

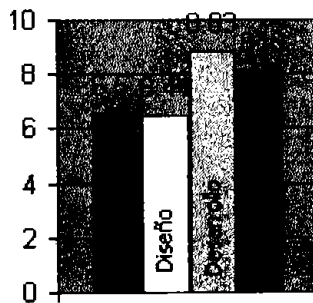
#### 5.4.1.4. Pruebas y mantenimiento

Pregunta	Ponderación	SI	No	N/A	Calificación
1	5	✓	*	*	4
2	5	*	*	✓	0
3	5	*	*	✓	0
4	5	✓	*	*	5
5	5	✓	*	*	3
6	5	✓	*	*	5
7	5	*	*	✓	0
8	5	✓	*	*	4
9	5	✓	*	*	5
10	5	✓	*	*	5
11	5	*	*	✓	0
12	5	✓	*	*	5
13	5	*	*	✓	0
14	5	✓	*	*	3
15	5	✓	*	*	3
16	5	✓	*	*	3
16	80	11	0	5	45

		Subtotal	Total
SI	11		
No	0	11	
N/A	5		16

	Ponderación	Estimado	Real
Preguntas	16	11	11
Valores	80	55	45
Calificación	10	10	8.181

#### 5.4.1.5. Análisis de resultados



**Promedio del equipo de trabajo: 7.47**

## 5.4.2. Segunda Revisión

### 5.4.2.1. Análisis

Pregunta	Ponderación	SI	No	N/A	Calificación
1	5	✓			3
2	5	✓			5
3	5	✓			4
4	5	✓			5
5	5			✓	5
6	5			✓	5
7	5	✓			5
8	5	✓			5
9	5	✓			5
10	5	✓			3
11	5	✓			3
12	5		✓		0
13	5	✓			3
14	5	✓			3
15	5	✓			4
16	5	✓			5
17	5	✓			5
18	5	✓			5
19	5	✓			5
20	5	✓			3
21	5	✓			3
22	5	✓			3
23	5	✓			3
24	5	✓			5
25	5	✓			5
26	5	✓			4
27	5	✓			3
28	5	✓			4
29	5	✓			4
30	5	✓			4
31	5	✓			4
32	5	✓			4
33	5	✓			4
34	5	✓			4
35	5	✓			4

36	5	✓			4
37	5	✓			4
38	5	✓			3
39	5	✓			3
39	195	36	1	2	153

		Subtotal	Total
SI	36		
No	1	37	
N/A	2		39

	Ponderación	Estimado	Real
Preguntas	39	37	37
Valores	195	185	153
Calificación	10	10	8.27

**5.4.2.2. Diseño**

Pregunta	Ponderación	SI	No	N/A	Calificación
1	5	✓			4
2	5	✓			4
3	5	✓			5
4	5	✓			3
5	5	✓			3
6	5	✓			5
7	5	✓			4
8	5	✓			5
9	5	✓			5
10	5	✓			4
11	5	✓			5
12	5	✓			4
13	5	✓			4
14	5	✓			4
15	5	✓			4
16	5	✓			4
17	5	✓			5
18	5	✓			5
19	5	✓			5
20	5	✓			4
21	5	✓			4
22	5	✓			4
23	5	✓			4
24	5	✓			3
25	5	✓			3
26	5		✓		5
27	5		✓		5
28	5		✓		5
29	5	✓			5
30	5	✓			5
31	5	✓			4
32	5	✓			3
33	5	✓			4
34	5			✓	0
<b>34</b>	<b>170</b>	<b>30</b>	<b>3</b>	<b>1</b>	<b>140</b>

		Subtotal	Total
SI	30		
No	3	33	
N/A	1		34

	Ponderación	Estimado	Real
Preguntas	34	33	33
Valores	170	165	140
Calificación	10	10	8,48



### 5.4.2.3. Diseño, Programación, Implementación y Pruebas

Pregunta	Ponderación	SI	No	N/A	Calificación
1	5	✓			3
2	5	✓			4
3	5	✓			3
4	5	✓			4
5	5	✓			4
6	5	✓			5
7	5	✓			5
8	5	✓			4
9	5	✓			4
10	5	✓			5
11	5	✓			5
12	5	✓			5
13	5	✓			4
14	5	✓			4
15	5	✓			5
16	5		✓		5
17	5	✓			5
18	5	✓			5
19	5	✓			5
20	5	✓			5
21	5	✓			5
22	5	✓			5
23	5	✓			5
24	5	✓			5
25	5		✓		5
26	5	✓			5
27.1	5	✓			5
27.2	5	✓			5
27.3	5	✓			5
27.4	5	✓			3
27.5	5	✓			5
27.6	5	✓			5
28	5	✓			5
29	5		✓		5
30	5	✓			5
31	5	✓			5

32	5	✓			5
33	5	✓			5
34	5	✓			5
35	5	✓			4
36	5	✓			5
37	5	✓			5
38	5	✓			5
39	5	✓			4
40	5	✓			5
41	5	✓			4
42	5	✓			4
43	5	✓			4
44	5	✓			5
45.1	5	✓			5
45.2	5	✓			5
45.3	5	✓			5
45.4	5	✓			5
45.5	5	✓			5
45.6	5	✓			5
45.7	5	✓			4
45.8.1	5	✓			4
45.8.2	5	✓			4
46	5	✓			4
47	5	✓			5
48	5	✓			5
49	5	✓			5
50	5	✓			5
51	5	✓			5
52	5	✓			5
53	5	✓			4
54	5	✓			4
55	5	✓			4
56	5	✓			5
57	5	✓			4
58	5	✓			5
59	5	✓			3
<b>73</b>	<b>365</b>	<b>70</b>	<b>3</b>	<b>0</b>	<b>332</b>

		Subtotal	Total
SI	70		
No	3	73	
N/A	0		73

	Ponderación	Estimado	Real
Preguntas	73	73	73
Valores	365	365	332
Calificación	10	10	9.09

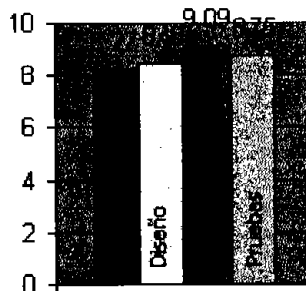
#### 5.4.2.4. Mantenimiento

Pregunta	Ponderación	SI	No	N/A	Calificación
1	5		✓		0
2	5	✓			5
3	5	✓			5
4	5	✓			5
5	5	✓			5
6	5	✓			5
7	5	✓			5
8	5	✓			5
8	40	7	1		35

		Subtotal	Total
SI	7		
No	1	8	
N/A	0		8

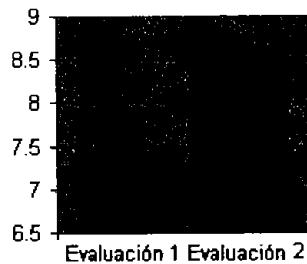
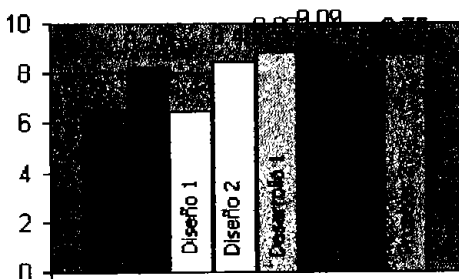
	Ponderación	Estimado	Real
Preguntas	8	8	8
Valores	40	40	35
Calificación	10	10	8.75

#### 5.4.2.5. Análisis de resultados



Promedio del equipo de trabajo: 8.65

#### 5.4.2.6. Análisis Comparativo



## **5.5. Modelos de Calidad**

### **5.5.1. Aplicación de Calidad**

Como se comentó en el apartado 1.8.3 este proceso consta de 4 herramientas básicas, las cuales son Análisis De La Voz Del Cliente, "Benchmarking", Deficiencias y Fortalezas Productivas y Análisis De Mercado, y éstas a su vez tienen más puntos.

En párrafos posteriores se expondrán los 3 primeros puntos, ya que el análisis de mercados es un herramienta orientada a empresas, se evoca más a aspectos selectivos de las ventas y del mercado de la organización, así como de proyecciones a futuro de ésta, es por ello que para los propósitos de este sistema resulta innecesario llevar acabo este estudio, además que se cuenta con sesiones regulares en las que se tiene contacto con los usuarios y se pueden detectar necesidades presentes y futuras.

#### **5.5.1.1. Aplicación Análisis de la Voz del Cliente**

##### **5.5.1.1.1. Etapas**

El proceso de medición de satisfacción al cliente cuenta con 5 etapas, las cuales son:

#### **1. Planeación del proyecto. Incluye:**

- A. Hay dos directores de proyecto, un encargado de análisis y planeación del proyecto, un encargado de desarrollo e implantación, dos diseñadores de Interfaz y un encargado de calidad.
- B. La manera en que se utilizarán los resultados es al integrarlos primeramente en el arranque del proyecto, es decir, hacer una encuesta para detectar las medidas de satisfacción del cliente según la ingeniería de software y sacar provecho de los puntos críticos para integrarlos en el sistema y así lograr una "comodidad" en el entorno del sistema para los usuarios, posteriormente se llevará a cabo un cuestionario de evaluación de usuario y sistema.
- C. Nuestros "clientes" son todas las personas que puedan llegar a hacer uso del sistema, a los cuales llamaremos usuarios. Siendo más concretos, estudiantes, profesores y docentes interesados en el área de estudio de la odontología. Los clientes son los usuarios del sistema.
- D. El desarrollar una lista de clientes y una subclasificación relevante no es requerido, el proyecto no lo requiere así, ya que nuestros usuarios ya están debidamente reconocidos y se contará con un sistema interno de identificación de usuarios.
- E. Las responsabilidades se asignan de la siguiente manera:
  - Los directores del proyecto se encargan de evaluar, autorizar los cuestionarios y el método de encuesta elegido, así como todas las etapas pertenecientes al análisis de la voz del cliente.

- El analista estará encargado de llevar a cabo el análisis completo del sistema del sistema que planea llevarse a cabo. El comisionado del diseño de sistema, programación, implementación y pruebas es el diseñador de sistema, los elementos del equipo de diseño de Interfaz se encargan de contestar, sugerir y si es pertinente evaluar los cuestionarios y el método de encuesta.
  - La encargada de calidad proporcionará los indicadores de beneficio a los clientes, los elementos que deberán ser evaluados durante todo el proceso, obtener los resultados del proceso de análisis de la voz del cliente, así como sugerir y evaluar los objetos del análisis.
- F. Involucrar a todo el equipo se logra a través de juntas semanales que se llevarán a cabo periódicamente hasta el día en que se ponga en funcionamiento el sistema, se tenga terminado este trabajo o la fecha que se acuerde por el equipo de trabajo.

## **2. Mediciones**

Las mediciones se obtienen en base a los resultados de los cuestionarios. La medición planifica la puesta en práctica de las formas directas de investigación, a través de:

- A. Se optó por el método de encuesta del cuestionario, pero observando los Factores de calidad internos y externos del software. Internamente se llevó a cabo un cuestionario de evaluación de la calidad para cada grupo de trabajo. De forma externa, la encuesta se llevará a cabo a través de cuestionarios, se tomará un grupo de posibles usuarios, en este caso interesados en el sistema, para aplicar el cuestionario una vez que ya esté funcionando el sistema.
- B. El cuestionario se implanta en base a los requerimientos de calidad que se propusieron en cada etapa del sistema, así como lineamientos de interfaz e ingeniería de software.
- C. En base a cuestionarios de auditoria de software se llevó un análisis de los puntos de observancia relevantes que fueron evaluados en ésta y se retomaron para evaluación en el sistema.
- D. La muestra en este caso fue el propio equipo de trabajo, que además de contestar el cuestionario, se les requirió que analizaran la porción que les había tocado contestar para ver si consideraban que faltaba o algo se estaba quedando fuera. Para la muestra posterior a la implantación se tomarán de 5 a 8 usuarios y se presentará primero la prueba del cuestionario a los directores del proyecto.
- E. Al desarrollar el cuestionario incluí instrucciones por cada etapa en que se dividió el cuestionario para facilitar así el contestarlo (ver anexo 2 y 3) y se aclararon dudas que surgieron de el cuestionario, además que se revisaron varias veces con el encuestado para mayor precisión.

- F. Las encuestas se condujeron conforme se contestaba el cuestionario y surgían dudas, se aclaraban, además se realizaron diversas revisiones antes de pasar a la primera versión del cuestionario, ya que se planea hacer una versión más, antes de que salga el producto.

### **3. Diagnóstico**

Se obtiene a partir de los resultados de los cuestionarios analiza el rendimiento de la empresa para brindar satisfacción al cliente, y debe considerar ciertos puntos. Estos puntos son:

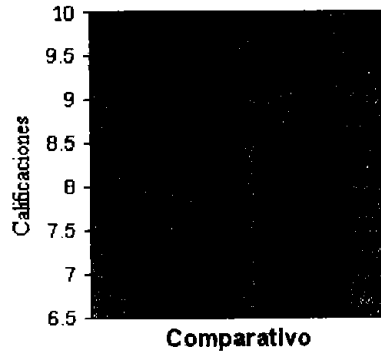
- A. Los cuestionarios se aplicaron a los integrantes del proyecto, además de a un grupo controlado de usuarios, pero todas estas actividades en mi presencia y bajo supervisión y revisión de los directores del proyecto.
- B. El cuestionario que se aplicó a los usuarios evalúa criterios de ingeniería de software en cuanto a interfaces de usuario.
- C. Los cuestionarios ya descritos llevarán casillas de calificación y el rango será del número uno al cinco, y se calificará según se cumplan los puntos en cuestión, el número uno es el valor más bajo y el cinco el más alto.
- D. Analizar medidas directas e indirectas para la satisfacción del usuario. La IS nos permite tener pautas de dirección de diseño, de pantallas, de interfaz, y de una serie de cosas más, esto permite que se reduzcan en gran número los esfuerzos para lograr la satisfacción del cliente.
- E. Se realizó un análisis por parte del quipo de diseño como una investigación informativa, lo cual arroja:
  - 1. Ubicación del usuario potencial (maestros y alumnos de la carrera de Odontología)
  - 2. Análisis del entorno sociocultural del usuario
  - 3. Detección de necesidades y expectativas del usuario sobre el sistema e implementación de soluciones para estas
- F. Cada integrante del equipo, desempeña un rol, ya sea de análisis, diseño, etcétera, a cada uno se les proporcionó un documento en el que se describían sus responsabilidades precisas en cuanto a calidad (Ver Anexo 1), y se espera que a través del seguimiento que se le de a estas medidas se pueda adquirir el compromiso necesario para lograr el nivel de calidad deseado y así exceder la satisfacción del cliente.

### **4. Evaluación de las actividades**

Este punto es posterior a la implementación de las mejoras tomadas de los cuestionarios. La evaluación de los resultados del estudio implica:

- A. El análisis preliminar de los datos en un formato que los haga fáciles de entender e interpretar se encuentra a continuación comparando el primer cuestionario de evaluación con el segundo, la mejora reflejada de la aplicación de los principios y factores de calidad se puede ver aplicada en la siguiente grafica:

B. El análisis de la Información cualitativa, el cual determina los atributos preliminares de rendimiento, analizando las respuestas de los cuestionarios. El primer cuestionario interno arrojó los siguientes resultados de rendimiento como equipo de 7.47, calificación ni buena ni mala para el comienzo de un proyecto, aunque es deseable que hubiese sido mejor el desempeño. Pero en el segundo se obtuvo una mejora de 1.18, lo que nos da un promedio de 8.65.



## 5. Mejora productiva continua

Esto significa una retroalimentación, no cerrar este ciclo, sino continuar con él en lo sucesivo. La mejora productiva continua es indispensable para respaldar y exceder las expectativas en calidad del usuario. El plus que le vamos a dar al usuario es precisamente nuestra aplicación, ya que hasta donde estuvimos investigando no hay un sitio tan especializado y precisamente para albergar documentos de ese tipo. Y se pretende que con el seguimiento de los principios de la Ingeniería de Software en complemento con los puntos que maneja ISO se logre un mejor nivel de calidad.

### 5.5.1.2. Aplicación del Benchmarking

La aplicación de esta herramienta se da en la etapa de diseño, como una forma de ver las mejores alternativas en el mercado en cuanto a propuestas gráficas y de distribución del contenido y texto, así como para ver las tendencias que hay en aplicaciones en el momento de desarrollo de la etapa de diseño.

#### 5.5.1.2.1. Indicadores de Desempeño y Facilitadores

- **Indicadores de desempeño.** En este caso los indicadores de desempeño son todos los factores que se observan en una aplicación como son:
  - Diseño de pantallas (tipo de letra, imágenes, Colores, etcétera)
  - Presentaciones consistentes
  - Movimientos entre pantallas (desplazamiento, solicitud de mayor detalle, Dialogo de pantallas, etcétera)
  - Facilidad de uso de las aplicaciones
  - Navegación

- Carga rápida
- Entre otros
- **Facilitadores.** Los facilitadores del *Benchmarking* son:
  - Definir rutas de navegación
  - Las opciones de navegación deben ser obvias
  - Establecer mapas de sitio
  - Controles de acceso para los diferentes usuarios.
  - Definir la mecánica de los enlaces (enlaces basados en texto, íconos, botones, etcétera).
    - ◎ Elegir los enlaces de navegación acordes al contenido.
    - ◎ Establecer las conexiones y ayudas adecuadas (íconos y enlaces gráficos adecuados)
    - ◎ Utilizar un color que indique los enlaces de texto.
    - ◎ Indicar que se ha elegido una opción de navegación.
    - ◎ Indicar los enlaces por los que se ha navegado.
  - Mantener pantallas sencillas
  - Manejar y mantener presentación consistente
  - Facilitar los movimientos de usuario entre pantallas (desplazamiento, solicitud de mayor detalle, Dialogo de pantallas, etcétera)
  - Crear pantallas atractivas (manejando tipo de letra, imágenes, Colores, etcétera)
  - Familiaridad del usuario con otros sistemas ya utilizados
  - Consistencia. Esto es igualdad en los componentes.
  - Mínima sorpresa. Que no exista un cambio muy radical en los componentes o versiones. Que el comportamiento del sistema debe ser predecible.
  - Guiar al usuario. Proporcionar ayudas. Guías de usuario. Ayuda, arreglos y su manejo.
  - Diversidad de usuarios. Observar las características de todos los posibles usuarios y tratar de incorporarlas. Esto es proporcionar características adecuadas a cada tipo de usuario.
  - Facilidad de uso (selección de menús, listas, llenado de formularios)
  - Evitar símbolos de construcción
  - Evitar links rotos o inservibles
  - Evitar que el usuario recorra la pantalla
  - El diseño no deberá depender de las funciones del navegador
  - Documentación de errores, para posterior corrección antes de la liberación del sistema.
  - Adaptación. Modelo de trabajo a adoptar de acuerdo al usuario.
  - Limitar el número de colores utilizados y ser conservador al momento de utilizarlos

- Utilizar un cambio de color para mostrar el cambio en el estado del sistema
- Utilizar el código de colores predefinido en una forma consistente
- Ser cuidadosos al utilizar juegos de colores
- Manejar un grado razonable de retroalimentación para Incrementar el grado de confianza

Los recursos con los que se cuenta para llevar a cabo la investigación son los sistemas que se tomaron como referencia, los equipos de cómputo con los que se cuenta para acceder a internet y poder tener ingreso a ellos.

#### **5.5.1.2.2. Benchmarking en Cinco Etapas**

- 1. Planificar.** Determinar a que se le va aplicar el "*benchmarking*". El "*benchmarking*" va a ser aplicado a un sistema, y concretamente a la etapa de diseño. Los sistemas que van ser estudiados son de dos tipos:
  1. Sistemas comerciales
  2. Sistemas educativos
- 2. Actuar.** Formar un equipo de "*Benchmarking*". Está conformado por:
  - Los líderes de proyecto, que se encargan de evaluar, autorizar, entre otras tareas, los objetos del "*benchmarking*".
  - Los elementos del equipo de diseño, quienes se encargan de elegir, sugerir y evaluar los objetos del "*benchmarking*".
  - La encargada de calidad, quien proporcionará los indicadores de "*benchmarking*", elementos que deberán ser evaluados durante todo el proceso, obtener los facilitadores del "*benchmarking*", así como sugerir y evaluar los objetos del "*benchmarking*".
- 3. Corregir.** Identificar los socios del "*benchmarking*". Nuestras fuentes serán, dada la naturaleza del proyecto, cien por ciento de internet, y únicamente obtenidas de observación e identificación de las mejores prácticas industriales y organizativas.
- 4. Estandarizar.** Recopilar y analizar la información de "*Benchmarking*". Deberán ser siempre observados los indicadores de "*benchmarking*" previamente establecidos en cada uno de los sistemas que se hagan, sin perder de vista ninguno de ellos, adecuándolos a las diferencias según el tipo de cada sistema ya sea tratándose de sistemas educativos o comerciales.
- 5. Retroalimentación.** Adaptar, mejorar y aplicar los Facilitadores. Esto es, tomar las ventajas y adecuarlas a nuestras necesidades, detectar los errores de los sistemas consultados para evitarlos y mejorar lo que podamos hacer a través de los facilitadores.

#### **5.5.1.2.3. Tipos de Benchmarking**

En el presente trabajo se tomaron en cuenta dos tipos de "*Benchmarking*": el competitivo y el funcional que a continuación se detallan.



## **a) Competitivo**

### **1. Sistemas Educativos**

En el caso de los sistemas educativos el nivel de uso de "benchmarking" es el de las mejores prácticas en la industria o de competidores, ya que se toma en cuenta el ámbito de operación que tienen los sistemas. Se tomó como referencia a diversas instituciones educativas debido a que el fin del sistema es educativo, y se formó explícitamente para ser una herramienta para la educación. Los puntos que se tomaron como ejemplo, por destacar, y variar mucho en los indicadores de desempeño son los siguientes:

- Estructura. La distribución de la página es favorable para su presentación al usuario, además de que la presentación en pantalla no está desorganizada o tiene sobrecargada paneles, imágenes o banners, sino que puntualiza su contenido gráfico para su exposición organizada en el navegador.
- Manejo información. La información fue distribuida de tal forma que no se vea sobrecargada la página por la cantidad de texto, sino que especifica su contenido a estrictamente el necesario, sin llegar a atestar el contenido de cada pantalla en cuanto a texto.
- La tipografía es sencilla, no rebuscada. Lo que proporciona, una muy buena forma de presentar un página, por que nos permite tener más limpieza en las líneas de texto, y mejor presentación al usuario.
- Diseño y propuesta gráfica. Su presentación es bastante agradable visualmente, los gráficos son acordes al contexto que se maneja, las animaciones son pocas, minúsculas, y discretas
- Muy buen manejo de color. Los colores son apropiados para el tema del sistema o sistemas, no son colores chillantes o que lastimen la vista, sino que en general proporcionan un ambiente agradable y no agresivo para quienes navegan por esos sistemas. Los colores blanco y azul son más aplicables a la medicina.
- Todos los sistemas educativos tienen el menú en la parte superior, lo cual facilita la navegación y aumenta el grado de amigabilidad que el sistema puede proporcionar a los usuarios.
- Conocimientos de educación bien aplicados. Ya que es visiblemente notorio que hicieron el desarrollo de su sistema siguiendo algunas bases de diseño.
- Estructura de la información más clara. Esto se representa en que la información tiene un orden cronológico, o bien lógico para su acomodamiento dentro de las páginas y la apertura de éstas y el desplazamiento entre ellas.
- Navegación sencilla. Los menús se presentan en lugares estratégicos, de forma que el usuario no se pierda al desplazarse a través de las diversas páginas de contenido, además de un fácil desplazamiento, en ocasiones solicitud de mayor detalle o diálogos en pantalla.
- Así como otros factores particulares de cada página como son:
  - Menú horizontal, parte superior

- o Menú con 5-7 botones máximo
- o No cuentan con buscador en la página de inicio
- o Página de inicio sin "frames"
- o Ligas de botones con palabra completa

También destaca una característica importante de los sistemas educativos, y es la forma en que está organizada la información dentro de la página que permite identificar una estructura bastante definida y organizada, así como la estructura general de la misma.

Páginas consultadas para el "Benchmarking" Competitivo:

<http://www.nobelllearning.com/nobel/viewpage.cfm?pageID=2>

<http://learningcommons.evergreen.edu/>

<http://yarranet.net.au/e-learning/>

<http://www.cwd.dk/inspiration.asp?mode=4&siteid=3648>

<http://www.dryvescohen.com/>

<http://www.oeil-dentaire.com/>

<http://www.fda-france.com/>

## **b) Funcional**

### **2. Sistemas Comerciales**

El nivel de uso de "benchmarking" es el de mejores prácticas funcionales ya que se observa la aplicación en sistemas de diferentes ámbitos y ciertos factores claves. Resulta conveniente saber cuales son las tendencias Web para diseño de interfaces, manejo de colores, y todas las nuevas formas de atraer la atención del cibernauta, los factores que se tomaron en cuenta o propiamente llamados indicadores de desempeño se detallan a continuación:

- Buscador. En todos los sistemas comerciales se presenta un buscador en una de las partes principales de la página, generalmente es en la parte superior izquierda, inmediatamente después del menú o "banner" principal.
- Sobrecarga de información. No se cumple con una pantalla sencilla, hay demasiada carga de texto en pantalla, no se toma en cuenta que el contenido esté muy amontonado, solo se trata de poner toda la información posible dentro de la página para que el mensaje llegue al navegante, sin importar que éste pueda perderse en tanto contenido o resulte pesado para él leer todo, cosa que en ocasiones produce que la información sea desdeñada fácilmente.
- Barra de menú muy extensa. Las ligas presentadas en la barra principal, son muchas, por lo que no permite que resulte atractivo a los usuarios, o como se dicta por los lineamientos para diseño de pantallas, esto es, que no mantienen una pantalla sencilla. Está ubicada en la parte superior central.
- Colores que manejan. Manejan una paleta de colores muy extensa para su estructura general, no hacen juegos de colores armoniosos, o si se aplica esto, no concuerda con el resto de los colores manejados en la página.

- Logotipo y nombre de sistema van arriba y junto al logotipo. Este es un punto relevante, ya que nos permite situar elementos importantes en este lugar, y esto se basa en el hecho de que en la parte superior derecha es lo que como usuario se ve primero, también en ocasiones va del lado izquierdo superior, pero por observación la mayoría lo ubican arriba a la derecha.
- Mucha carga visual. Los factores que influyen en esto son el tipo de letra, las imágenes, y el color, que son manejados inadecuadamente, amontonando unos con otros, sin respetar un espacio para cada cosa y sin importar su distribución.
- Colocación de los elementos. Este punto importante, ya que se observó que los sistemas comerciales tienen elementos comunes como son el buscador en la parte superior izquierda inmediatamente después del menú principal que se encuentra en la parte superior central media después de algunos "banners" o sin ellos, así como el manejo de submenús a los costados o pestañas para las diversas páginas que componen el sistema.
- Distribución del sitio. No manejan pantallas sencillas, no tienen una presentación consistente, en ocasiones no facilitan los movimientos de usuario entre pantallas por el amontonamiento, imágenes y "banners" por todos lados, y su estructura no está bien definida.
- Así como otros factores particulares de cada página como son:
  - Menú vertical
  - Menú sin límite de botones
  - Ubicaciones del buscador (parte superior)
  - Saturación de información
  - Arreglo de la página de inicio en 3 o 4 frames
  - Sustitución de palabras por íconos

Páginas consultadas para el "Benchmarking" Funcional:

<http://www.t1msn.com.mx/>

<http://www.google.com.mx/>

<http://flashxl.com>

<http://cwd.dk>

Se pudo observar que las estructuras de los sistemas educativos y comerciales eran notablemente diferentes, tanto en estructura, distribución, colores y diseño en general.

#### **5.5.1.2.4. Resultados del Benchmarking**

Del análisis de "benchmarking" se desprende lo siguiente:

**a) Propuesta Gráfica.** La propuesta en la interfaz gráfica desarrollada pretende transmitir a través del lenguaje gráfico los valores de los siguientes elementos:

- Comunidades de aprendizaje
- Uso de la tecnología (internet)

- Sistema de Ciencias de la salud
- Profesores de Iztacala (intercambio académico y docencia)
- Elementos de comunicación humana

**b) Logotipo.** Considerando estos elementos diseñamos un logotipo en el que se representará por un lado la comunicación humana en un sentido amplio incluyendo la parte académica, y al mismo tiempo el uso de la tecnología. Es por eso que decidimos utilizar los rostros de perfil y en el fondo un CD. Este logotipo se diseñó en tercera dimensión para darle un carácter original y sobre todo actual, para uso digital.

Pensando en los requerimientos para impresión del logotipo diseñamos (con base en la propuesta de 3D) una silueta versión en 2D del mismo. Esta versión la manejamos en color.

**c) Color.** El color es un elemento gráfico básico para definir o fortalecer el mensaje que queremos transmitir.

En este caso se decidió utilizar 3 colores básicos en la interfaz gráfica, dos tonos de azul para uso general y un tono de verde para destacar. Lo que se aplicará de la siguiente forma:

Los botones, zonas activas y enlaces se identifican mediante el manejo de colores y animación.

- Los botones principales (barra de navegación superior) cuentan con distintivo que cambia a color rojo cuando se pasa el puntero por encima ("rollover").
- Los enlaces de texto idealmente, el color del enlace será azul claro (#0066ff).
- Cuando se coloque el puntero arriba, el texto cambiará a negritas (mismo estilo, mismo color).
- Los botones de la barra de navegación tienen aspecto tridimensional (relieve) para referir que en ese lugar debe existir un clic.

Se usó la paleta Web con los siguientes tonos:

- Azul principal 0099FF
- Azul claro 66CCFF
- Verde 99FFFF

Empleando la psicología y el significado del color, el azul representa paz, tranquilidad, suavidad, pureza, limpieza y entendimiento. Y es un color muy usado en el área de la medicina y la salud, y se liga con la aplicación directa del sistema. Por otra parte el color verde que utilizamos es refrescante y revitalizante.

**d) Tipografía.** Una vez lograda la abstracción del logotipo buscamos una tipografía adecuada para integrarla de manera funcional. Utilizando la fuente "humanistic" 521 BT en los estilos regular y "bold". Y la fuente futura MD BT.

### 5.5.1.3. Aplicación de Deficiencias y Fortalezas Productivas

Para la realización de esta tarea se necesita llevar a cabo la técnica de las 5 M's, la cual se encuentra en el Anexo 5. Los resultados que este ejercicio arrojó son los siguientes:

#### 5.5.1.3.1. Análisis de la Aplicación de la Técnica de las 5 M's

A continuación se presenta una lista de los principales problemas detectados a través de esta herramienta y las posibles soluciones para los mismos.

Falta de iniciativa. Se puede remediar esto sugiriendo apoyos bibliográficos, o de cualquier tipo que sirvan para profundizar en el conocimiento del tema.

Incumplimiento de tiempos de trabajo. Es conveniente fijar un calendario de trabajo con actividades y responsabilidades ligadas a calificaciones asignadas a cada entrega, para así saber que sucederá en caso de incumplimiento.

Elección de metodologías y herramientas desconocidas. Aunque por un lado elegir las mejores metodologías es bueno para el proyecto, ocasiona un atraso en tiempo al aprenderlas, por lo que se debe fijar desde un principio el uso de una metodología y tenerlo contemplado en el calendario de actividades, especificar cuando estas deben aprenderse y elegir ayudas para que todo el equipo conozca de lo que se está hablando o en dado caso dar una explicación a todo el equipo.

Discontinuidad en avances del proyecto. La discontinuidad generalmente se da por el desconocimiento del tema, pero esto se puede arreglar con libros o si alguno de los integrantes del equipo conoce el tema explique al resto del equipo, o a las personas necesarias para lograr una mejor coordinación y un nivel de conocimiento adecuado.

Comunicación incorrecta entre el equipo, deficiente uso de los medios electrónicos para trabajo en equipo, ausentismo en las reuniones de trabajo. Se dio por falta de interés y compromiso con el resto del equipo, lo que se puede remediar con reflejarlo en las calificaciones del calendario de actividades. También se puede remediar usando esta herramienta para detectar qué es lo que está fallando.

Avances lentos en entregas para el proyecto. Se avanzaba lento por que no se conocían las metodologías elegidas, pero una vez que se conocieron se pudieron ver avances sustanciales, esto se puede aminorar si se pone una fecha límite para dominar las herramientas o metodologías.

Discordancia entre la decisión de uso de una metodología. Se presentó entre el uso de una visión estructurada u orientada a objetos, ya que algunos elementos no estaban familiarizados con la segunda, esto se puede poner a consideración del jefe de proyecto, ya que lo más conveniente será a lo que se le pueda dar un seguimiento adecuado con los recursos que se cuenten.

Falta de claridad en la presentación de la investigación, dependencia no justificada hacia los demás integrantes del equipo, falta de claridad del papel a desempeñar. Esto se puede resolver fijando metas y herramientas específicas independientemente de la persona que vaya a desempeñar la tarea, así se tendrá claridad en el papel a desempeñar, y los resultados esperados.

Duplicidad de trabajo. Cuando no se tiene una comunicación fluida pasan este tipo de cosas, se puede solucionar implementando un inventario del material y recursos con que se cuenta, y distribuirlo a todo el equipo, para que se conozca lo que se tiene y qué falta.

Cargas de trabajo mal distribuidas. Si se fijan responsabilidades y tiempos desde un principio, no será necesario lidiar con esto, y si se llega a dar, se deberá ver reflejado en la calendarización de actividades con su respectiva calificación, lo cual proporcionará un estimado del desempeño de cada integrante.

Falta de tiempo y horarios incompatibles. Se puede evitar poniendo horarios de trabajo establecidos, y se deberá verificar el cumplimiento, de no ser así, se verá reflejado en el calendario de actividades.

Falta de Software necesario. Esto se dio en cuanto a herramientas de diseño, ya que se contaba con unas, pero se necesitaron otras que se desconocían, posteriormente se adquirieron gratuitamente de Internet. Se espera que de la misma manera se adquieran las necesarias en un momento dado.

A continuación se explican algunos factores a favor detectados, así como la forma para difundirlos de ser posible.

Conocimiento del papel a desempeñar, conocimiento del problema a solucionar, buen planteamiento de la metodología y buena comunicación entre algunos integrantes del equipo. Se puede aprovechar esto si se hace extensivo al resto del equipo, si se logra que la comprensión del tema, se ganará tiempo, y se empleará en otras tareas. Esto se puede lograr a través de presentar en un formato las actividades específicas que se desempeñan y cómo se están logrando.

Iniciativa en el aprendizaje de nuevas metodologías, creatividad aplicada en el proyecto, iniciativa en la recopilación de apoyos, manejo excelente de herramientas utilizadas, capacidades adicionales por parte de algunos integrantes del equipo y obtención de software faltante. Esto es una gran ventaja, se puede aprovechar más asignando tareas similares a estos elementos, y motivar al resto del equipo para que también lo hagan, además de que se aproveche enseñando lo aprendido al resto del equipo.

Se contaba con lo mejor en cuanto a material de cómputo, medios y periféricos necesarios. Esto debe aprovecharse al máximo sin derrochar recursos, realizando una descripción detallada de lo que se necesita y en base a ello se analice lo que se tiene y se elige lo que se ajuste más a las necesidades presentadas.

Beca otorgada para realización de actividades. Sin duda un estímulo económico es bastante estimulante y se debe calendarizar también de manera que se entregue la beca de acuerdo a los avances, o a fechas que concuerden con las entregas del proyecto.

Utilización de herramientas de punta. Aunque por un lado elegir las mejores metodologías es bueno para el proyecto, ocasiona un atraso en tiempo al aprenderlas, por lo que se debe fijar desde un principio el uso de una metodología y tenerlo contemplado en el calendario de actividades, especificar para cuando estas deben aprenderse y elegir ayudas para que todo el equipo conozca de lo que se está hablando o en dado caso dar una explicación a todo el equipo.

Buena comunicación con el equipo. Esto debe ir calificado en el calendario de actividades como un factor más a evaluar.

Buena elección en las herramientas elegidas. Esto se logra a través de un análisis de las herramientas disponibles, destacando aquellas que son más apropiadas para lo que habrá de desarrollarse.

Eficiente uso de los medios electrónicos para trabajo en equipo. Es indispensable conocer y utilizar estas herramientas, para así coordinar las actividades propias del proyecto.

Buena investigación y aplicación de metodologías, visión clara de la implementación y buena aplicación de técnicas de diseño. Cuando se conoce a fondo las herramientas o metodologías se sabe que habrá de lograrse un buen trabajo, pero esto debe llevarse a cabo paulatinamente, de forma que se haga en todas las áreas del proyecto, para tener un buen fundamento.

Avances en tiempo y forma. Esto debe lograrse en todo, dentro de lo posible, para así garantizar la entrega del proyecto a tiempo, lo cual también debe lograrse en cada área del proyecto.

## **5.6. Aplicación de las Características de Calidad de Programas Educativos y de Información.**

Para poder aplicar estos principios, se realiza un cuestionario que se aplicará a los posibles usuarios del sistema, para consultar esto se puede recurrir al Anexo 3. El análisis de los resultados de la aplicación son los siguientes:

### **5.6.1. Análisis de las características externas de calidad.**

Para poder llevar a cabo este cuestionario se tomó una muestra de 8 profesores, se había descrito que deberían ser de 5 a 8 participantes, así que se optó por 8 por conveniencia. El cuestionario fue aplicado directamente a los profesores que usarán el sistema en la FES Iztacala, y lo calificaron una vez que ya habían hecho uso del sistema.

### 5.6.1.1. Cuestionario usuarios

	Si	No	N/A	Suma Calificación	Calificación Final
Cuestionario 1	35	14	0	189	7.714
Cuestionario 2	32	17	0	187	7.632
Cuestionario 3	39	10	0	226	9.224
Cuestionario 4	26	19	3	198	8.081
Cuestionario 5	38	10	1	201	8.204
Cuestionario 6	33	16	0	205	8.367
Cuestionario 7	25	23	1	181	7.387
Cuestionario 8	35	14	1	214	8.734
<b>TOTAL</b>					<b>8.167</b>

En comparación con los cuestionarios de calidad interna, se puede observar que con la aplicación de los factores de calidad se logra una calificación regularmente buena, pero se pueden mejorar factores como:

- Mayor grado de interacción con el usuario
- Mas indicaciones de estados y avance de sistema



# CAPÍTULO 6: APLICACIÓN DE SEGURIDAD

## 6.1. Cuestiones Generales

En el presente capítulo se hará la presentación de las herramientas empleadas para la implementación de la seguridad en el proyecto descrito en la parte de Contexto de Desarrollo del Proyecto. Lo que se pretende es implementar en el sistema llamado "Sistema de Comunidades de Aprendizaje Iztacala", que forma parte de los proyectos desarrollados por el Laboratorio de Interacción Humano - Máquina del CCADET de la UNAM, las herramientas de seguridad necesarias.

Para la evaluación de riesgos se pretende usar el método llamado OCTAVE<sup>39</sup> de la Universidad Carnegie Mellon en Pittsburgh, EE.UU. Se observarán los parámetros indicados en los manuales oficiales de Red Hat Linux, así mismo se seguirán los parámetros de seguridad implementados en DGSCA<sup>40</sup> y algunas recomendaciones de fuentes bibliográficas y otros organismos más.

Con el fin de poder proporcionar los atributos de un sistema seguro, se optó por implantar el sistema bajo una plataforma Linux con Red Hat, puesto que se acopla más a las necesidades del equipo de trabajo, como se presenta en la siguiente comparación de Windows con Linux:

	<b>Windows</b>	<b>Linux</b>
<b>Estabilidad</b>	En la GUI(interfaz gráfico de usuario) se tiene una desventaja, ya que esta forma parte del núcleo de Windows, y en caso de bloqueo de la GUI, requiere de reinicio.	Dado que el GUI no forma parte del núcleo de Linux, en caso de bloqueo si se "mata" el servidor X se arregla el problema.
<b>Uso de recursos, rendimiento</b>	El consumo de recursos de Windows aumenta conforme evolucionan las versiones de sus sistemas operativos, lo que hace que cada vez se exijan mayores recursos.	El uso de recursos en Linux es mucho menor que en las distintas versiones de Windows, incluso en equipos viejos.
<b>Facilidad de uso</b>	Cuando se trata de configuración de opciones avanzadas, en Windows cada vez se vuelve más complicado, ya que no se da acceso franco a estos recursos.	En Linux se dota al sistema operativo de mayores posibilidades de configuración, y el tiempo de aprendizaje es mayor.
<b>Acceso al código</b>	El código no está a disposición de los usuarios del sistema, por lo que no se puede acceder a él de forma directa.	Cuenta con código abierto, da la posibilidad y permiso de modificación y distribución, una mejora y solución de errores más rápida
<b>Precio</b>	Windows es un producto comercial. El soporte técnico del software ya está incluido en el precio de Windows.	Linux es un sistema operativo de distribución libre. Si se quiere adquirir soporte técnico del software en hay que pagar.
<b>Licencias</b>	Si se quiere instalar Windows en más de una máquina hay que pagar licencias de uso.	No es necesario pagar licencia si se quiere instalar Linux en más de una máquina

<sup>39</sup> Operationally Critical Threat, Asset, and Vulnerability Evaluation

<sup>40</sup> Dirección General de Servicios de Cómputo Académico

Observando las características anteriores, se eligió Red Hat Linux, principalmente por la estabilidad que proporciona y el precio de optar por este sistema, pues puede garantizar que los servicios estarán disponibles para el usuario y que los recursos no serán desaprovechados, además que permite implementar servicios de manera libre (no licencias), y cuenta con herramientas de desarrollo gratuitas, en su mayoría.

**Postura Ante Riesgos.** La postura adoptada para el sistema es la prudente, donde se proporciona únicamente los servicios necesarios, para que funcione el sistema al 100%. "Lo que no está expresamente permitido está prohibido".

## 6.2. Amenazas y Riesgos

Como anteriormente se menciona en esta tesis, para esta tarea se eligió el método OCTAVE y se implementó junto con el equipo de desarrollo del sistema, teniendo siempre en cuenta los principios de seguridad descritos en los capítulos de seguridad. A continuación se da una explicación de los resultados de este método. Los árboles de amenazas aplicados para detección de amenazas se encuentran en anexo 6.

### 6.2.1. Aplicación de OCTAVE

Este proceso cuenta con tres principales pasos que se describen a continuación:

- 1) Vista Organizacional. Consta de las siguientes etapas:
  - a) Recursos.
    - Instalaciones. Esto comprende los equipos que forman la red directamente relacionada con sistema, y todos los componentes que la conforman (impresora, escáner, cámaras, reguladores, etcétera). Especialmente el servidor del sistema.
    - Medios de almacenamiento. Todos los discos, CD's, y cualquier unidad extraíble que contenga información a cerca del sistema en cuestión.
    - Archivos. Incluye cualquier registro relacionado con el sistema, sin importar el contenido, pueden ser archivos de contenidos, bases de datos, páginas de internet, etcétera. En este caso los archivos críticos son los que forman parte constituyente del sistema.
    - Respaldos. Deben llevarse a cabo periódicamente respaldos de cada uno de los archivos del sistema realizado, la periodicidad de los mismos depende de la velocidad de los cambios generados en el sistema.
  - b) Amenazas. Estas se pueden dar de dos formas de acceso: físicamente o vía red. En los árboles de amenaza del anexo 6 se ve detalladamente que debe hacerse en caso de que cualquier ataque suceda con cualquiera de los siguientes recursos:
    - Información
    - Sistema
    - Servicios y aplicaciones
  - c) Prácticas actuales. Como el sistema acaba de ser puesto en marcha bajo esta nueva plataforma, se ha tomado en cuenta medidas mínimas de seguridad, pero se trabaja en ello para dejar el sistema con buena seguridad.

- d) Vulnerabilidades de la Organización. Las posibles vulnerabilidades que se pueden tener en ataques al sistema y pueden originar pérdida, modificación, interceptación, robo, publicación, interrupción, falsificación o suplantación, las instalaciones del servidor permanecen seguras, así que la preocupación principal, son los ataques vía red.
  - e) Requisitos de Seguridad. Se necesita seguridad lógica para el administrador de sistema, en cuentas, archivos, y en el sistema de archivos.
- 2) Vista tecnológica. Se necesita seguridad física para el servidor, y periféricos anexos a él.
- f) Componentes clave.
    - Equipo servidor.
    - Conexiones de red.
    - Regulador.
  - g) Vulnerabilidades tecnológicas
    - Descompostura del equipo. Prever esto asegurando la información contenida en él a través de respaldos de sistema.
    - Fallo en el enlace. Asegurar en lo posible la infraestructura de red como cables y equipos activos, lo cual puede hacerse colocando el equipo en un lugar donde no se pueda acceder fácilmente.
    - Fallo de provisión de energía del regulador. Procurar tener instalaciones adecuadas de electricidad, para evitar cortos circuitos.
  - h) Estrategia y desarrollo de un plan. Para ello se encuentran las políticas de seguridad que se detalla en la siguiente sección.
    - Riesgos. Es un evento que tiene un impacto, los cuales se determinan en los árboles de amenazas detallados en el anexo 6.
    - Estrategia de Protección. La protección será de acuerdo a las medidas de seguridad que se explicarán en secciones subsecuentes.
    - Planes de mitigación. Deberán seguirse las medidas de seguridad que se explicarán en secciones posteriores.

### **6.3. Políticas de Seguridad**

Después de detectadas las posibles acciones que pudiesen atacar al sistema, se desarrollaron las políticas de seguridad, las cuales se presentan a continuación:

#### **Creación de cuentas nuevas.**

- Crear cuentas única y exclusivamente para usuarios legítimos y autorizados por el líder del proyecto o encargado del proyecto.

- Proveer única y exclusivamente del espacio y recursos primordialmente indispensables para el usuario. De ser posible, asignar una cuota fija y estándar a todos los usuarios en su directorio *home*.
- Dar a los usuarios la cantidad mínima de privilegios que ellos necesitan.
- Estar enterados de cuándo y dónde se dan conectan al sistema los usuarios y tomar nota de ello.
- Dar de baja cuentas inactivas y que éstas sean detectadas automáticamente por algún mecanismo.
- Deberá contarse con implementación de un mecanismo de autenticación en todos los equipos activos.
- Está prohibida la creación de cuentas grupales bajo ningún motivo.

### **Seguridad del password**

- El tamaño del password debe ser mayor de 6 caracteres
- El password debe formarse a partir de la combinación de letras, números y caracteres especiales, sin exceptuar uno de los tres.
- Considerar el uso de una frase que contenga un espacio en blanco.
- El password no debe ser igual al nombre de usuario.
- No debe ser una palabra del diccionario.
- No utilizar partes o palabras usadas anteriormente.
- No utilizar partes de palabras o palabras que integren al nombre del usuario.
- No utilizar nombres propios.
- El password debe ser fácilmente recordable para el usuario.
- Asignar un tiempo de vigencia al password.

### **Seguridad de root**

- Usar sólo la cuenta de Root para tareas muy cortas y específicas, trabajando, en lo posible como usuario normal.
- Antes de usar cualquier comando complejo, hacer pruebas en ficheros o archivos de poca importancia.

### **Seguridad en el sistema de Archivos y archivos**

- Definir una partición en la cual los dispositivos y procesos de sistema van a poder correr sus procesos, programas o aplicaciones.
- Definir una partición en la cual los usuarios como grupo van a poder correr sus procesos, programas o aplicaciones. Se puede controlar los límites por usuario que éstos usan en `/etc/pam.d/limits.conf`

- No instalar, usar o correr programas SUID y SGID.
  - En caso de encontrar alguno de estos programas se debe registrar su existencia, donde se encontró, a qué usuario concede privilegios y cuándo sucedió esto, además de tomar nota de esta actividad para protegerse de ataques futuros.
- Eliminar archivos desconocidos en el sistema, que no tienen ningún dueño, o pertenece a ningún grupo.
- Dar permisos adecuados a los archivos, que sean suficientemente seguros y funcionales.
- Comprobar la integridad periódicamente.
- Llevar estricto control de programas instalados en el servidor.
- Implantar el uso de un sistema de criptografía para hacer seguras las transacciones que se lleven a cabo en el sistema.
- Prohibir el uso de comunicación remota con el servidor o hacerlo a través de secure shell.

## Seguridad en Respaldos

- Etiquetar los respaldos. Tomando en cuenta los siguientes aspectos:
  - Indicar la utilería utilizada y con qué condiciones.
  - Especificar los archivos y/o respaldados de archivos de sistema.
  - Nombre de la máquina servidor.
  - Tamaño de lo respaldado.
  - Guardar los respaldos fuera del sitio.
  - Reducir las actividades al máximo cuando se está haciendo un respaldo.
  - Verificar que el respaldo esté bien hecho.
  - El intervalo de tiempo para respaldar es la siguiente:

	INTERVALO
www Internet	Cada vez que se cambie intencionalmente
Base de Datos	Cada vez que se cambie intencionalmente
Desarrollo	Cada vez que se cambie intencionalmente
Sistema Operativo	1 mes

## 6.4. Seguridad Física y Lógica

### 6.4.1. Seguridad Física

Como ya se dijo anteriormente, la Seguridad Física (SF) se refiere a todo lo relacionado con el sistema a proteger que sea tangible, todo lo que pueda tocarse. Y para proteger esto se deben tomar en cuenta los siguientes puntos:

- Reinicio. Una de las maneras en que la máquina puede estar comprometida, o vulnerada es cuando es reiniciada o apagada. Las únicas circunstancias bajo las cuales se debe reiniciar un servidor es:

- Para actualizaciones el SO
- Cambios en el hardware
- Es necesario que esté fuera de servicio
- Protección con cerraduras. Hay algunas computadoras que cuentan con cerraduras incluidas, o entradas para candados, para así asegurar que no se roben hardware del equipo, o dañarlo de alguna forma.
- Seguridad en el BIOS. El BIOS es el nivel más bajo de "software" que configura o manipula el hardware, otros métodos de "booted"<sup>41</sup> de Linux acceden al BIOS para determinar cómo iniciar la máquina, para efectos de protección se puede usar el BIOS para impedir a los intrusos reiniciar la máquina o manipularla.
  - Algo recomendable para evitar que intrusos tengan acceso franco al equipo es poner una contraseña de *boot* (en caso de querer cambiar la unidad de inicio).
  - Considerar la importancia de que el equipo al iniciar sólo cargue información de memoria, no de CD, o disquete, sólo si se le configura para ello.
  - Es importante quitar contraseñas predefinidas que los fabricantes dan a los equipos, esto se desactiva dando una nueva contraseña.
  - Es necesario tener presente que si se puso una contraseña al BIOS, la máquina no podrá iniciar en forma desatendida, es decir, cada vez que sea reiniciada, deberá darse contraseña para que pueda iniciar.
- Seguridad en el "booted" y en terminales. Se puede poner contraseña de "booted", para proteger a nuestra máquina un poco más, pero es importante recordar lo siguiente:
  - Desactivar "booted" desde otras unidades que no sean la que deseamos en el BIOS, y estar seguros de que también este está protegido con contraseña.
  - Proteger el fichero que contenga las contraseñas del BIOS y del boot.
  - Es necesario tener presente que si se puso una contraseña al BIOS, la máquina no podrá iniciar en forma desatendida, es decir, cada vez que sea reiniciada, deberá darse contraseña para que pueda iniciar.
- Cuando la máquina no está siendo usada, es necesario proteger las sesiones que se han abierto para ocuparla, para que nadie pueda manipularla para esto hay dos herramientas muy útiles: *xlock* y *vlock*.
  - Xlock es un desplegador de X (así se llama a la interfaz grafica en Linux). Está incluido en cualquier distribución de Linux con interfaz grafica. Se puede ejecutar el *xlock* y cerrará con llave el despliegue de X y exigirá su contraseña para abrir.
  - Vlock es un programa pequeño simple que le permite cerrar con llave alguno o todas las consolas virtuales en Linux. Es mejor asegurar todas las consolas, para evitar que intrusos entren por otras. Esta versión está incluida en Red Hat Linux, que es la plataforma de nuestro sistema.
- Seguridad en dispositivos locales. Si se tiene una cámara web o micrófono conectado al sistema, puede haber peligro de que un intruso tenga acceso a esos dispositivos. Cuando no estén en uso, desconectarlos o quitarlos son opciones y tener cuidado del "software" que los maneja, que este no proporcione el acceso a los dispositivos.

---

<sup>41</sup> Es la forma en que inicia una máquina, donde busca y lee los archivos de configuración de sistema. Puede ser en la memoria de la máquina, CD o disquete.

## 6.4.2. Seguridad Lógica

### 6.4.2.1. Seguridad local

Este punto consta de los siguientes incisos:

#### 6.4.2.1.1. Seguridad en cuentas

Es necesario asegurarse de que se proporcionan los servicios mínimos, lo estrictamente necesario, para las tareas que los usuarios necesitan hacer. Las siguientes reglas son para permitir acceso legítimo:

- Dar a los usuarios la cantidad mínima de privilegios necesiten.
- Vigilar donde y cuándo se conectan los usuarios y llevar nota.
- Quitar cuentas inactivas (comando "*last*") y registrarlo si es necesario.
- Evitar la aparición de cuentas:
  - Por defecto
  - Abiertas
  - Dormidas
  - Restringidas
  - privilegiadas
- Usar el comando "*userid*" en todas las computadoras y redes para el mantenimiento de las cuentas.
- No permitir la creación de cuentas grupales, para fincar responsabilidades directas.

No se han usado muchos cuentas usuarios locales que se usan en los compromisos de seguridad en meses o años. Desde que nadie está usándolos ellos, proporcione el vehículo del ataque ideal.

#### 6.4.2.1.2. Seguridad de "root"

La cuenta de "root" o super usuario tiene la autoridad de toda la máquina también puede incluir la autoridad en otras máquinas de la red, por lo que esta cuenta sólo puede usarse para tareas muy cortas, específicas, y preferentemente usar la cuenta de un usuario normal, ya que mientras menos tiempo pasa con los privilegios de "root", más seguro será el sistema. Hay varias formas de no hacer cosas indeseadas en el sistema siendo "root":

- Cuando haga algún comando complejo, haga pruebas primero en una forma no destructiva, en especial con comandos globales, que hacen varias tareas a la vez.
- Provea a sus usuarios con un "alias" o seudónimo predefinido (grupal), para el comando "rm" que sea útil al pedir confirmación de borrar archivos.
- Sólo entre al sistema como "root" para hacer tareas específicas.

- El comando "path" es el ambiente de la variable de ambiente, especifica el directorio en el cual el "shell" busca por programas.
- Trate de limitar este comando para "root" tanto como posible y nunca ponga incluye que es el directorio actual, en su path. Adicionalmente, no tenga directorios escribibles en el camino de búsqueda, pues puede permitir a intrusos modificar o poner nuevos archivos en el camino de búsqueda permitiéndoles correr como "root" la próxima vez que corra ese comando.
- Nunca debe usarse el conjunto de herramientas de rlogin/rsh/rexec, que son llamadas las utilidades de "r" como root, pues resulta muy peligroso cuando se corren como root. Así mismo nunca crear un archivo .rhosts para root.
- El archivo de /etc/securetty contiene una lista de terminales desde donde root puede conectarse, en Red Hat por defecto esto se permite solamente a las consolas virtuales.
- Es importantísimo recordar que todas las acciones que se hagan como root podrían afectar en muchos aspectos al sistema.

#### **6.4.2.1.3. Seguridad de contraseñas.**

La contraseña de usuario es la llave de entrada que tiene el mismo para entrar al sistema, es por ello que debe permanecer segura todo el tiempo, Linux incluye programas para *passwd* (carpeta que alberga contraseñas) a través de ellos no se permite la creación de contraseñas fáciles de adivinar.

Las siguientes son recomendaciones para escoger una buena contraseña:

- Tamaño de la contraseña debe ser mayor o igual a 6 u 8 caracteres, no menor.
- La contraseña debe formarse de la combinación de letras, números y caracteres especiales.
- La contraseña no debe ser igual al "login" o nombre de usuario.
- No debe ser una palabra del diccionario.
- No usar patrones de teclado.
- No utilizar patrones en la contraseña. Por ejemplo la utilización de la fecha o el numero de semestre que se este cursado.
- No usar contraseñas ya usados en cualquier otra cuenta o que integren al nombre del "login" o nombre de usuario.
- Pensar en el uso de una frase que contenga un espacio en blanco.
- No utilizar nombre propios.
- La contraseña debe ser fácilmente recordable para el usuario.



### 6.4.3. Seguridad en el sistema de archivos y archivos individuales.

Es importante proteger un sistema antes de ponerlo en línea y resguardar los datos guardados en él, y protegerlo una vez que esté en uso, para lo cual se pueden tomar las siguientes medidas.

- No se debe permitir bajo ninguna circunstancia alojar o correr en el directorio de usuarios (llamado *home* por defecto en linux), correr programas SUID O SGID, ya que atentan contra el sistema. Algunas herramientas para protegerse de esto son las siguientes:
  - Usar la opción *nosuid* del */etc/fstab* para particiones escribibles por otros además de root.
  - Usar *nodev* y *noexec* en la partición del *home* de los usuarios, así como */var*, prohibiendo la ejecución de programas, y creación de archivos para dispositivos que no deben hacer los usuarios.
- Si se exportó un sistema de archivos usando NFS, se debe configurar */etc/exports* con acceso lo más restrictivo posible. Esto significa no usar "*wild cards*", no permitir escritura en raíz<sup>42</sup>
- Configurar con la herramienta *umask* la creación archivos de usuarios siendo tan restrictivo como posible. Lo cual se hace de la siguiente forma:

La orden del *umask* puede usarse para determinar el modo de creación de archivo predefinido en su sistema. Es el complemento octal del modo de archivo deseado. Se deben crear archivos considerando los permisos ya que se puede permitir lectura o escritura a alguien que no debe tener éstos permisos. *Umask* incluye los permisos 022, 027, y 077 (qué es el más restrictivo). Normalmente el *umask* se encuentra en */etc/profile*, aplica a todos los usuarios de sistema. La máscara de creación de archivo puede calcularse restando el valor deseado de 777, si se pusiera este permiso a archivos creados recientemente, no permitiría lectura, escritura o ejecución para nadie, o una máscara de 666 pondría una máscara de 111.

- Si se montó el sistema de archivos usando un sistema de red como NTFS, como NFS, se debe configurar */etc/exports* con restricciones convenientes. Típicamente, usando "*nodev*", "*nosuid*", y "*noexec*", si se desea.
- Ponga límites en espacio de memoria al sistema de archivos en lugar de espacio ilimitado, valor por defecto. Se puede controlar los límites de espacio en memoria por usuario o grupo.
- Los archivos */var/log/wtmp* y */var/run/utmp* contienen el "*login*" para todos los usuarios en su sistema, por ello su integridad debe mantenerse, ya que pueden usarse para determinar cuando y dónde un usuario (o potencial intruso) ha entrado en el sistema. Deben tener permisos 644.
- El comando *chattr(1)* puede usarse para prevenir de borrado accidental, o sobrescritura a través del "*bit immutable*", también previene de ligas a algunos archivos.

---

<sup>42</sup> Lugar de origen o punto de partida del sistema de archivos.

- Archivos SUID y SGID en su sistema son un riesgo de seguridad potencial y deben monitorearse de cerca, ya que proporcionan privilegios especiales al usuario que los ejecuta, y es por ello que resulta necesario asegurar que este tipo de programas no se instalen. Es importante averiguar si se tiene corriendo éstos programas en el sistema y guardar una descripción de qué hacen, para vigilar cambios en el mismo. El siguiente comando es para encontrar programas de SUID/SGID en el sistema:

```
Root # find / -type f \( -perm -04000 -o -perm -02000 \)
```

Se puede ejecutar durante la noche un programa para detectar archivos de SUID. Lo cual se puede ver en `/var/log/setuid`.

Para quitar permisos SUID o SGID a programas sospechosos con el comando `"chmod"`, y se restaurarlo si es absolutamente necesario.

- El sistema de archivos, por ser *re-escribibles*, puede ser un agujero de seguridad si un intruso gana acceso al sistema y los modifica, además al permitir a los intrusos agregar o quitar archivos que desee, por ello es importante localizar este tipo de archivos, cosa que se logra con el siguiente comando:

```
root # find / -perm -2 ! -type l -ls
```

Se debe saber por que éstos archivos son *re-escribibles*. Hay algunos archivos que deben ser *re-escribibles* incluyendo algunos de `/dev` y algunas ligas simbólicas como `i` -type `l` que los excluye del comando anterior.

- Archivos sin dueño también pueden ser una indicación de que un intruso ha entrado al sistema, para localizar archivos en el sistema que no tienen dueño, o que no pertenecen a ningún de grupo se da el comando:

```
root # find / \( -nouser -o -nogroup \) -print
```

- Otra tarea muy importante es encontrar archivos `.rhosts` ya que estos archivos no deben permitirse en el sistema, porque un intruso sólo necesita una cuenta insegura para ganar acceso a la red. Para localizar archivos `.rhosts` se da la siguiente orden:

```
root# find /home -name .rhosts -print
```

- Antes de cambiar permisos en el sistema de archivos, se debe estar seguro de entender lo que se hace, determine siempre por qué el archivo tiene esos permisos antes de cambiarlo.

### 6.4.3.1. Permisos en archivos

Los permisos en Linux, son fundamentales, ya que involucra el acceso, la seguridad, la accesibilidad y la comodidad de los usuarios para ejecutar sus tareas de acuerdo al nivel de acceso que poseen en el sistema. Los permisos son un mecanismo proporcionado por el sistema para proteger ficheros de usuarios del sistema, y de la manipulación de otros usuarios, pues Linux es multiusuario. Los permisos están divididos en tres tipos:

1. Lectura, representado por la letra "r".
  - Para poder ver el contenido de un archivo
  - Para poder leer un directorio
2. Escritura, representado por la letra "w".

- Para poder agregar a o cambiar un archivo
  - Para poder borrar o mover los archivos en un directorio
3. Ejecución, representado por la letra "x".
- Para poder ejecutar un programa binario o script en shell.
  - Para poder buscar en un directorio, combinado con el permiso de la lectura.

A su vez estos permisos pueden ser fijados para tres clases de usuarios:

1. Dueño del archivo: "u".
2. Grupo al que pertenece el archivo: "g".
3. El resto de usuarios: "o".

Hay siempre exactamente un dueño, cualquier número de miembros del grupo, y todos los demás.

El comando *chmod* es el que va a permitir manipular todos los permisos, consta de 3 operadores, veamos cuales son:

1. "+" para agregar permisos.
2. "-" para quitar permisos.
3. "=" para asignar permisos.

La sintaxis es la siguiente: `chmod "operador" "permiso" "fichero"`. Se puedan dar permisos sobre determinado archivo a root ("u"), a grupos ("g"), o al resto de los usuarios ("o").

Es importante tener cuidado de que los permisos serán aplicados al directorio en que se encuentra, y si no se tiene acceso al directorio, no se podrá tener acceso a los archivos que éste contiene.

Para definir permisos también se puede usar el sistema octal, en este sistema los números representan permisos. Estos se basan en la suma de los permisos de: ejecución, escritura y lectura en ese orden. La combinación de estos, nos da números del cero al siete:

- 0 = sin permisos.
- 1 = ejecución.
- 2 = escritura.
- 3 = escritura y ejecución.
- 4 = lectura.
- 5 = lectura y ejecución.
- 6 = lectura y escritura.
- 7 = lectura, escritura y ejecución.

Los permisos son números de tres dígitos, y se otorgan de la siguiente forma:

0	0	0
Root	Grupo	El resto de los usuarios

Además de los permisos anteriores existen otros permisos que hay que conocer: SUID, SGID y STICKY BIT. Cuando se necesita dar privilegios especiales a cierto usuario se usan esta clase de programas, que permiten ejecutar programas con los privilegios del dueño y/o grupo del programa. Cada usuario está identificado por el sistema con un número de identificación tanto para él, como para el grupo. Este número se denomina UID (user ID) para el caso de los usuarios y GID para el caso de los grupos. En el caso del root, este tiene UID 0 y GID 0.

- **SUID y SGID.** Lo que se efectúa con el sistema SUID es una adquisición temporal de un UID o GID distinto al propio cuando se está ejecutando el programa. Cuando un programa cambia de UID se denomina SUID (se establece UID). Cuando cambia de GID se denomina SGID (se establece GID) Un programa puede ser SUID y SGID al mismo tiempo. Es importante tener en cuenta que:
  - 4000 Establece el número de identificación de usuario al ejecutarse SUID.
  - 2000 Establece el número de identificación de grupo al ejecutarse SGID.

Para darse cuenta si un programa es SUID o SGID basta con hacer un listado largo con el comando `ls -l` y se verá que donde tendría que estar una `x`, que asigna permisos de ejecución, va a estar una letra `s`. En caso de querer saber que programas utilizan el SUID se buscan de la siguiente forma:

```
$find / -perm +4000
```

Hay dos tipos de aplicación de este permiso:

- SGID para archivos. Si se da en el grupo de permisos, este bit controla el estado de un archivo "set group id". Se comporta de la misma forma que SUID, a excepción que el grupo es, en cambio, afectado. El archivo debe ser ejecutable para que esto tenga cualquier efecto.
- SGID para directorios. Si se pusiera el bit SGID en un directorio (con `chmod g+s directorio`), los archivos creados en ese directorio tendrán su grupo en el directorio del grupo. Los posibles participantes son: root (el dueño del archivo), el Grupo (al que pertenece), o todos (cualquiera en el sistema que no es el dueño o miembro del grupo).

Para mayor seguridad con este tipo de permisos se pueden hacer las siguientes tareas:

- Quitarlos. Se hace con el siguiente comando: `chmod -s archivo`
- Eliminar los programas con este tipo de permiso que ya no sean útiles.
- Asegurarse de que no se puede escribir en los script de SUID.
- Instalar alguna herramienta que verifique los archivos *suid* del sistema (COPS).

- **STICKY BIT.** Es un bit que tiene un significado para los directorios. 1000 Establece el bit adhesivo (sticky bit). Cuando este bit está activo, hace que un usuario sólo pueda borrar los ficheros que son de su propiedad en dicho directorio. Esto es particularmente útil en el directorio /tmp. El sticky bit se activa de la siguiente forma:

```
$ chmod +t directorio
```

Ya sabiendo esto es importante observar que los archivos de sistema no deben ser editados por usuarios o grupos que no estén encargados del mantenimiento de sistema.

### 6.4.3.2. Chequeos de integridad

Los chequeos de integridad se hacen con el fin de verificar si el sistema no se ha corrompido, se corren comprobadores de integridad que ejecutan un número de chequeos en todo los archivos binarios y de configuración importantes y los comparan contra un banco de datos de valores anteriores, valores conocidos como referencia segura, de esta forma se marca cualquier cambio en los archivos. Un comprobador de integridad para Linux es Tripwire.

Es mejor instalar esta clase de programas en unidades externas (disquetes, CD, discos duros, etc.), y protegerlos físicamente, de esta manera los intrusos no pueden manipular los comprobadores de integridad o cambiar el banco de datos.

Se puede agregar una entrada del crontab para ejecutar el comprobador desde la unidad externa todas las noches y mandar por correo los resultados en la mañana. Lo cual se hace de la siguiente forma:

```
# set mailto
MAILTO=kevin
# run Tripwire
15 05 * * * "root" /usr/local/adm/tchecar/tripwire
```

La orden anterior indica: mandar un correo con el informe cada mañana a las 5:15am.

Cuando en el sistema cambian muchos archivos, se debe tener mucho cuidado en lo que es la actividad del intruso y lo que los cambios que usted ha hecho.

Un peligro constante contra la integridad de un sistema son los Caballos de Troya (Trojan Horses), se llama así al hecho de que un intruso distribuye un programa que parece bueno, y logra que se baje y ejecute en una máquina ajena con los privilegios de "root", este virus se ejecuta junto con una función plenamente reconocida, puede ser un programa que parezca legítimo, pero que en realidad tiene código malicioso que puede suplantar a esa u otras funciones o ejecutarse de modo paralelo.

Por ello es importante tener cuidado de qué programas se instalan, Red Hat proporciona chequeos con MD5 y firmas PGP en los archivos RPM, también se puede verificar si se está instalando algo genuino. De ser posible obtener la fuente del programa del sitio de distribución real, así como examinar la fuente y verificarla.

## 6.4.4. Seguridad con criptografía

Las siguientes secciones fueron extraídas del manual oficial de Red Hat, por lo que solo se considera como referencia y no tiene alteraciones.

### 6.4.4.1. Configuración de SSH

Red Hat contiene los paquetes de servidor (`openssh-server`) y cliente (`openssh-clients`) OpenSSH, al igual que el paquete OpenSSH general (`openssh`) que debe ser instalado para que cualquiera de los dos funcione. Consulte *Official Red Hat Linux Customization Guide* para obtener las instrucciones de instalar y desplegar OpenSSH.

Los paquetes OpenSSH requieren del paquete OpenSSL (`openssl`). OpenSSL instala varias bibliotecas criptográficas importantes que sirven para que OpenSSH proporcione comunicaciones cifradas. Debe instalar el paquete `openssl` antes de instalar cualquiera de los paquetes OpenSSH.

**Configurar un servidor OpenSSH.** Para poner en funcionamiento un servidor OpenSSH, debe asegurarse primero que su sistema tiene los paquetes RPM instalados. Se requiere el paquete `openssh-server`, que depende del paquete `openssh`. Ambos paquetes están incluidos en Red Hat.

El demonio de OpenSSH utiliza el fichero de configuración `/etc/ssh/sshd_config`. Por defecto, el fichero de configuración instalado con Red Hat debería ser suficiente para la mayoría de las configuraciones. Si quiere configurar su propio demonio de otra manera que no sea la proporcionada por defecto en el `sshd_config`, lea el manual de `sshd`, que le proporciona una lista de palabras reservadas que pueden ser utilizadas por el fichero de configuración.

Para iniciar un servicio OpenSSH, utilice el comando `/sbin/service sshd start`. Para parar el servicio OpenSSH, utilice el comando `/sbin/service sshd stop`.

### 6.4.4.2. Kerberos

Cuando configure Kerberos, instale primero el servidor en primer lugar. Para instalar un servidor Kerberos:

1. Asegúrese de que tiene una sincronización de reloj y DNS funcionando en su servidor antes de instalar Kerberos 5. Ponga especial atención a la sincronización del tiempo entre el servidor Kerberos y sus diversos clientes. Si los relojes de servidor y cliente difieren más de 5 minutos (esta cantidad por defecto puede ser configurada en Kerberos 5), a los clientes Kerberos no les será posible autenticarse en el servidor. Esta sincronización de reloj es necesaria para evitar que un agresor use un autenticador antiguo para hacerse pasar por un usuario válido.

Debería configurar un Network Time Protocol (NTP) compatible con una red cliente/servidor usando Red Hat Linux, aunque no utilice Kerberos. Red Hat Linux 7.1 incluye el paquete `ntp` para una instalación fácil.

2. Instale los paquetes `krb5-libs`, `krb5-server` y `krb5-workstation` en el ordenador que ejecutará su KDC. Este ordenador debe ser seguro — si es posible, no debería ejecutar ningún servicio aparte de KDC.

Si quiere usar la utilidad Graphical User Interface (GUI) para administrar Kerberos, debería instalar también el paquete `gnome-kerberos`. `gnome-kerberos` contiene `krb5`, una herramienta GUI para administrar tickets y `gkadmin`, una herramienta GUI para administrar dominios Kerberos.

3. Modifique los ficheros de configuración `/etc/krb5.conf` y `/var/kerberos/krb5kdc/kdc.conf` para reflejar su nombre de entorno y sus mapas de entorno a dominio. Un entorno sencillo puede ser construido para reemplazar instancias de `EXAMPLE.COM` y `example.com` con su nombre del dominio (asegúrese de mantener los nombres en mayúsculas y minúsculas en el formato correcto) y cambiando el KDC desde `kerberos.example.com` por el nombre de su servidor Kerberos.

Por convenio, todos los nombres de entornos son en mayúsculas y todos los nombres de host DNS y nombres de dominios son en minúsculas. Para más detalles sobre el formato de estos ficheros, lea las páginas de manual correspondientes.

4. Cree la base de datos usando la utilidad `kdb5_util` desde el intérprete de comandos de la shell:

```
/usr/kerberos/sbin/kdb5_util create -s
```

El comando `create` crea la base de datos que será usada para guardar las claves de sus entornos Kerberos. La opción `-s` fuerza la creación de un fichero `stash` en el que se guarda la clave del servidor maestro. Si no existe ningún fichero `stash` del que leer la clave, el servidor Kerberos (`krb5kdc`) pedirá al usuario la contraseña del servidor maestro (que puede ser usada para regenerar la clave) cada vez que inicie.

5. Modifique el fichero `/var/kerberos/krb5kdc/kadm5.acl`. `kadmind` usa este fichero para determinar qué principals tienen acceso a la base de datos Kerberos y su nivel de acceso. La mayoría de las organizaciones podrán estar en una sola línea:

```
*/admin@EXAMPLE.COM *
```

6. La mayoría de usuarios están representados en la base de datos por un sólo principal (con un `NULL`, o vacío, instance, como `joe@EXAMPLE.COM`). Con esta configuración, los usuarios con un segundo principal con un instance de `admin` (por ejemplo, `joe/admin@EXAMPLE.COM`) podrán ejercer un dominio completo sobre la base de datos del entorno Kerberos.
7. Una vez que `kadmind` esté iniciado en un servidor, algunos usuarios podrán acceder a sus servicios ejecutando `kadmin` o `gkadmin` en cualquiera de los clientes o servidores en el entorno. Sin embargo, sólo los usuarios listados en el fichero `kadm5.acl` podrán modificar la base de datos de cualquier modo, excepto para cambiar sus propias contraseñas.

**Nota.** Las utilidades `kadmin` y `gkadmin` comunican con el servidor `kadmind` sobre la red. Naturalmente, necesita crear el primer principal antes de que pueda conectarse al servidor sobre la red para administrarlo. Cree el primer principal con el comando `kadmin.local`, que está especialmente diseñado en el mismo host que el KDC y no usa Kerberos para la autenticación.

8. Teclee el siguiente comando `kadmin.local` en el terminal KDC para crear el primer principal:

```
/usr/kerberos/sbin/kadmin.local -q "addprinc username/admin"
```

9. Inicie Kerberos utilizando los siguientes comandos:

```
/sbin/service krb5kdc start  
/sbin/service kadmin start  
/sbin/service krb524 start
```

10. Añada principals para sus usuarios utilizando el comando `addprinc` con `kadmin` o usando la opción del menú Principal => Añadir en `gkadmin`. `kadmin` (y `kadmin.local` en el maestro KCD) es una interfaz de línea de comandos para el sistema de administración de Kerberos. Como tales, muchos comandos están disponibles tras la puesta en marcha del programa `kadmin`. Vea la página de manual `kadmin` para más información.
11. Verifique que su servidor emite los tickets. En primer lugar, ejecute `kinit` para obtener un ticket y guardarlo en un fichero de caché credencial. A continuación use `klist` para ver la lista de credenciales en su caché y use `kdestroy` para destruir el caché y los credenciales que contiene.

Nota. Por defecto, `kinit` intenta autentificarle usando el nombre de login de usuario de la cuenta que ha usado cuando se registró en su sistema por primera vez (no en el servidor Kerberos) . Si el nombre de usuario del sistema no corresponde con una base de datos principal de Kerberos, obtendrá un mensaje de error. Si esto sucede, dé a `kinit` el nombre de su principal como un argumento en la línea de comandos (`kinit principal`).

Una vez que se hayan completado los pasos anteriores, su servidor Kerberos debería estar funcionando. A continuación, necesitará configurar sus clientes Kerberos.

#### 6.4.4.3. Contraseñas de Shadow.

El uso de utilidades Shadow (también conocidas como contraseñas shadow) sirve para la protección incrementada ofrecida por los ficheros de autenticación de su sistema. Durante la instalación de Red Hat la protección de la contraseña shadow para su sistema se habilita por defecto, así como contraseñas MD5 (método alternativo para cifrar contraseñas para el almacenamiento en su sistema).

Las contraseñas shadow ofrecen ventajas respecto a los almacenamientos de contraseñas estándar anteriores de UNIX y LINUX que incluyen:

- Seguridad del sistema mejorada al trasladar las contraseñas cifradas (que se encuentran normalmente en `/etc/passwd`) a `/etc/shadow` que tan sólo puede ser leído por root.
- Información referente a la antigüedad de la contraseña (tiempo transcurrido desde la última vez que se cambió la contraseña)
- Control sobre el tiempo que puede permanecer una contraseña antes de que al usuario se le pida que la cambie.
- Habilidad para usar el fichero `/etc/login.defs` para reforzar la política de seguridad, especialmente en lo referente a la antigüedad de la contraseña.



El paquete shadow-utils contiene un número de utilidades que soportan:

- Conversión de contraseñas normales a shadow y viceversa (pwconv, pwunconv)
- Verificación de la contraseña, grupo y ficheros shadow asociados (pwck, grpck)
- Métodos estándar de industria para añadir, borrar y modificar las cuentas de usuario (useradd, usermod, y userdel)
- Métodos estándar de industria para añadir, borrar y modificar los grupos de usuario (groupadd, groupmod, y groupdel)
- Método estándar de industria para administrar el fichero /etc/group utilizando gpasswd

**Notas.** 1) Las utilidades funcionarán dependiendo de si se permite o no el shadowing.  
2) Las herramientas en el paquete shadow-utils no están habilitadas ni por Kerberos ni por LDAP. Los usuarios nuevos serán sólo locales.

## 6.4.5. Seguridad del Núcleo (Núcleo).

En esta sección se presenta una descripción de las opciones de configuración del núcleo que se relacionan con la seguridad, una explicación de lo que hacen, y cómo usarlos.

### 6.4.5.1. Opciones de compilación del Núcleo

Las opciones siguientes son en gran parte del documento /linux/Documentation/Configure.help.

- IP: Drop source routed frames (CONFIG\_IP\_NOSR). Esta opción debería estar activada. Source routed frames contienen la ruta completa de sus destinos dentro del paquete. Esto significa que los ruteadores a través de los que circula el paquete no necesitan inspeccionarlo, y sólo lo reenvían.
- IP: Firewalling (CONFIG\_IP\_FIREWALL). Esta opción es necesaria si va a configurar su máquina como un cortafuego, hacer enmascaramiento o desea proteger su estación de trabajo con línea telefónica de que alguien entre a través de su interfaz PPP.
- IP: forwarding/gatewaying (CONFIG\_IP\_FORWARD). Si se activa reenvío IP (IP forwarding), la máquina Linux esencialmente se convierte en un ruteador. Se puede activar y desactivar el reenvío IP (IP forwarding) dinámicamente usando el siguiente comando:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

y desactivarlo con el comando:

```
root# echo 0 > /proc/sys/net/ipv4/ip_forward
```

- IP: firewall packet logging (CONFIG\_IP\_FIREWALL\_VERBOSE). Esta opción le da información sobre los paquetes que el cortafuegos recibe, como remitente, recipiente, puerto, etc.

- IP: always defragment (CONFIG\_IP\_ALWAYS\_DEFRAG). Generalmente esta opción está desactivada, pero si se está construyendo una máquina cortafuegos o para enmascaramiento, deberá activarla. Cuando se envía de una máquina a otro, no siempre se envía como un simple paquete de datos, sino que se fragmenta en varios trozos. El problema de esto es que los números de puerto sólo se almacenan en el primer fragmento, por lo que alguien puede insertar información en el resto de los paquetes para la conexión que se supone que no deberían.
- IP: syn cookies (CONFIG\_SYN\_COOKIES). El ataque SYN es un ataque de denegación de servicio (denial of service DoS) que consume todos los recursos de la máquina forzando un reinicio, por lo que se debe activar esta opción con:

```
"root"# echo 1 > /proc/sys/net/ipv4/tcp_syncookies <P>
```

- Packet Signatures (CONFIG\_NCPFS\_PACKET\_SIGNING). Esta es una opción disponible en los núcleos 2.1 que firmarán los paquetes NCP para una mayor seguridad. Normalmente puede dejarlo desactivado, pero está allí por si se necesita.
- IP: Firewall packet netlink device (CONFIG\_IP\_FIREWALL\_NETLINK). Esta es una opción que le permite analizar los primeros 128 bytes de los paquetes en el espacio de programa de usuario, para determinar si le gustaría aceptar o denegar el paquete basado en su validez.
- IP: masquerading (CONFIG\_IP\_MASQUERADE). Si una de las computadoras en la red local para las cuales la máquina Linux da soporte como firewall quiere enviar algo al exterior, esta máquina puede hacerse pasar por ese *host*, es decir, él envía el tráfico al destino intencional, pero hace ver que vino del firewall.
- IP: ICMP masquerading (CONFIG\_IP\_MASQUERADE\_ICMP). Esta opción agrega ICMP masquerading a la opción anterior de sólo masquerading TCP o tráfico UDP.
- IP: transparent proxy support (CONFIG\_IP\_TRANSPARENT\_PROXY). Esto permite a un firewall redirigir transparentemente cualquier tráfico de la red originado de la red local y destinado de una máquina remota a un servidor local, llamado servidor proxy transparente. Esto hace que las computadoras locales crean que se están comunicando con el host remoto, mientras, de hecho ellos se conectan al proxy local.

#### 6.4.5.2. Dispositivos del Núcleo

Los dispositivos están divididos en dos tipos: los dispositivos de carácter y los dispositivos de bloque. Los dispositivos de bloque tienen un búfer para las peticiones, y pueden escoger en qué orden las van a responder, los dispositivos de bloque sólo pueden aceptar bloques de entrada y de salida (cuyo tamaño puede variar según el dispositivo), y los dispositivos de carácter pueden usar muchos o unos pocos bytes como ellos quieran. Se puede saber cuándo un fichero de dispositivo es para un dispositivo de carácter o de bloque mirando el primer carácter de la salida de `ls -l`. Si es "b" entonces es un dispositivo de bloque, y si es "c" es un dispositivo de carácter.

Hay algunos dispositivos de bloque y de carácter disponibles en Linux que también lo ayudará con la seguridad. Los dos dispositivos que `/dev/random` y `/dev/urandom` son proporcionados por el Núcleo para proporcionar los datos aleatorios cuando se quiera.

`/dev/random` y `/dev/urandom` deben ser suficientemente seguros para usarse en generar llaves PGP, cuestionar el ssh, y otras aplicaciones dónde se requieren números aleatorios seguros. Los atacantes deben ser incapaces de predecir el siguiente número dado cualquier secuencia inicial de números de estas fuentes. Se podrá leer a cerca de los dispositivos con:

```
"root"#head -c 6 /dev/urandom | mimercode
```

## 6.5. Seguridad de Red

### 6.5.1. Servicios del sistema

En esta sección se describe cómo verificar qué servicios se necesita ofrecer, y desactivar los que no sean necesarios:

- Se observa en el fichero `/etc/inetd.conf` y se identifica qué servicios están siendo ofrecidos por `inetd` (demonio de los servicios de red) y desactivar los que no necesite comentándolos (`#` al principio de la línea), y después enviar a su proceso `inetd`.
- Eliminar (o comentar) servicios en el fichero `/etc/services`, con lo que los clientes locales tampoco podrán encontrar el servicio si lo elimina.
- Cuando se sabe que no se va a utilizar algún paquete en particular, se puede borrar entero, con `rpm -e` con la distribución Red Hat.

Es importante desactivar las siguientes utilidades para que no sean iniciados en `/etc/inetd.conf`, pues estos protocolos son muy inseguros, ya que pueden causar la caída del sistema.

- `rsh/rlogin/rcp`, incluyendo `login` (usado por `rlogin`)
- `shell` (usado por `rcp`)
- `exec` (usado por `rsh`)

Comprobar cada uno de los ficheros `/etc/rc.d/rcN.d`, donde `N` es el nivel de ejecución del sistema (systems run level) y ver si alguno de los servidores iniciados en esos directorios no son necesarios, éstos ficheros son enlaces simbólicos al directorio `/etc/rc.d/init.d`, renombrar el fichero en el directorio `init.d` tiene el efecto de desactivar todos los enlaces simbólicos que hay en `/etc/rc.d/rcN.d`. Si se desea desactivar un servicio para un determinado nivel de ejecución, renombre el fichero adecuado con una 's' minúscula en lugar de la 'S' mayúscula.

Para ayudar con esta tarea se puede usar `tcp_wrappers` que "cubre" todos los servicios tcp. Se deberá crear un fichero `/etc/hosts.allow` y añadirle sólo aquellos host que necesitan tener acceso a los servicios de la máquina.

## 6.6. Seguridad preventiva

### 6.6.1. Respaldos

La aplicación práctica de los respaldos recae en el administrador de sistema, y para ello debe regirse por las políticas de seguridad descritas en la sección 6.3. A continuación se describen algunas herramientas útiles para hacer respaldos en Linux:

- **TAR.** Comando que permite empaquetar archivos en un contenedor *tar*, es decir almacena otros archivos en uno solo. El contenedor puede ser otro archivo, una cinta o una tubería. Los archivos deben ser ordinarios, FS completos. Su nombre significa Tape Archiver.

**Sintaxis:** `tar llave [argumentos] archivo | directorios [...]`

*Llave.* Debe contener alguna de las siguientes funciones:

- c.** Crear un respaldo.
- x.** Extraer archivos de un respaldo.
- t.** Listar los archivos de un respaldo,

*Argumentos. Opciones más utilizadas:*

- v.** Proporcionar información de lo que se está haciendo (verbose)
- f file.** Utilizar el archivo específico en vez de la unidad de cinta por defecto para hacer el respaldo. Se usa para hacer respaldos en un archivo en disco o en dispositivos alternos.  
Se puede utilizar - para especificar la entrada o salida estándar.

En *tar* los argumentos no se ponen inmediatamente después de su letra correspondiente, sino que todas las letras se agrupan en un solo bloque y después se tienen que poner todos los argumentos en el orden correcto. Ejemplo:

```
$ tar -cvbf 20 /dev/rmt8
```

Ejemplos:

```
$ tar c
```

Crear un respaldo en la cinta por defecto del directorio actual y todos sus subdirectorios.

```
$ tar cvf backup.tar ./usr ./etc ./bin
```

Crear el archivo backup.tar que contenga los directorios *usr*, *etc* y *bin* del directorio actual.

```
tar cvf - | gzip > ../respaldo.tar.gz
```

Crear un respaldo del directorio actual en la salida estándar, que es pasado al *gzip* para que lo comprima, y el resultado es puesto en el archivo *respaldo.tar.gz*

```
tar cvf - | ( cd /tmp/dup; tar xvf - )  
tar cvf - | tar xvCf /tmp/dup -
```

Son formas equivalentes de crear un duplicado perfecto del directorio actual en el directorio */tmp/dup*

Se recomienda no usar rutas absolutas. *Tar* no puede cambiar la ruta de un archivo al momento de recuperarlo. Por ejemplo, si un respaldo se realizó con el siguiente comando:

```
$ tar cv /usr/users/jqz
```

Al momento de recuperado, *tar* va recuperar los archivos con la misma ruta exactamente, de manera que si */usr/users/jqz* ya existía, sus archivos serán reemplazados por los del respaldo. La solución es utilizar únicamente rutas relativas al momento de crear un respaldo con *tar*. Por ejemplo, en vez del comando anterior, se puede ejecutar:

```
$ cd /usr/users
$ tar cv ./jqz
```

Con lo que el respaldo se podrá, recuperar al directorio *jqz* debajo del directorio actual, sin importar cual sea.

- **DD.** Comando que permite copiar de dispositivo a dispositivo.

**Sintaxis:** `dd [opcion=valor]`

Opciones:	if	Input file
	of	Output file
	ibs	Tamaño de bloque de entrada
	obs	Tamaño de bloque de salida
	cbs	Tamaño de bloque de conversión
	fskip	Salta archivos
	count	Número de bloques a transferir
	conv	Tipo de conversión ascii. EBCDIC --> ASCII swab. Invierte cada par de bytes,

Ejemplo:

```
# dd if=/dev/rmt0 of=/tmp/root/respaldo ibs=20 obs=20 conv=swab
# dd if=archivo1 of=archivo2
```

Copiar una cinta completa

```
if=/dev/tape of=cinta cbs=20b
if=cinta of=/dev/tape abs=20b
```

- **CPIO**

Comando que permite empaquetar archivos en un contenedor *cpio*, es decir almacena otros archivos en uno solo. El contenedor puede ser un disco, otro archivo, una cinta o una tubería. Los archivos pueden ser ordinarios, dispositivos o FS completos.

**Sintaxis:** `cpio -o [llaves]`  
`cpio -i [llaves] [patrones]`  
`cpio -p [llaves] [directorios]`

<b>Opciones:</b>	o	Genera un contenedor en la salida estándar.
	i	Toma archivos en un contenedor de la entrada estándar.
	p	Lee de la entrada estándar los nombres de los archivos que hay que copiar y los copia en el directorio especificado.
	m	Conserva los atributos de los archivos
	t	Crea una tabla de contenidos
	v	Modo verbose
	d	Especificar un directorio

Ejemplos:

```
find /var | cpio -o > /dev/tape
find . -name "*.txt" | cpio -o /tmp/textos.cpio
cpio -l < /dev/st0
find cpio -pd /tmp/nuevo
```

- **Restaurar respaldos.** Restore

**Sintaxis:** restore    opciones    argumentos [archivos y directorios]

<b>Opciones:</b>	r	lee y restaura la cinta completa
	R	selecciona una cinta de un respaldo multivolumen
	x	extrae
	f	dispositivo de entrada
	h	no recursivo
	v	verbose
	i	interactivo

Ejemplo:

```
# cd /usr/users
# restore xf //dev/rmt1 yoli otro/tmp/dos
```

## 6.6.2. Proteger las bitácoras

Para poder proteger las bitácoras del sistema es necesario resguardar los permisos de acceso y escritura que los diversos usuarios tienen, pero también se puede configurar el *syslog* para enviar una copia de los datos más importantes a un lugar seguro periódicamente.

Algunas características para verificar entradas no autorizadas a las bitácoras son:

- Bitácoras cortas o incompletas.
- Bitácoras que contienen tiempos raros o fuera de nuestro horario de trabajo.
- Bitácoras con propiedad o permisos incorrectos.
- Registros de reinicio del equipo o de servicios.
- Bitácoras perdidas.
- Entradas o "logins" de lugares extraños.

### 6.6.3. Proteger /etc/syslog.conf

*/etc/syslog.conf* este archivo le dice a *syslogd* (demonio de bitácoras de sistema) dónde anotar los varios mensajes, y así se puede saber dónde se están guardando las bitácoras del sistema viendo este archivo.

Si se necesita configurar el periodo de escritura de registros o el demonio para guardar los registros por más tiempo y poder examinarlos, para ello está el paquete del *logrotate*. Si los archivos de la bitácora han sido manipulados de alguna forma, hay que determinar cuándo empezó esta actividad, y qué ha sido manipulado.

Para proteger las bitácoras se pueden llevar a cabo las siguientes actividades:

- Es importante hacer respaldos periódicos de las bitácoras de sistema en un lugar seguro
- El demonio del *syslog* puede configurarse para enviar los datos de las bitácoras automáticamente, pero no se envía cifrado, esto puede revelar información de la red, para ello hay demonios disponibles que cifran los datos del *syslog*.
- Es fácil falsificar los mensajes del *syslog* con programas especiales.
- *Syslog* acepta entradas a las bitácoras de equipos locales sin indicar su verdadero origen.

Generalmente los intrusos modifican las bitácoras para borrar posibles rastros de su "visita", pero hay formas de detectar cuando hay intrusos hurgando en las bitácoras, es decir, descubrir al intruso en el acto, lo cual se puede hacer de la siguiente forma:

- Deducirlo en base a cambios en el sistema: cuentas nuevas, archivos nuevos, modificación
- Deducirlo en base al comportamiento del sistema: caídas frecuentes, mal desempeño, negación de servicio, etc.
- Recibir el mensaje del administrador de otro sitio reportando actividad extraña originada desde la máquina local.

Es conveniente, si posible, configurar el *syslog* para enviar una copia de los datos más importantes a un lugar seguro donde no lleguen "las manos" de los intrusos.

## 6.7. Herramientas de seguridad aplicadas

Red Hat tiene herramientas de seguridad, la gran mayoría de ellas trabajan para proteger activamente el sistema. Algunas de las más comunes y útiles herramientas son:

- Utilidades Shadow. Es una colección de herramientas para administrar usuarios locales y grupos en un sistema que utiliza contraseñas cifradas. Su configuración se puede consultar en la sección 6.4.4.3.
- Kerberos. es un sistema seguro que proporciona servicios de autenticación<sup>43</sup> de redes. No permite que contraseñas de texto común pasen sobre una red para obtener acceso a los servicios. Para mayor detalle ir a la sección 6.4.4.2.

---

<sup>43</sup> Autenticar es verificar la identidad de ser, en verdad, quien se dice ser.

- Secure Shell, OpenSSH y OpenSSL se encuentran en la sección 6.4.4.1.

La única que falta detallar es Tripwire que se explica a en los párrafos siguientes.

### Instalación de Tripwire

Los pasos a tomar para instalar, usar y mantener Tripwire apropiadamente son los siguientes:

1. Instalar Tripwire y personalizar el fichero de política. Se puede instalar con el RPM<sup>44</sup> de tripwire durante el procedimiento de instalación de Red Hat, pero si ya se instaló Red Hat, se puede instalar el RPM de Tripwire con los CD-ROMs de Red Hat, llevando a cabo los siguientes pasos:
  - a) Localice el directorio RedHat/RPMS en el CD-ROM de Red Hat.
  - b) Localice el RPM binario de tripwire tecleando `ls -l tripwire*` en el directorio RedHat/RPMS.
  - c) Teclear `rpm -Uvh <name>` (<name> es el nombre del RPM de Tripwire del paso b).
  - d) Después de haber instalado el RPM de tripwire, seguir las instrucciones de post-instalación abajo.

**Nota.** Las notas de versión y el fichero LÉAME se encuentran en `/usr/share/doc/tripwire-<version-number>`. Estos documentos contienen información importante sobre el fichero de política predeterminado y otras cuestiones

2. Personalizar la configuración de muestra (`/etc/tripwire/twcfg.txt`) y los ficheros de política (`/etc/tripwire/twpol.txt`) y ejecutar la secuencia de comandos (`/etc/tripwire/twinstall.sh`).
3. Inicializar la base de datos de Tripwire — construir una base de datos de los ficheros de sistema esenciales para supervisarlos basándose en el contenido del fichero de política Tripwire nuevo y firmado (`/etc/tripwire/tw.pol`).
4. Cuando Tripwire inicializa su base de datos, crea una colección de objetos de sistema de ficheros basada en las reglas en el fichero de política. Esta base de datos funciona como la línea base para controles de integridad. Utilizar el siguiente comando para inicializar la base de datos de Tripwire:

```
/usr/sbin/tripwire -INIT
```

Este comando puede tardar varios minutos en ejecutarse.

5. Ejecutar un control de integridad Tripwire — comparar la base de datos Tripwire recién creada con los ficheros de sistema reales en busca de ficheros modificados o desaparecidos. La ejecución de un control de integridad se hace de la siguiente manera:

---

<sup>44</sup> Es el instalador autoejecutable para los programas de instalación en UNIX.



Examinar el fichero de informes Tripwire con `twprint` para distinguir las violaciones a la integridad. Se puede imprimir un informe con el comando `twprint -m r` que muestra el contenido de un informe Tripwire en texto sin cifrar. Usted debe decirle a `twprint` cual informe poner en pantalla.

Un comando `twprint` para imprimir informes Tripwire sería semejante a lo siguiente (todo en una línea):

```
/usr/sbin/twprint -m r --twfile  
/var/lib/tripwire/report/<name>.twr
```

La opción `-m r` en el comando ordena a `twprint` que descifre el informe Tripwire.

La opción `--twfile` le ordena a `twprint` que use un fichero de informe Tripwire específico.

El nombre del informe Tripwire contiene:

- Nombre del host que Tripwire ha controlado para generar el informe
- Fecha y hora de la creación

Se pueden consultar informes guardados previamente en cualquier momento tecleando `ls /var/lib/tripwire/report` para ver una lista de informes.

Los informes Tripwire pueden ser algo largos, según la cantidad de violaciones encontrada o los errores generados.

6. Tomar medidas de seguridad adecuadas. Si los ficheros bajo supervisión han sido modificados en modo inadecuado, los puede reemplazar con los originales salvados en copias de seguridad o reinstalar el programa.
7. Actualizar el fichero de la base de datos de Tripwire. Si las violaciones a la integridad son intencionales y válidas, como si root hubiese intencionalmente modificado un fichero o reemplazado un determinado programa, se debe avisar al fichero de base de datos de Tripwire que no lo indique como violación en informes futuros. Para ello es importante actualizar de la base de datos después de un control de integridad lo cual se explica a continuación.

Actualización de la base de datos. Cuando se ejecuta un control de integridad y Tripwire encuentra violaciones, y hay que determinar si las violaciones detectadas son huecos en la seguridad o son modificaciones autorizadas. Si se ha instalado recientemente una aplicación o ha modificado ficheros de sistema esenciales, Tripwire le informará de violaciones a la integridad. En este caso se deberá actualizar la base de datos Tripwire para que esos cambios no vuelvan a aparecer en los informes como violaciones, para ello se debe especificar el informe que desea usar para actualizar su base de datos y se debe usar el informe más reciente con el siguiente comando (todo en una línea):

```
/usr/sbin/tripwire --update --twfile  
/var/lib/tripwire/report/<name>.twr
```

"name" es el nombre del informe que deber usarse. El informe se mostrará en el editor de textos vi (predeterminado, se debe especificar en el fichero de configuración Tripwire en la línea EDITOR).

Cuando se trate de cambios no autorizados a ficheros de sistema que generan violaciones al control de integridad, entonces se deberá restablecer el fichero original de una copia de seguridad o reinstalar el programa.

Las actualizaciones a la base de datos deben iniciar con una [x] antes del nombre del fichero. Se pueden dar los siguientes casos:

- Cuando se desea omitir que una violación válida sea añadida a la base de datos de Tripwire, quite la x de la casilla.
- Para aceptar como cambio cualquier fichero con la x al lado, se escribe el fichero en el editor y se sale del editor de textos.

Después de que se haya escrito una nueva base de datos de Tripwire, las violaciones a la integridad recién integradas no volverán a aparecer como advertencias cuando se ejecute el siguiente control de integridad.

8. Actualizar el fichero de política. si necesita cambiar la lista de ficheros que Tripwire supervisa, o la manera en que trata las violaciones a la integridad, debería actualizar su fichero de muestra de política (/etc/tripwire/twpol.txt), regenerar una copia firmada (/etc/tripwire/tw.pol), y actualizar su base de datos de Tripwire. Consulte la sección de nombre Actualización del fichero de política para obtener más información.

Una vez instalado, Tripwire debe además ser correctamente inicializado para poder mantener una supervisión cuidadosa de sus ficheros.

### **Instrucciones de post-instalación de Tripwire**

El RPM de Tripwire instala los ficheros de programa necesarios para ejecutar el software. Después de haber instalado Tripwire, debe configurarlo para su sistema como se describe en los siguientes pasos:

1. Si ya sabe de varios cambios que deberían efectuarse al fichero de configuración (/etc/tripwire/twcfg.txt) o al fichero de política (/etc/tripwire/twpol.txt), modifique ahora esos ficheros.

**Nota** Como se tienen que modificar los ficheros de configuración y de política para personalizar, para ello no es necesario usar Tripwire. Si tiene intenciones de modificar el fichero de configuración o el de política, deberá efectuar esos cambios antes de ejecutar la secuencia de comandos de configuración (/etc/tripwire/twinstall.sh). Si modifica el fichero de configuración o el de política después de haber ejecutado la secuencia de comandos de configuración, deberá volver a ejecutar la secuencia de comandos de configuración antes de inicializar el fichero de la base de datos. Recuerde que puede modificar los ficheros de configuración y de política después de inicializar el fichero de la base de datos y de haber ejecutado un control de integridad.

2. Teclee `/etc/tripwire/twinstall.sh` en la línea de comandos como `root` y pulse Intro para ejecutar la secuencia de comandos de configuración. La secuencia de comandos `twinstall.sh` le acompaña a través de los procedimientos para fijar frases de contraseña, generar las claves criptográficas que protegen los ficheros de configuración y de política de Tripwire, y firmar estos ficheros.

**Nota.** Una vez cifrados y firmados, el fichero de configuración (`/etc/tripwire/tw.cfg`) y el de política (`/etc/tripwire/tw.pol`) generados al ejecutar la secuencia de comandos `/etc/tripwire/twinstall.sh` no deberían ser movidos ni se les debe cambiar de nombre.

3. Inicializar el fichero de la base de datos de Tripwire utilizando el comando `/usr/sbin/tripwire --init` desde la línea de comandos.
4. Ejecutar el primer control de integridad comparando su base de datos de Tripwire nuevo con sus ficheros de sistema lanzando el comando `/usr/sbin/tripwire --check` desde la línea de comandos y buscando errores en el informe generado.

Una vez que termine estos pasos exitosamente, Tripwire tiene una "foto" de línea de base de su sistema de ficheros que necesita para revisar si hay cambios hechos a los ficheros esenciales. Además, el RPM de tripwire añade un fichero llamado `tripwire-check` en el directorio `/etc/cron.daily` que ejecutará un control de integridad automáticamente una vez al día.

Hay algunas actividades generales que se pueden hacer para proteger el sistema:

- Limitar el número de usuarios que pueden ejecutar comandos como "`root`". Ya sea intencionalmente que accidentalmente de alguien que conoce la contraseña de "`root`" o a quien ha sido dado permiso por medio de "`sudo`" para ejecutar un comando en el ámbito de "`root`".
- Saber qué paquetes de software se han instalado en el sistema y permanecer alerta para detectar huecos de seguridad, así saber qué paquetes hay que supervisar y cuándo hay que actualizarlos.
- Limitar los servicios que se ejecutan en el sistema sólo a aquellos estrictamente necesarios. Es importante conservar los recursos de sistema y desinstalar paquetes que se usan. Para ello se puede ejecutar una herramienta como "`ntsysv`" para evitar que servicios innecesarios inicien con el sistema a la hora del arranque.
- Requerir que los usuarios creen contraseñas seguras y cambiarlas a menudo o manejar las cuentas de usuario a través del administrador de sistema y dar contraseñas por defecto.
- Asegurarse que los permisos de ficheros no estén abiertos sin ser necesario, ya que la mayoría de los ficheros no deberían ser escribibles.
- Convertir en una rutina la actividad de supervisión de los registros de sistema. Red Hat guarda datos útiles en los registros de sistema situados en el directorio `/var/log`, especialmente en el fichero `messages`. Una tarea sencilla ejecutada como "`root`", como el comando `grep "session opened for user root" /var/log/messages | less`, permite desempeñar una revisión parcial del sistema y supervisar quién está en él como "`root`". Es importante considerar que este no es un método a prueba de fallos, pues alguien con el permiso de escritura en un fichero de sistema importante podría modificar el fichero `/var/log/messages` para borrar su rastro.

# CONCLUSIONES

---

Tras el proceso de aplicación de calidad y seguridad puedo decir que el sistema cumple con los requerimientos de calidad propuestos por ISO 9000, pues se logró una aplicación exitosa de sus herramientas, aunque los resultados no fueron excepcionales, se logra un nivel de calidad que puede mejorarse con las propuestas hechas en las secciones 5.5.1.3.1. y 6.3. pero es muy importante que se de un seguimiento continuo a estas pautas para que el nivel de calidad se siga acrecentando. El modelo evolutivo elegido, para desarrollo, resultó muy conveniente para la organización del equipo, ya que cada etapa pudo llevarse a cabo y cumplir así el ciclo, además que permitió que el trabajo se hiciese eficientemente.

La ingeniería de software se complementó en una manera tal con la calidad que se pudo tomar una serie de parámetros de ella para la aplicación y así lograr una mejor y más completa visión de la calidad.

Respecto a la orientación a objetos que se le dió al sistema es conveniente, ya que de esta manera se asegura que el sistema pueda tener mejoras y/o crecer en un periodo de tiempo aceptable sin que tenga que recurrirse a metodologías de programación diferentes

# BIBLIOGRAFÍA

---

- a) Mediavilla Mauriz, Manuel. Seguridad en Unix. Editorial Alfaomega. México. 1998.
- b) Fine, Leonard H. Seguridad en centros de cómputo: políticas y procedimientos. Traducción de: Computer security. A handbook for management. Editorial Trillas. México. 1994.
- c) Sommerville, Ian. Software Engineering. 6ª edición. Inglaterra. Addison – Wesley, Pearson Education. 2001.
- d) Pressman, Roger S. Ingeniería del software: Un enfoque práctico. Cuarta Edición. México. McGraw-Hill. 1998.
- e) Norton, Peter. Introducción a la computación. México. McGRAW-HILL. 1995.
- f) Booch, Grady., Análisis y Diseño Orientado a Objetos, 2a edición, Addison Wesley, 1996.
- g) Sommerville, Ian. User Interface Design. En su: Software Engineering. USA, Addison Wesley, 2001. pp. 345.
- h) Berard, E. V., Essays on Object Oriented Software Engineering, Addison – Wesley, 1993.
- i) Booch, G., Object Oriented Design, Benjammings Cummings, 1991
- j) Coad, P. y E. Yourdon, Object Oriented Analysis, 2a edición, Prentice Hall, 1991
- k) Jacobson, I., Object Oriented Software Engineering, Addison – Wesley, 1992.
- l) Rumbaugh et al., Object Oriented Modeling and Design, Prentice Hall, 1991.
- m) Wirfs-Brock, R., B. Wilkerson y L. Weiner, Designing Object Oriented Software, Prentice Hall, 1990.
- n) Meyer, Bertrand, Object Oriented Software Construction, 2ª edición, Prentice-Halls. 1998.

Páginas en internet consultadas:

- ⇒ Schools Pirvate Schools At Nober Learning Center Discover Organizational Learning Nea. Nobel Learning Communities Inc. 1992. <http://www.nobellearning.com/nobel/viewpage.cfm?pageID=2>
- ⇒ Nacional Learning Communities Project Home. Learning Communities Commons. 2001. <http://learningcommons.evergreen.edu/>
- ⇒ Cool Web design Inspiration. Cool Web design. Diciembre 2003. <http://www.cwd.dk/inspiration.asp?mode=4&siteid=3648>

- ⇒ Bienvenu Sur Le Site Du Docteur Yves Coen. Les Motivation. <http://www.dryvescohen.com/Animation2.htm>
- ⇒ L'Oeil Dentaire - Le spécialiste de la vidéo dentaire. L'Oeil Dentaire. <http://www.oeil-dentaire.com/>
- ⇒ Formation Dentaire Appliquée - Formation Continue Pour Chirurgien Dentist. FDA - Formation Dentaire Appliquée. Octobre 2003 <http://www.fda-france.com/>
- ⇒ Métodos y Herramientas. AITECO Consultores. 2003. <http://www.aiteco.com/herramie.htm>
- ⇒ Linux Security Howto. Kevin Fenzi. Enero 2004. <http://www.tldp.org/HOWTO/Security-HOWTO/index.html>
- ⇒ Análisis de los requerimientos tecnológicos para la implementación de servidores web seguros. Ing. Silvio A. Sandoval Ramos. 11 de Diciembre de 2002. <http://www.monografias.com/trabajos12/rete/rete.shtml>
- ⇒ Cool Web Design. Cool Web Design. 2001. <http://cwd.dk>
- ⇒ Políticas de seguridad. Next Vision. <http://www.nextvision.com/template.php3?sec=poleseg>
- ⇒ Segurinfo - Aplicación Web de Seguridad Informática en la Universidad de Pinar del Río. MSc. Rolando Díaz Hernández. <http://www.monografias.com/trabajos14/segur-informatica/segur-informatica.shtml#CONCL>
- ⇒ Guia De Seguridad Informatica. SEDISI. [http://www.sedisi.es/05\\_Estudios/guia02.htm](http://www.sedisi.es/05_Estudios/guia02.htm)

# ANEXOS

---

---

## ANEXO 1: ASIGNACIÓN DE TAREAS

---

---

### ANÁLISIS

- Estabilidad de los requerimientos para el diseño
- No perder de vista la calidad del producto en todas las etapas
- Examinación efectiva de todas las variantes o módulos
- Asegurar una implementación efectiva contemplando todos los requerimientos y previendo a futuro
- Identificación de requisitos funcionales
- Identificación de requisitos de comportamiento para la aplicación
- Deberán contestarse las siguientes preguntas:
  - ¿Cuál es la motivación principal para desarrollar la aplicación?
  - ¿Por qué es necesaria la aplicación?
  - ¿Quién va a utilizar la aplicación?

Se deben formular metas enfocadas al seguimiento de los requerimientos del usuario final. Las hay de dos tipos:

1. Metas Informativas. Indican la intención de proporcionar el contenido y/o información específicos para el usuario final.
2. Metas Aplicables. Indican la habilidad de realizar algunas tareas dentro de la aplicación.

Se deben hacer 4 tipos de análisis:

1. Análisis del contenido. Se trata de la identificación del aspecto completo del contenido q se va a proporcionar. En el contenido se puede incluir datos de texto, gráficas, imágenes, video y sonido.
2. Análisis de la interacción. Se trata de la descripción detallada de la interacción del usuario y la aplicación. Esto incluye todas las interacciones que pudieran darse en el sistema.
3. Análisis funcional. Los escenarios de utilización creados como parte del análisis de interacción definen las operaciones q se aplican en el contenido del sistema e implican otras funciones de procesamiento.
4. Análisis de la configuración. Se efectúa una descripción detallada del entorno y de la infraestructura en donde reside el sistema (intranet, extranet o internet).

Se debe llevar a cabo una revisión abierta del modelo de desarrollo por parte del grupo de trabajo de manera que se tome el modelo más adecuado para ese sistema. Debe pasar por:

- Discusión,
- Evaluación,
- Aprobación

## **ANÁLISIS Y DISEÑO**

### **Principios del Software**

1. Modularidad. En este punto deberá poderse:
  - a) Tener la capacidad de descomponer el sistema total en partes funcionales.
  - b) Integración del sistema completo a partir de los componentes existentes.
  - c) Comprensión del sistema observándolo en partes
  - d) Tendrá el nivel de cohesión necesario para que las partes puedan interactuar libremente, pero así mismo deberá estar preparado para la intervención de nuevos actores, eliminación de alguno de los módulos o reemplazo de cualquiera de las partes.
2. Abstracción. Proceso mediante el cual se identifican los aspectos importantes de un fenómeno y se ignoran los detalles. El cual debe ser lo más genérico posible, es decir, formar patrones que puedan ser repetibles durante el proceso que nos permita obtener el modelo.
3. Rigor. Definir y aplicar normas con precisión y exactitud durante la construcción del sistema para obtener calidad y eficiencia.
4. Formalidad. Plasmar dentro de un marco de modelos metodologías y procesos, las soluciones para el desarrollo.
5. Anticipación al cambio. Esto implica un esfuerzo especial para tratar de predecir cómo y cuánto ocurrirán los cambios y a partir de eso adaptar nuestro sistema a los cambios locales, nacionales y mundiales.

### **Principios de Calidad**

1. Usabilidad. ¿Qué hago?
  - a) Capacidad de comprensión del sitio global
  - b) Servicios de ayuda y retroalimentación en línea
  - c) Capacidades estéticas y de interfaz
  - d) Servicios especiales
2. Funcionalidad. ¿Qué mi aplicación realice las tareas para la cual fue desarrollada?
  - a) Capacidad de recuperación y búsqueda
  - b) Servicios de búsqueda y navegación
  - c) Servicios relacionados con el servicio y dominio de la aplicación
3. Fiabilidad. Eliminar errores.



- a) Proceso correcto de enlace
  - b) Recuperación de errores
  - c) Validación y recuperación de la entrada del usuario
4. Eficiencia. Rapidez, rendimiento óptimo.
- a. Rendimiento del tiempo de respuesta
  - b. Velocidad de generación de páginas
  - c. Velocidad de generación de gráficos
5. Capacidad de mantenimiento
- a. Facilidad de corrección
  - b. Adaptabilidad
  - c. Extensibilidad

### **Factores de Calidad Internos**

- Corrección. Debe estar preparado para responder a cierto grado de calidad
- Robustez. Responder a situaciones extremas como se espera. Tener cierto grado de flexibilidad.
- Extensibilidad / Evolutividad. Que se adecue a las necesidades de crecimiento. Modificarlo fácilmente de acuerdo al cambio bajo dos principios:
  - 1) Simplicidad en el diseño.
  - 2) Descentralización, lo que se traduce en modularidad.
- Reutilización. Definir un patrón de manera que pueda re-usarse el código. Definir patrones.
- Portabilidad. Transferir de un lado a otro el sistema sin importar hardware o software.
- Verificabilidad. Preparar procedimientos de prueba y fallo para detectar errores.
- Mantenimiento. Sostener al sistema que siga funcionando adecuadamente.

### **Factores de Calidad Externos**

- Rapidez. Respuesta rápida a un evento cualquiera que este fuese.
- Confiabilidad. Que la respuesta sea la esperada por el usuario.
- Interfaz. Crear una navegación fácil al usuario.
- Factibilidad de uso. Simplicidad práctica. Sencillez en la forma de realizar cualquier evento.

### **Diseño de Navegación**

Se definen las rutas de navegación q permitan al usuario acceder al contenido y a los servicios de las aplicaciones. Para que el diseñador pueda llevarlo a cabo debe:

- Identificar la semántica de la navegación para los diferentes usuarios del sitio, lo que se traduce en controles de acceso.

- Definir la mecánica de cada enlace de navegación, esto se refiere a los enlaces basados en texto, íconos, botones, etcétera.
  - Se deben elegir los enlaces de navegación adecuados para el contenido.
  - Se debe establecer las conexiones y ayudas adecuadas, esto es íconos y enlaces gráficos con aspecto "clicable".
  - Para navegación basada en texto se deberá utilizar el color que indica los enlaces de navegación.
  - Proporcionar una indicación de que se ha elegido una opción de navegación.
  - Proporcionar una indicación de los enlaces por los que se ha navegado.

### **Lineamientos Para Diseño de Pantallas**

- Mantener una pantalla sencilla
- Manejar y mantener una presentación consistente
- Facilitar los movimientos de usuario entre pantallas
  - Desplazamiento
  - Solicitud de mayor detalle
  - Dialogo de pantalla
- Crear pantallas atractivas
  - Tipo de letra
  - Imágenes
  - Color
- En las transacciones que se hagan en el sistema se debe cumplir con los siguientes puntos:
  - El sistema aceptó la entrada
  - La entrada se encuentra en forma correcta
  - La entrada no se encuentra en forma correcta
  - Habrá un retraso en el procesamiento
  - La solicitud ha sido concluida
  - El sistema es incapaz de concluir la petición

### **Lineamientos de Interfaz**

- Familiaridad del usuario
- Familiaridad con otros sistemas ya utilizados
- Consistencia. Esto es igualdad en los componentes.
- Mínima sorpresa. Que no exista un cambio muy radical en los componentes o versiones. Que el comportamiento del sistema debe ser predecible
- Recuperabilidad
  - Confirmación de acciones destructivas
  - Proveer un recurso de deshacer

- Guiar al usuario. Proporcionar ayudas. Guías de usuario. Ayuda, arreglos y su manejo.
- Diversidad de usuarios. Observar las características de todos los posibles usuarios y tratar de incorporarlas. Esto es proporcionar características adecuadas a cada tipo de usuario.
- Facilidad de uso.
  - Selección de menús, listas
  - Llenado de formularios

### **Lineamientos de Interfaz Web**

- Se deben establecer mapas de sitio
- Evitar símbolos de construcción
- Evitar "links" rotos o inservibles
- Evitar que el usuario recorra la pantalla
- El diseño no deberá depender de las funciones del navegador
- Documentación de errores, para posterior corrección antes de la liberación del sistema.
- Las opciones de navegación deben ser obvias

Para que un sistema pueda ser considerado adecuado en su interfaz de be observar los atributos de usabilidad de una interfaz que son:

- Aprendizaje. El grado en el que el usuario está obteniendo una retroalimentación del sistema.
- Velocidad de la operación. El tiempo que hay desde que el usuario está tramitando una operación hasta que esta es completada o concluida.
- Robustez.
- Recuperación
- Adaptación. Modelo de trabajo a adoptar de acuerdo al usuario.

### **Lineamientos Para la Utilización de Color en las Interfaces del Usuario**

- Limitar el número de colores utilizados y ser conservador al momento de utilizarlos
- Utilizar un cambio de color para mostrar el cambio en el estado del sistema
- Utilizar el código de colores predefinido en una forma consistente
- Ser cuidadosos al utilizar "juegos" de colores
- Manejar un grado razonable de retroalimentación para el usuario en dos sentidos:
  - Para incrementar el grado de confianza
  - Para saber la manera en que incrementa el trabajo

## **PROGRAMACIÓN**

### **Al Escoger un Lenguaje de Programación**

Al escoger un lenguaje de programación debo tomar en cuenta los siguientes puntos:

- Las características de la aplicación. Esto es escoger la herramienta indicada para el trabajo que se está observando.
- El uso externo o interno. Si es el mismo uso que se le va a dar al sistema por los usuarios que el que le va a dar el administrador del sistema.
- Complejidad. Cuán grande es el trabajo
- Lenguajes y paquetes. Definir para qué o en qué áreas del proyecto voy a usar el lenguaje y en dónde intervendrá la paquetería, cuales son las alternativas.
- Estandarización. La necesidad de lenguajes estandar que se puedan implementar con facilidad en una variedad de equipos y que se permita transportar el sistema de una computadora a otra.
- Claridad, sencillez y unidad. El lenguaje debe constituir una ayuda para el programador mucho antes de la etapa misma de codificación. Debe proveer un conjunto claro, sencillo y unificado de conceptos que se puedan usar como fundamentos en el uso del algoritmo.
- Ortogonalidad. Es el atributo de ser capaz de combinar varias características de un lenguaje en todas las combinaciones posibles de manera que todas ellas tengan significado.
- Naturalidad para la aplicación. El lenguaje deberá suministrar estructuras de datos operacionales, estructuras de control y una sintaxis natural apropiada, para el problema que se va a resolver.
- Apoyo para la abstracción. Ayuda para las operaciones abstractas que caracterizan la solución de un problema y la estructura de datos primitivos y operaciones particulares integradas en un lenguaje.
- Facilidad de escritura. Que el lenguaje de programación facilita el desarrollo limpio, conciso, y libre de errores de los programas
- Chequeo de errores. Que el lenguaje facilite el desarrollo de programas libres de errores, como son los depuradores y mensajes de compilación.
- Legibilidad. Que los programas sean escritos en cierto lenguaje que les permita ser fáciles de escribir
- Autodocumentación. Qué el código fuente por sí mismo pueda documentar el programa, ayudado de comentarios en los casos complicados.

### **Al Programar**

Para programar se debe observar lo siguiente:

- Extensibilidad. Que el código fuente se pueda extender en el programa para agregar nuevos elementos al lenguaje.
- Portabilidad. Que sea independiente de la arquitectura.
- Eficiencia. Que el uso de los recursos de cómputo y humanos sea la óptima.

- Que la eficiencia del algoritmo sea la óptima.
- El uso de los recursos de la máquina, sea el menor posible, sin limitar el rendimiento de la aplicación.
- Accesibilidad al personal interno para poder hacer uso del código resultante.
- La complejidad psicológica que afectan la habilidad del programador para:
  - Crear,
  - Modificar y,
  - Comprender el software
 Se pueden ver afectados, pero resulta indispensable para lograr un nivel de calidad satisfactorio.

### **En la Nomenclatura**

- Cada módulo programado deberá contener lo siguiente en forma clara y concisa:
  - Nombre del módulo,
  - Autor,
  - Fecha de creación,
  - Fecha de última modificación,
  - Proyecto del que forma parte,
  - Observaciones,
  - En las consultas (queries) poner una descripción breve.
  - En las funciones:
    - Descripción breve,
    - Declaración
    - parámetros
- Los nombres deben ser suficientemente descriptivos, sin llegar a ser demasiado largos.
- Los nombres deben proporcionar información sobre el tipo de entidad, programa y contexto en el que se encuentran.
- Los nombres deben referirse al qué y no al cómo.
- La nomenclatura debe ser expresiva y sencilla para facilitar la lectura y mantenimiento de los programas.
- La nomenclatura debe ser consistente y homogénea a través de los programas.
- La nomenclatura debe ser cómoda, congruente y simétrica.

## ANEXO 2: CUESTIONARIO PARA EL ANÁLISIS DE CALIDAD INTERNO 1

### Instrucciones

- Conteste el siguiente cuestionario con numeración del número 1 al 5. Donde el número 1 es el valor más bajo y el 5 el más alto.
- En las preguntas que corresponda o sea necesario conteste en el campo de comentario con **Si** o **No**.
- En las preguntas que no puedan ser contestadas rellene el campo de **N/A** (no aplica) y en el campo de **Comentario** ponga la razón por la que esa pregunta no aplica.
- El campo **Ponderación** está reservado de toda modificación. Esto es que no se tiene que anotar nada en él.
- **De antemano gracias.**

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:	Real			
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente		Elaboro:	15/07/04			
Etapa:		Análisis		Supervisor:				
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios
	1	¿Están bien definidos al momento los requerimientos para el diseño y funcionales?	5	<input checked="" type="checkbox"/>			3	No porque aun no se tiene completo el análisis y el diseño del sistema
Nota:	2	¿Hay un control de calidad establecido en esta etapa?	5	<input checked="" type="checkbox"/>			3	Estoy tomando en cuenta estándares de la IEE 830 para poder redactar la especificación de requerimientos y hacer el respectivo análisis y diseño orientado a objetos. Así como la calidad según ISO.
Nota:	3	¿Se ha previsto que el sistema tendrá credimiento futuro?	5	<input checked="" type="checkbox"/>			4	Se pretende que el sistema se pueda mantener y extender a nuevos requerimientos de la propia activad docente.
Nota:								

4	¿Cuál es la motivación principal para desarrollar la aplicación?	5				✓	4	Se necesita cubrir una necesidad de aplicación tecnológica para el proyecto de PAPIIT.
Nota:								
5	¿Por qué es necesaria la aplicación?	5				✓	4	Para brindar un servicio de calidad haciendo uso de un sistema que nos permita la interacción humano máquina y se hagan uso de las tecnologías informáticas de la UNAM en una aplicación hacia la docencia. Específicamente en la Facultad de Odontología de la FES Iztacala.
Nota:								
6	¿Quien va a utilizar la aplicación?	5				✓	4	Los usuarios que pertenecen a la comunidad de aprendizaje ya sean profesores o alumnos de la facultad de Odontología, los usuarios navegantes que solamente utilizaran servicios públicos del sistema y los administradores del sistema.
Nota:								
7	¿Se han definido las Metas Informativas?	5			✓		5	Esto está especificado en los casos de uso
Nota:								
8	¿Se han definido las Metas Aplicables?	5			✓		5	Dentro de la especificación de requerimientos
Nota:								
9	¿Ya se cuenta con el análisis del contenido?	5			✓		3	Se está trabajando en esa parte del análisis junto con los diseñadores gráficos.
Nota:								
10	¿Ya se cuenta con el análisis de la interacción?	5			✓		3	Aun se está trabajando en esa parte no se ha terminado hasta no acabar los requerimientos de usuario
Nota:								
11	¿Ya se cuenta con el análisis funcional?	5			✓		2	Aun se está trabajando
Nota:								
12	¿Ya se cuenta con el análisis de la configuración?	5			✓		0	No se tiene
Nota:								
13	¿El modelo de desarrollo ya pasó por discusión, evaluación y aprobación por el equipo completo?	5			✓		1	Esta en discusión hasta terminar el análisis y diseño
Nota:								
14	¿Se lleva a cabo la documentación de cada módulo que conforma el sistema?	5			✓		4	Se está modulando

Nota:	15	¿Se supervisa la realización de la documentación de cada módulo del sistema?	5	<input checked="" type="checkbox"/>		4	Los estoy trabajando junto con la Ing. Josefina Bárcenas
Nota:	16	¿Se realizan estudios de viabilidad del equipo que soportara el sistema?	5	<input checked="" type="checkbox"/>		5	Se hizo benchmarking y además de equipo para la plataforma de desarrollo y de implantación.
Nota:	17	¿Se cuenta con la información técnica del equipo que se va a implementar junto al sistema?	5	<input checked="" type="checkbox"/>		5	Se hizo un estudio de viabilidad para poder escoger la plataforma de desarrollo y las pruebas del sistema.
Nota:	18	¿Existe una persona encargada de autorizar los módulos del proyecto?	5	<input checked="" type="checkbox"/>		5	La Ing. Josefina Barcenas es la que coordina directamente el proyecto y aprueba o desapruaba junto con el equipo de trabajo las propuestas de desarrollo.
Nota:	19	¿Los datos principales del sistema pueden ser potables a otras plataformas?	5	<input checked="" type="checkbox"/>		5	Por eso se esta haciendo el análisis para no redundar datos y sean accesibles.
Nota:	20	¿Se cuenta con una representación grafica de los módulos que constituyen al sistema?	5	<input checked="" type="checkbox"/>		4	Es parcial la representación pero nos sirve de base para el diseño general del sistema
Nota:	21	¿Cuentan con políticas de calendarización de procesos para el desarrollo del sistema?	5	<input checked="" type="checkbox"/>		4	Se está trabajando en ello.
Nota:	22	¿Existe un control sobre la calendarización de procesos?	5	<input checked="" type="checkbox"/>		2	Estamos llevando un plan de trabajo, pero aun no lo he actualizado con los requerimientos del equipo de trabajo.
Nota:	23	¿Se estiman los tiempos de duración en base a criterios estándar?	5	<input checked="" type="checkbox"/>		1	Se está trabajando en ello.
Nota:	24	¿Estos proyectos cuentan con fechas programadas de implementación?	5	<input checked="" type="checkbox"/>		2	Se tiene planeado tentativamente la primera semana de enero
Nota:	25	¿Se llevan a cabo revisiones del sistema para determinar si aun se cumple con el objetivo del sistema?	5	<input checked="" type="checkbox"/>		5	Todavía no vamos en esa etapa.
Nota:							



	26	¿MI aplicación realiza las tareas para las cuales fue desarrollada?	5	<input checked="" type="checkbox"/>		1	Aun no acabos ni la primer etapa.
Nota:							
	27	¿Aplicó las normas de calidad con precisión y exactitud que se le encomendaron durante el desarrollo de su etapa?	5	<input checked="" type="checkbox"/>		4	Ya estamos trabajando en eso pero falta que den el visto bueno para su diseño, tanto de interfaces como los procesos que se desarrollarían de ese modulo.
Nota:							
	28	¿Se puede descomponer el sistema total en partes funcionales?	5	<input checked="" type="checkbox"/>		4	
Nota:							
	29	¿Se puede integrar del sistema completo a partir de los componentes existentes?	5	<input checked="" type="checkbox"/>		3	
Nota:							
	30	¿Se puede comprender el sistema observándolo en partes?	5	<input checked="" type="checkbox"/>		3	
Nota:							
	31	¿El sistema esta preparado para la eliminación de alguno de los módulos?	5	<input checked="" type="checkbox"/>		3	
Nota:							
	32	¿El sistema esta preparado para el reemplazo de cualquiera de las partes	5	<input checked="" type="checkbox"/>		3	
Nota:							
	33	¿Cuento con un sistema de recuperación?	5	<input checked="" type="checkbox"/>		4	Se generara una clase para manejo de errores que pueda informar que tipo de errores se cometen con mayor frecuencia para poder tomar medidas necesarias para disminuir esos errores.
Nota:							
	34	¿Cuento con un sistema de búsqueda?	5	<input checked="" type="checkbox"/>		4	Eso ya estaba previsto desde el principio y es uno de los requerimientos del sistema además se debe generar un buscador con ciertas especificaciones.
Nota:							
	35	¿Se cuenta con un sistema de eliminación de errores?	5	<input checked="" type="checkbox"/>		4	
Nota:							
	36	¿Se cuenta con un sistema de recuperación de errores?	5	<input checked="" type="checkbox"/>		4	
Nota:							

	37	¿Se cuenta con controles de acceso?	5	✓		4	Para los distintos tipos de usuario en los caso de uso, después se desarrollará.
Nota:							
	38	¿El sistema provee un recurso de deshacer?	5	✓		0	
Nota:							
	39	¿El sistema maneja confirmación de acciones destructivas?	5	✓		3	
Nota:							

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:	Estimado	Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Elabora:	Irma R. Carrión Bravo		
Etapas:		Diseño			Supervisor:	Ing. Josefina Bárcenas López Ing. José A. Domínguez		
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios
	1	¿Se lleva a cabo la documentación de cada módulo que conforma el sistema?	5	<input checked="" type="checkbox"/>			3	
Nota:								
	2	¿Se supervisa la realización de la documentación de cada módulo del sistema?	5	<input checked="" type="checkbox"/>			2	Por parte de la Ing. Josefina Bárcenas
Nota:								
	3	¿Cuentan con políticas de calendarización de procesos para el desarrollo del sistema?	5	<input checked="" type="checkbox"/>			5	El departamento de diseño se rige por un calendario semanal, mediante el cual se presentan avances cada miércoles, se someten a comentarios, y la semana siguiente se presentan nuevas ideas y modificaciones.
Nota:								
	4	¿Existe un control sobre la calendarización de procesos?	5	<input checked="" type="checkbox"/>			1	
Nota:								
	5	¿Se estiman los tiempos de duración en base a criterios estándar?	5	<input checked="" type="checkbox"/>			1	
Nota:								
	6	¿Estos proyectos cuentan con fechas programadas de implementación?	5	<input checked="" type="checkbox"/>			2	El departamento de Diseño tiene contemplado presentar una versión piloto la segunda primera de diciembre o segunda semana de enero.
Nota:								
	7	¿Se llevan a cabo revisiones del sistema para determinar si aún se cumple con el objetivo del sistema?	5	<input checked="" type="checkbox"/>			3	Nuestro objetivo es desarrollar la interfaz, cosa que hasta el momento está marchando bien, dado que se ha llevado una revisión semanal por parte del grupo de trabajo.
Nota:								
	8	¿MI aplicación realiza las tareas para las cuales fue desarrollada?	5			<input checked="" type="checkbox"/>	0	No se ha realizado una prueba piloto.
Nota:								

9	Para el diseño de pantallas de captura y consulta, ¿se tomo en cuenta al usuario?	5	<input checked="" type="checkbox"/>			5	Se realizó un análisis 4. Ubicación del usuario potencial (maestros y alumnos de la carrera de Odontología) 5. Análisis del entorno sociocultural del usuario 6. Detección de necesidades y expectativas del usuario sobre el sistema e implementación de soluciones para estas Esto realizado por parte del quipo de diseño como una investigación informativa más que indagatoria.
Nota:	No existen pantalla definidas hasta este momento.						
10	¿Existe ergonomía en las pantallas?	5	<input checked="" type="checkbox"/>			0	
Nota:	Se realizó un análisis de páginas en línea relacionadas con comunidades de aprendizaje y portales de odontología. Ver documento de benchmarking						
11	¿Se basó en algún prototipo o estándar quien diseñó las pantallas de captura y consulta?	5	<input checked="" type="checkbox"/>			3	
Nota:	Se han observado en algunos casos y en otros no, puesto que no se ha terminado el diseño de algunos módulos.						
12	¿Aplicó las normas de calidad con precisión y exactitud que se le encomendaron durante el desarrollo de su etapa?	5	<input checked="" type="checkbox"/>			1	
Nota:	Ver Nota 1						
13	¿La navegación es fácil al usuario?	5	<input checked="" type="checkbox"/>			1	
Nota:	Ver Nota 2						
14	¿Hay sencillez en la forma de realizar cualquier evento?	5	<input checked="" type="checkbox"/>			0	
Nota:	Ver Nota 2						
15	¿Se cuenta con una mecánica definida para cada enlace de navegación?	5	<input checked="" type="checkbox"/>			4	
Nota:	Está contemplado manejar un aspecto funcional y la vez sencillo, esto incluye a los enlaces.						
16	¿Son los enlaces de navegación adecuados para el contenido?	5	<input checked="" type="checkbox"/>			0	
Nota:	Los botones principales (barra de navegación superior) cuentan con distintivo que cambia a color rojo cuando se pasa el puntero por encima (rollover).						
17	¿Los enlaces de navegación están indicados con algún color?	5	<input checked="" type="checkbox"/>			5	
Nota:							

	18	¿Se proporciona una indicación de que se ha elegido una opción de navegación?	5			✓	1	El vínculo activo es color rojo (#ff0000) .
Nota:								
	19	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5			✓	1	Cuando el enlace haya sido visitado, cambiará a color gris (#666666).
Nota:								
	20	¿Se mantiene una pantalla sencilla?	5		✓		3	La propuesta es manejar una misma pantalla (estilo de imágenes, posición de botones y enlaces, etc.) de pantalla a pantalla para no realizar cambios radicales en el entorno que se le presenta al usuario.
Nota:								
	21	¿Se mantiene una pantalla consistente?	5		✓		3	Se trata de manejar plantillas con las imágenes y sus derivados y logotipo en lo mayor posible dentro de las pantallas.
Nota:								
	22	¿Los movimientos de usuario entre pantallas son fáciles?	5			✓	4	No se puede evaluar hasta las pruebas alfa.
Nota:								
	23	¿Las pantallas son lo suficientemente atractivas en tipo de letra, imágenes y color?	5		✓		4	Creemos que sí, pues con los análisis del benchmarking pudimos observar puntos clave que le agradan al usuario. Tomamos en cuenta la estructura de la página.
Nota:								
	24	¿El sistema incorpora características adecuadas a cada tipo de usuario?	5			✓	1	Estamos en desacuerdo con la personalización del sistema. Se toman en cuenta las características relevantes de cada grupo de usuario del sistema y se toman como parámetro para realizar la propuesta final de la interfaz de sistema.
Nota:								
	25	¿Se cuenta con un mapa de sitio?	5		✓		3	No en su totalidad al momento, pero si se tiene contemplado dentro de la propuesta.
Nota:								
	26	¿Hay símbolos de construcción en el sitio?	5			✓	5	
Nota:								
	27	¿Hay links rotos o Inservibles?	5			✓	5	
Nota:								
	28	¿El número de colores utilizados es mayor a 7?	5			✓	5	
Nota:								

	29	¿Se utiliza un cambio de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>		5	Ver Nota 2
Nota:							
	30	¿Se utilizó el mismo código de colores predefinido en todo la aplicación consistentemente?	5	<input checked="" type="checkbox"/>		5	
Nota:							
	31	¿El usuario se puede familiarizar con facilidad con la interfaz?	5	<input checked="" type="checkbox"/>		0	Se está trabajando en ello.
Nota:							
	32	¿El usuario relaciona la aplicación con otros sistemas ya utilizados?	5		<input checked="" type="checkbox"/>	0	El usuario que está familiarizado con el uso de Internet ya conoce el protocolo de navegación. Sin embargo, se realizó un esfuerzo para considerar los diferentes grados de experiencia en navegación de los usuarios del sistema.
Nota:							
	33	¿Se percibe igualdad en los componentes?	5	<input checked="" type="checkbox"/>		3	
Nota:							
	34	¿Existen cambios muy radicales en los componentes o versiones?	5		<input checked="" type="checkbox"/>	3	
Nota:							

## NOTAS

**Nota 1.** -Primeramente se diferenciaron los tipos de usuario que pueden visitar el sistema, usuarios con acceso total, alumnos con acceso limitado y visitantes en general, que deben tener acceso a la información general y básica presentada en el sistema, y que también, pueden solicitar su ingreso a la comunidad mediante inscripción.

## Nota 2.

- Los botones, zonas activas y enlaces se identifican mediante un manejo de colores y animación.
- Los botones principales (barra de navegación superior) cuentan con distintivo que cambia a color rojo cuando se pasa el puntero por encima (rollover).
- Los enlaces de texto idealmente, el color del enlace será azul claro (#0066ff).
- Cuando se coloque el puntero arriba, el texto cambiará a negritas (mismo estilo, mismo color).
- Los botones de la barra de navegación tienen aspecto tridimensional (relieve) para referir que en ese lugar debe existir un clic.

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Estimado		15/07/04	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Fecha:		02/11/2003	
Etapas:		Desarrollo y programación			Elaboro:		Irma R. Carrión Bravo	
					Supervisor:		Ing. Josefina Bárcenas López Ing. José A. Domínguez	
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios
	1	¿Se lleva a cabo la documentación de cada módulo que conforma el sistema?	5	<input checked="" type="checkbox"/>			4	
Nota:	2	¿Se supervisa la realización de la documentación de cada módulo del sistema?	5		<input checked="" type="checkbox"/>		2	Si se supervisara la documentación del sistema por la Ing. Josefina Barcenas y la encargada de calidad, de manera que pueda servir para futuras revisiones.
Nota:	3	¿Cuentan con políticas de calendarización de procesos para el desarrollo del sistema?	5		<input checked="" type="checkbox"/>		0	Se está trabajando en ello.
Nota:	4	¿Existe un control sobre la calendarización de procesos?	5		<input checked="" type="checkbox"/>		0	Se está trabajando en ello.
Nota:	5	¿Se estiman los tiempos de duración en base a criterios estándar?	5		<input checked="" type="checkbox"/>		0	Se está trabajando en ello.
Nota:	6	¿Estos proyectos cuentan con fechas programadas de implementación?	5	<input checked="" type="checkbox"/>			4	La Primera semana de enero del 2004.
Nota:	7	¿Se llevan a cabo revisiones del sistema para determinar si aún se cumple con el objetivo del sistema?	5		<input checked="" type="checkbox"/>		0	Aun no se comienza con el desarrollo.
Nota:	8	¿MI aplicación realiza las tareas para las cuales fue desarrollada?	5		<input checked="" type="checkbox"/>		0	Aun no se comienza con el desarrollo, por lo tanto no se puede calificar este punto
Nota:	9	¿Aplicó las normas de calidad con precisión y exactitud que se le encomendaron durante el desarrollo de su etapa?	5	<input checked="" type="checkbox"/>			4	Si, se usará una nomenclatura para el desarrollo de los módulos con la información necesaria.
Nota:	10	¿Está documentada la función de cada programa?	5	<input checked="" type="checkbox"/>			5	

11	¿Se establecieron estándares para los archivos y/o datos que maneja el sistema?	5	✓			2	Hasta el momento se tienen contemplados los que se manejan en las especificaciones de calidad pues apenas está por hacerse.
Nota:							
12	¿Se tiene contemplado el manejo de errores dentro del sistema?	5	✓			5	Se generara una clase para manejo de errores que pueda informar que tipo de errores se cometen con mayor frecuencia para poder tomar medidas necesarias para disminuir esos errores.
Nota:							
13	¿Existen procedimientos para identificar errores o condiciones defectuosas?	5	✓			5	Si, pero de momento aun no se hace el análisis de como funcionarían.
Nota:							
14	¿El sistema cuenta con mensajes de error o de condición defectuosa?	5	✓			5	Se tiene planeado que así sea, se implementarán ya que es necesario para el usuario.
Nota:							
15	En la ejecución de procesos del sistema, ¿existe alguna prioridad?	5	✓			5	Si, primero están los procesos del sistema (demonios) después los procesos del sistema y al final están los procesos de usuarios.
Nota:							
16	¿Se tiene establecido el tiempo de respuesta al introducir y recibir datos finales?	5	✓			5	
Nota:							
17	¿Existe algún procedimiento para el formato de los datos?	5	✓			5	Será conforme a los requerimientos del sistema.
Nota:							
18	En la ejecución de procesos del sistema, ¿existe alguna prioridad?	5	✓			5	
Nota:							
19	Si se presenta un error e incongruencia, ¿se sigue algún procedimiento para resolverlo?	5	✓			5	Se comunicara del error, pero aun no determinamos que procesos pueden contar con un procedimiento para resolver los errores.
Nota:							
20	¿Existe validación de datos?	5	✓			5	Validar por medio de una clase encargada para cada caso y sus métodos.
Nota:							
21	¿Los datos de salida son controlados para evitar incongruencias?	5	✓			5	Se usan acciones de confirmación.
Nota:							
22	¿Existe un responsable de validar la salida de las operaciones o módulos?	5	✓			1	La clase encargada de ella
Nota:							



	23	¿Le es enviada a alguna persona la salida de las operaciones o módulos una vez validados?	5	<input checked="" type="checkbox"/>					5	Al usuario y a la base de datos.
Nota:										
	24	¿Al programar se forman patrones repetibles?	5	<input checked="" type="checkbox"/>					2	Conexiones a bases de datos, consultas, manejo de errores.
Nota:										
	25	¿La respuesta del sistema es inesperada (Impredecible por el usuario)?	5	<input checked="" type="checkbox"/>					5	
Nota:										
	26	¿El tiempo de respuesta, generación de páginas y generación de gráficos, es tolerable (preferentemente menor a 10 segundos)?	5	<input checked="" type="checkbox"/>					5	
Nota:										
	27	En las transacciones que hace el sistema indica que:								
Nota:										
	27.1	¿El sistema aceptó la entrada?	5	<input checked="" type="checkbox"/>					5	
Nota:										
	27.2	¿La entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>					5	
Nota:										
	27.3	¿La entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>					5	
Nota:										
	27.4	¿Habrá un retraso en el procesamiento?	5	<input checked="" type="checkbox"/>					1	
Nota:										
	27.5	¿La solicitud ha sido concluida?	5	<input checked="" type="checkbox"/>					4	
Nota:										
	27.6	¿El sistema es incapaz de concluir la petición?	5	<input checked="" type="checkbox"/>					4	
Nota:										
	28	¿Hay un control de calidad establecido en esta etapa?	5	<input checked="" type="checkbox"/>					5	Aun no
Nota:										
	29	¿El lenguaje de programación es muy complejo?	5	<input checked="" type="checkbox"/>					3	
Nota:										
	30	¿Se definió en qué, para qué o en qué áreas del proyecto voy a usar el lenguaje?	5	<input checked="" type="checkbox"/>					4	
Nota:										
	31	¿Se definieron otras alternativas?	5	<input checked="" type="checkbox"/>					4	Pero se descartaron de inmediato
Nota:										
	32	¿Se escogieron lenguajes estándar?	5	<input checked="" type="checkbox"/>					5	
Nota:										

33	¿El lenguaje de programación es claro, sencillo y es unificado en sus conceptos?	5	✓					5
Nota:								
34	¿El lenguaje suministra estructuras de datos operacionales, estructuras de control y una sintaxis natural apropiada, para el problema que se va a resolver?	5	✓					5
Nota:								
35	¿El lenguaje de programación facilita el desarrollo limpio, conciso, y libre de errores de los programas?	5	✓					4
36	¿Los programas escritos en ese lenguaje, les permite ser fáciles de escribir?	5	✓					5
Nota:								
37	¿El lenguaje facilita el desarrollo de programas libres de errores, como son los depuradores y mensajes de compilación?	5	✓					5
Nota:								
38	¿El código fuente por sí mismo puede documentar el programa, ayudado de comentarios en los casos complicados?	5	✓					5
Nota:								
39	¿El código fuente se puede extender en el programa para agregar nuevos elementos al lenguaje?	5	✓					4
Nota:								
40	¿El código fuente es independiente de la arquitectura?	5	✓					5
Nota:								
41	¿El uso de los recursos de cómputo y humanos es el óptimo?	5	✓					5
Nota:								
42	¿La eficiencia del algoritmo es la óptima?	5	✓					5
Nota:								
43	¿El uso de los recursos de la máquina, es el menor posible, sin limitar el rendimiento de la aplicación?	5	✓					4
Nota:								
44	¿El personal autorizado puede hacer uso del código fuente?	5	✓					5
Nota:								
								Solo en términos y condiciones específicas, por ejemplo, para rehacer sistemas externos no se podrá tener acceso, para copiar código, o simplemente por lo que no se podrá, pero para modificar, es obvio.

Nota:	45	¿Cada módulo programado contiene lo siguiente en forma clara y concisa?							
Nota:	45.1	Nombre del módulo	5	✓				5	
Nota:	45.2	Autor	5	✓				5	
Nota:	45.3	Fecha de creación	5	✓				5	
Nota:	45.4	Fecha de última modificación	5	✓				5	
Nota:	45.5	Proyecto del que forma parte	5	✓				5	
Nota:	45.6	Observaciones	5	✓				5	
Nota:	45.7	En las consultas (queries) ponen una descripción breve	5	✓				3	Tales descripciones también deberán encontrarse en la parte del diseño.
Nota:	45.8	En las funciones:							
Nota:	45.8.1	Descripción breve	5	✓				5	
Nota:	45.8.2	Declaración	5	✓				3	Se encontrarán en el análisis y diseño
Nota:	46	Parámetros	5	✓				3	
Nota:	47	¿Los nombres son suficientemente descriptivos, sin llegar a ser demasiado largos?	5	✓				5	Deberá ser así, pero es dependiendo del análisis.
Nota:	48	¿Los nombres proporcionan información sobre el tipo de entidad, programa y contexto en el que se encuentran?	5	✓				5	Deberá ser así, pero es dependiendo del análisis.
Nota:	49	¿Los nombres se refieren al qué y no al cómo?	5	✓				5	
Nota:	50	¿La nomenclatura es expresiva y sencilla para facilitar la lectura y mantenimiento de los programas?	5	✓				5	
Nota:									

51	¿La nomenclatura es consistente y homogénea a través de los programas?	5	<input checked="" type="checkbox"/>					5
52	¿La nomenclatura es cómoda, congruente y simétrica?	5	<input checked="" type="checkbox"/>					5
Nota:								
Nota:								

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina				Estimado	02/11/2003	Real	15/07/04
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente				Fecha:	Irma R. Carrión Bravo		
Etapas:		Pruebas y Mantenimiento				Elaboro:	Ing. Josefina Bárcenas López		
						Supervisor:	Ing. José A. Domínguez		
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios	
	1	¿Se planean instalaciones de prueba de módulos o de sistemas?	5	✓			4		
Nota:	2	¿Se someten a pruebas rigurosas los módulos y el sistema ya estructurado?	5	✓			0	No está algún módulo 100% listo	
Nota:	3	¿Las pruebas realizadas corresponden a problemas que se tienen que enfrentar en la vida real?	5	✓			0	Deben ser hechas así en su momento	
Nota:	4	¿Existe alguien responsable de realizar estas pruebas?	5	✓			5	Carrión Bravo Irma Regina con supervisión de la Ing. Josefina Bárcenas López	
Nota:	5	¿Se realiza la documentación de estas pruebas?	5	✓			3	Deben ser hechas así	
Nota:	6	¿Existe alguien que evalúe el resultado de estas pruebas?	5	✓			5	Ing. Josefina Bárcenas López	
Nota:	7	¿Cuentan con manuales de operación, usuario y sistema?	5	✓			0	Deben ser desarrollados por la encargada del análisis	
Nota:	8	¿Cuentan con el personal capacitado para el desarrollo adecuado de los manuales de operación, usuario y sistema?	5	✓			4		
Nota:	9	¿Tienen contemplado el manejo de errores dentro del sistema?	5	✓			5		
Nota:	10	¿Existen procedimientos para identificar errores o condiciones defectuosas?	5	✓			5	Las pruebas	
Nota:									

11	¿Existe concordancia de la representación gráfica con los módulos ya implementados?	5			✓	0	No está algún módulo 100% listo
Nota:							
12	¿Existe un área que le ofrezca atención a usuarios?	5	✓			5	La Ing. Josefina Bárcenas
Nota:							
13	¿Se capacita a los usuarios del sistema?	5			✓	0	No se ha contemplado
Nota:							
14	¿Se han preparado procedimientos de prueba y fallo para detectar errores?	5	✓			3	Se está trabajando en ello
Nota:							
15	¿Se han preparado procedimientos que permitan que el sistema siga funcionando adecuadamente?	5	✓			3	Se está trabajando en ello
Nota:							
16	¿Hay un control de calidad establecido en esta etapa?	5	✓			3	Se está trabajando en ello
Nota:							

## ANEXO 3: CUESTIONARIO PARA EL ANÁLISIS DE CALIDAD INTERNO 2

### Instrucciones

- Conteste el siguiente cuestionario con numeración del número **1** al **5**. Donde el número **1** es el valor más bajo y el **5** el más alto.
- En las preguntas que corresponda o sea necesario conteste en el campo de comentario con **Si** o **No**.
- En las preguntas que no puedan ser contestadas rellene el campo de **N/A** (no aplica) y en el campo de **Comentario** ponga la razón por la que esa pregunta no aplica.
- Los campos **Referencia**, **Ponderación** y **Nota** están reservados de toda modificación. Esto es que no se tiene que anotar nada en él.
- **De antemano gracias.**

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:	Estimado		Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis Calidad		Elabora:	01/08/2004		10/09/2004	
Etapa:		Análisis		Supervisor:	Ing. Josefina Bárcenas López Ing. José A. Dominguez			
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios
	1	¿Están bien definidos al momento los requerimientos para el diseño y funcionales?	5	<input checked="" type="checkbox"/>			3	
Nota:	2	¿Hay un control de calidad establecido en esta etapa?	5	<input checked="" type="checkbox"/>			5	
Nota:	3	¿Se ha previsto que el sistema tendrá crecimiento futuro?	5	<input checked="" type="checkbox"/>			4	
Nota:	4	¿Se tiene una motivación principal para desarrollar la aplicación? ¿Cuál es?	5	<input checked="" type="checkbox"/>			5	Se necesita cubrir una necesidad de aplicación tecnológica para el proyecto de PAPIIT.
Nota:								

	5	¿Por qué es necesaria la aplicación?	5			✓	5	Para brindar un servicio de calidad haciendo uso de un sistema que nos permita la interacción humano máquina y se hagan uso de las tecnologías informáticas de la UNAM en una aplicación hacia la docencia. Específicamente en la Facultad de Odontología de la FES Iztacala.
<b>Nota:</b>	6	¿Quien va a utilizar la aplicación?	5			✓	5	Los usuarios que pertenecen a la comunidad de aprendizaje ya sean profesores o alumnos de la facultad de Odontología, los usuarios navegantes que solamente utilizaran servicios públicos del sistema y los administradores del sistema.
<b>Nota:</b>	7	¿Se han definido las Metas Informativas?	5		✓		5	Esto esta especificado en los casos de uso
<b>Nota:</b>	8	¿Se han definido las Metas Aplicables?	5		✓		5	Dentro de la especificación de requerimientos
<b>Nota:</b>	9	¿Ya se cuenta con el análisis del contenido?	5		✓		5	
<b>Nota:</b>	10	¿Ya se cuenta con el análisis de la interacción?	5		✓		3	
<b>Nota:</b>	11	¿Ya se cuenta con el análisis funcional?	5		✓		3	
<b>Nota:</b>	12	¿Ya se cuenta con el análisis de la configuración?	5		✓		0	No se tiene
<b>Nota:</b>	13	¿El modelo de desarrollo ya pasó por discusión, evaluación y aprobación por el equipo completo?	5		✓		3	
<b>Nota:</b>	14	¿Se lleva a cabo la documentación de cada modulo que conforma el sistema?	5		✓		3	
<b>Nota:</b>	15	¿Se supervisa la realización de la documentación de cada modulo del sistema?	5		✓		4	Por parte de la Ing. Josefina Bárcenas
<b>Nota:</b>								



	16	¿Se realizan estudios de viabilidad del equipo que soportara el sistema?	5	<input checked="" type="checkbox"/>				5
Nota:								
	17	¿Se cuenta con la información técnica del equipo que se va a implementar junto al sistema?	5	<input checked="" type="checkbox"/>				5
Nota:								
	18	¿Existe una persona encargada de autorizar los módulos del proyecto?	5	<input checked="" type="checkbox"/>				5
Nota:								
	19	¿Los datos principales del sistema pueden ser portables a otras plataformas?	5	<input checked="" type="checkbox"/>				5
Nota:								
	20	¿Se cuenta con una representación grafica de los módulos que constituyen al sistema?	5	<input checked="" type="checkbox"/>				3
Nota:								
	21	¿Cuentan con políticas de calendarización de procesos para el desarrollo del sistema?	5	<input checked="" type="checkbox"/>				3
Nota:								
	22	¿Existe un control sobre la calendarización de procesos?	5	<input checked="" type="checkbox"/>				3
Nota:								
	23	¿Se estiman los tiempos de duración en base a criterios estándar?	5	<input checked="" type="checkbox"/>				3
Nota:								
	24	¿Estos proyectos cuentan con fechas programadas de implementación?	5	<input checked="" type="checkbox"/>				5
Nota:								
	25	¿Se llevan a cabo revisiones del sistema para determinar si aún se cumple con el objetivo del sistema?	5	<input checked="" type="checkbox"/>				5
Nota:								
	26	¿MI aplicación realiza las tareas para las cuales fue desarrollada?	5	<input checked="" type="checkbox"/>				4
Nota:								
	27	¿Aplicó las normas de calidad con precisión y exactitud que se le encomendaron durante el desarrollo de su etapa?	5	<input checked="" type="checkbox"/>				3
Nota:								
	28	¿Se puede descomponer el sistema total en partes funcionales?	5	<input checked="" type="checkbox"/>				4
Nota:								

	29	¿Se puede integrar del sistema completo a partir de los componentes existentes?	5	<input checked="" type="checkbox"/>		4
Nota:						
	30	¿Se puede comprender el sistema observándolo en partes?	5	<input checked="" type="checkbox"/>		4
Nota:						
	31	¿El sistema esta preparado para la eliminación de alguno de los módulos?	5	<input checked="" type="checkbox"/>		4
Nota:						
	32	¿El sistema esta preparado para el reemplazo de cualquiera de las partes	5	<input checked="" type="checkbox"/>		4
Nota:						
	33	¿Cuento con un sistema de recuperación?	5	<input checked="" type="checkbox"/>		4
Nota:						
	34	¿Cuento con un sistema de búsqueda?	5	<input checked="" type="checkbox"/>		4
Nota:						
	35	¿Se cuenta con un sistema de eliminación de errores?	5	<input checked="" type="checkbox"/>		4
Nota:						
	36	¿Se cuenta con un sistema de recuperación de errores?	5	<input checked="" type="checkbox"/>		4
Nota:						
	37	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>		4
Nota:						
	38	¿El sistema provee un recurso de deshacer?	5	<input checked="" type="checkbox"/>		3
Nota:						
	39	¿El sistema maneja confirmación de acciones destructivas?	5	<input checked="" type="checkbox"/>		3
Nota:						

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:		01/08/2004		Estimado		Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis Calked			Elaboro:		Irma R. Carrión Bravo					
Etapas:		Diseño de interfaz			Supervisor:		Ing. Josefina Bárcenas López Ing. José A. Domínguez					
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios				
	1	¿Se lleva a cabo la documentación de cada modulo que conforma el sistema?	5	<input checked="" type="checkbox"/>			4					
Nota:												
	2	¿Se supervisa la realización de la documentación de cada modulo del sistema?	5	<input checked="" type="checkbox"/>			4	Por parte de la Ing. Josefina Bárcenas				
Nota:												
	3	¿Cuentan con políticas de calendarización de procesos para el desarrollo del sistema?	5	<input checked="" type="checkbox"/>			5					
Nota:												
	4	¿Existe un control sobre la calendarización de procesos?	5	<input checked="" type="checkbox"/>			3					
Nota:												
	5	¿Se estiman los tiempos de duración en base a criterios estándar?	5	<input checked="" type="checkbox"/>			3					
Nota:												
	6	¿Estos proyectos cuentan con fechas programadas de implementación?	5	<input checked="" type="checkbox"/>			5					
Nota:												
	7	¿Se llevan a cabo revisiones del sistema para determinar si aún se cumple con el objetivo del sistema?	5	<input checked="" type="checkbox"/>			4					
Nota:												
	8	¿MI aplicación realiza las tareas para las cuales fue desarrollada?	5	<input checked="" type="checkbox"/>			5					
Nota:												
	9	Para el diseño de pantallas de captura y consulta, ¿se tomo en cuenta al usuario?	5	<input checked="" type="checkbox"/>			5					
Nota:												
	10	¿Existe ergonomía en las pantallas?	5	<input checked="" type="checkbox"/>			4					
Nota:												
	11	¿Se basó en algún prototipo o estándar quien diseñó las pantallas de captura y consulta?	5	<input checked="" type="checkbox"/>			5					
Nota:												

12	¿Aplicó las normas de calidad con precisión y exactitud que se le encomendaron durante el desarrollo de su etapa?	5	✓		4
Nota:					
13	¿La navegación es fácil al usuario?	5	✓		4 Ver Nota 1
Nota:					
14	¿Hay sencillez en la forma de realizar cualquier evento?	5	✓		4 Ver Nota 2
Nota:					
15	¿Se cuenta con una mecánica definida para cada enlace de navegación?	5	✓		4 Ver Nota 2
Nota:					
16	¿Son los enlaces de navegación adecuados para el contenido?	5	✓		4
Nota:					
17	¿Los enlaces de navegación están indicados con algún color?	5	✓		5
Nota:					
18	¿Se proporciona una indicación de que se ha elegido una opción de navegación?	5	✓		5
Nota:					
19	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	✓		5
Nota:					
20	¿Se mantiene una pantalla sencilla?	5	✓		4
Nota:					
21	¿Se mantiene una pantalla consistente?	5	✓		4
Nota:					
22	¿Los movimientos de usuario entre pantallas son fáciles?	5	✓		4
Nota:					
23	¿Las pantallas son lo suficientemente atractivas en tipo de letra, imágenes y color?	5	✓		4
Nota:					
24	¿El sistema incorpora características adecuadas a cada tipo de usuario?	5	✓		3
Nota:					
25	¿Se cuenta con un mapa de sitio?	5	✓		3
Nota:					
26	¿Hay símbolos de construcción en el sitio?	5	✓		5
Nota:					
27	¿Hay links rotos o inservibles?	5	✓		5

Nota:	28	¿El número de colores utilizados es mayor a 7?	5	<input checked="" type="checkbox"/>	5
Nota:	29	¿Se utiliza un cambio de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>	5 Ver Nota 2
Nota:	30	¿Se utilizó el mismo código de colores predefinido en todo la aplicación consistentemente?	5	<input checked="" type="checkbox"/>	5
Nota:	31	¿El usuario se puede familiarizar con facilidad con la interfaz?	5	<input checked="" type="checkbox"/>	4
Nota:	32	¿El usuario relaciona la aplicación con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>	3
Nota:	33	¿Se percibe igualdad en los componentes?	5	<input checked="" type="checkbox"/>	4
Nota:	34	¿Existen cambios muy radicales en los componentes o versiones?	5	<input checked="" type="checkbox"/>	0 Es la primera versión.
Nota:					

**Nota 1.** -Primeramente se diferenciaron los tipos de usuario que pueden visitar el sistema, usuarios con acceso total, alumnos con acceso limitado y visitantes en general, que deben tener acceso a la información general y básica presentada en el sistema, y que también, pueden solicitar su ingreso a la comunidad mediante inscripción.

**Nota 2.**

- Los botones, zonas activas y enlaces se identifican mediante un manejo de colores y animación.
- Los botones principales (barra de navegación superior) cuentan con distintivo que cambia a color rojo cuando se pasa el puntero por encima (rollover).
- Los enlaces de texto idealmente, el color del enlace será azul claro (#0066ff).
- Cuando se coloque el puntero arriba, el texto cambiará a negritas (mismo estilo, mismo color).
- Los botones de la barra de navegación tienen aspecto tridimensional (relieve) para referir que en ese lugar debe existir un clic.

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:	Estimado		Real
Nombre del Cuestionario:		Cuestionario Para el Análisis Calidad		Elaboro:	01/08/2004		10/09/2004
Etapas:		Diseño, Programación, Implementación y Pruebas.		Supervisor:	Ing. Josefina Bárcenas López		
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.
	1	¿Se lleva a cabo la documentación de cada modulo que conforma el sistema?	5	<input checked="" type="checkbox"/>			4
Nota:	2	¿Se supervisa la realización de la documentación de cada modulo del sistema?	5	<input checked="" type="checkbox"/>			4
Nota:	3	¿Cuentan con políticas de calendarización de procesos para el desarrollo del sistema?	5	<input checked="" type="checkbox"/>			3
Nota:	4	¿Existe un control sobre la calendarización de procesos?	5	<input checked="" type="checkbox"/>			4
Nota:	5	¿Se estiman los tiempos de duración en base a criterios estándar?	5	<input checked="" type="checkbox"/>			4
Nota:	6	¿Estos proyectos cuentan con fechas programadas de implementación?	5	<input checked="" type="checkbox"/>			5
Nota:	7	¿Se llevan a cabo revisiones del sistema para determinar si aún se cumple con el objetivo del sistema?	5	<input checked="" type="checkbox"/>			5
Nota:	8	¿MI aplicación realiza las tareas para las cuales fue desarrollada?	5	<input checked="" type="checkbox"/>			4
Nota:	9	¿Aplicó las normas de calidad con precisión y exactitud que se le encomendaron durante el desarrollo de su etapa?	5	<input checked="" type="checkbox"/>			4
Nota:	10	¿Está documentada la función de cada programa?	5	<input checked="" type="checkbox"/>			5
Nota:	11	¿Se establecieron estándares para los archivos y / o datos que maneja el sistema?	5	<input checked="" type="checkbox"/>			5
Nota:	12	¿Se tiene contemplado el manejo de errores dentro del sistema?	5	<input checked="" type="checkbox"/>			5

Nota:	13	¿Existen procedimientos para identificar errores o condiciones defectuosas?	5	<input checked="" type="checkbox"/>				4
Nota:	14	¿El sistema cuenta con mensajes de error o de condición defectuosa?	5	<input checked="" type="checkbox"/>				4
Nota:	15	En la ejecución de procesos del sistema, ¿existe alguna prioridad?	5	<input checked="" type="checkbox"/>				5
Nota:	16	¿Se tiene establecido el tiempo de respuesta al introducir y recibir datos finales?	5	<input checked="" type="checkbox"/>				5
Nota:	17	¿Existe algún procedimiento para el formato de los datos?	5	<input checked="" type="checkbox"/>				5
Nota:	18	En la ejecución de procesos del sistema, ¿existe alguna prioridad?	5	<input checked="" type="checkbox"/>				5
Nota:	19	Si se presenta un error e incongruencia, ¿se sigue algún procedimiento para resolverlo?	5	<input checked="" type="checkbox"/>				5
Nota:	20	¿Existe validación de datos?	5	<input checked="" type="checkbox"/>				5
Nota:	21	¿Los datos de salida son controlados para evitar incongruencias?	5	<input checked="" type="checkbox"/>				5
Nota:	22	¿Existe un responsable de validar la salida de las operaciones o módulos?	5	<input checked="" type="checkbox"/>				5
Nota:	23	¿Le es enviada a alguna persona la salida de las operaciones o módulos una vez validados?	5	<input checked="" type="checkbox"/>				5
Nota:	24	¿Al programar se forman patrones repetibles?	5	<input checked="" type="checkbox"/>				5
Nota:	25	¿La respuesta del sistema es inesperada (impredecible por el usuario)?	5	<input checked="" type="checkbox"/>				5
Nota:	26	¿El tiempo de respuesta, generación de páginas y generación de gráficos, es tolerable (preferentemente menor a 10 segundos)?	5	<input checked="" type="checkbox"/>				5
Nota:								

	27	En las transacciones que hace el sistema indica que:						
Nota:	27.1	¿El sistema aceptó la entrada?	5	✓				5
Nota:	27.2	¿La entrada se encuentra en forma correcta?	5	✓				5
Nota:	27.3	¿La entrada no se encuentra en forma correcta?	5	✓				5
Nota:	27.4	¿Habrá un retraso en el procesamiento?	5	✓				3
Nota:	27.5	¿La solicitud ha sido concluida?	5	✓				5
Nota:	27.6	¿El sistema es incapaz de concluir la petición?	5	✓				5
Nota:	28	¿Hay un control de calidad establecido en esta etapa?	5	✓				5
Nota:	29	¿El lenguaje de programación es muy complejo?	5	✓				5
Nota:	30	¿Se definió en qué, para qué o en qué áreas del proyecto voy a usar el lenguaje?	5	✓				5
Nota:	31	¿Se definieron otras alternativas?	5	✓				5
Nota:	32	¿Se escogieron lenguajes estándar?	5	✓				5
Nota:	33	¿El lenguaje de programación es claro, sencillo y es unificado en sus conceptos?	5	✓				5
Nota:	34	¿El lenguaje suministra estructuras de datos operacionales, estructuras de control y una sintaxis natural apropiada, para el problema que se va a resolver?	5	✓				5
Nota:	35	¿El lenguaje de programación facilita el desarrollo limpio, conciso, y libre de errores de los programas?	5	✓				4
Nota:	36	¿Los programas escritos en ese lenguaje, les permite ser fáciles de escribir?	5	✓				5
Nota:	37	¿El lenguaje facilita el desarrollo de programas libres de errores, como son los depuradores y mensajes de compilación?	5	✓				5



Nota:	38	¿El código fuente por sí mismo puede documentar el programa, ayudado de comentarios en los casos complicados?	5	<input checked="" type="checkbox"/>				5	Aun que no en su totalidad, solo en lugares necesarios
Nota:	39	¿El código fuente se puede extender en el programa para agregar nuevos elementos al lenguaje?	5	<input checked="" type="checkbox"/>				4	
Nota:	40	¿El código fuente es independiente de la arquitectura?	5	<input checked="" type="checkbox"/>				5	
Nota:	41	¿El uso de los recursos de cómputo y humanos es el óptimo?	5	<input checked="" type="checkbox"/>				4	
Nota:	42	¿La eficiencia del algoritmo es la óptima?	5	<input checked="" type="checkbox"/>				4	
Nota:	43	¿El uso de los recursos de la máquina, es el menor posible, sin limitar el rendimiento de la aplicación?	5	<input checked="" type="checkbox"/>				4	
Nota:	44	¿El personal autorizado puede hacer uso del código fuente?	5	<input checked="" type="checkbox"/>				5	Estrictamente personal autorizado.
Nota:	45	¿Cada módulo programado contiene lo siguiente en forma clara y concisa?							
Nota:	45.1	Nombre del módulo	5	<input checked="" type="checkbox"/>				5	
Nota:	45.2	Autor	5	<input checked="" type="checkbox"/>				5	
Nota:	45.3	Fecha de creación	5	<input checked="" type="checkbox"/>				5	
Nota:	45.4	Fecha de última modificación	5	<input checked="" type="checkbox"/>				5	
Nota:	45.5	Proyecto del que forma parte	5	<input checked="" type="checkbox"/>				5	
Nota:	45.6	Observaciones	5	<input checked="" type="checkbox"/>				5	
Nota:	45.7	En las consultas (queries) se pone una descripción breve	5	<input checked="" type="checkbox"/>				4	
Nota:	45.8	En las funciones se pone:							
Nota:	45.8.1	Descripción breve	5	<input checked="" type="checkbox"/>				4	
Nota:									

45.8.2	Declaración	5	<input checked="" type="checkbox"/>			4
Nota:						
46	Parámetros	5	<input checked="" type="checkbox"/>			4
Nota:						
47	¿Los nombres son suficientemente descriptivos, sin llegar a ser demasiado largos?	5	<input checked="" type="checkbox"/>			5
Nota:						
48	¿Los nombres proporcionan información sobre el tipo de entidad, programa y contexto en el que se encuentran?	5	<input checked="" type="checkbox"/>			5
Nota:						
49	¿Los nombres se refieren al qué y no al cómo?	5	<input checked="" type="checkbox"/>			5
Nota:						
50	¿La nomenclatura es expresiva y sencilla para facilitar la lectura y mantenimiento de los programas?	5	<input checked="" type="checkbox"/>			5
Nota:						
51	¿La nomenclatura es consistente y homogénea a través de los programas?	5	<input checked="" type="checkbox"/>			5
Nota:						
52	¿La nomenclatura es cómoda, congruente y simétrica?	5	<input checked="" type="checkbox"/>			5
Nota:						
53	¿Se planean instalaciones de prueba de módulos o de sistemas?	5	<input checked="" type="checkbox"/>			4
Nota:						
54	¿Se someten a pruebas rigurosas los módulos y el sistema ya estructurado?	5	<input checked="" type="checkbox"/>			4
Nota:						
55	¿Las pruebas realizadas corresponden a problemas que se tienen que enfrentar en la vida real?	5	<input checked="" type="checkbox"/>			4
Nota:						
56	¿Existe alguien responsable de realizar estas pruebas?	5	<input checked="" type="checkbox"/>			5
Nota:						
57	¿Se realiza la documentación de estas pruebas?	5	<input checked="" type="checkbox"/>			4
Nota:						
58	¿Existe alguien que evalúe el resultado de estas pruebas?	5	<input checked="" type="checkbox"/>			5
Nota:						
59	¿Se han preparado procedimientos de prueba y fallo para detectar errores?	5	<input checked="" type="checkbox"/>			3
Nota:						

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:		Estimado		Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis Calidad		Elaboro:		01/08/2004		10/09/2004	
Etapa:		Pruebas y Mantenimiento.		Supervisor:		Irma R. Carrión Bravo			
						Ing. Josefina Bárcenas López			
						Ing. José A. Domínguez			
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios	
	1	¿Cuentan con manuales de operación, usuario y sistema?	5		✓		0		
Nota:	2	¿Cuentan con el personal capacitado para el desarrollo adecuado de los manuales de operación, usuario y sistema?	5	✓			5		
Nota:	3	¿Tienen contemplado el manejo de errores dentro del sistema?	5	✓			5		
Nota:	4	¿Existen procedimientos para identificar errores o condiciones defectuosas?	5	✓			5	Las pruebas	
Nota:	5	¿Existe concordancia de la representación grafica con los módulos ya implementados?	5	✓			5		
Nota:	6	¿Existe un área que ofrezca atención a usuarios?	5	✓			5	La Ing. Josefina Bárcenas	
Nota:	7	¿Se capacita a los usuarios del sistema?	5	✓			5		
Nota:	8	¿Hay un control de calidad establecido en esta etapa?	5	✓			5		
Nota:									

## ANEXO 4: CUESTIONARIO PARA EL ANÁLISIS DE CALIDAD EXTERNO

### Instrucciones

- Contesta el siguiente cuestionario con numeración del número **1** al **5**. Donde el número **1** es el valor más bajo y el **5** el más alto.
- En las preguntas que corresponda o sea necesario contesta en el campo de comentario con **SI** o **No**.
- En las preguntas que no puedan ser contestadas rellena el campo de **N/A** (no aplica) y en el campo de **Comentario** pon la razón por la que esa pregunta no aplica.
- Los campos sombreados, como son "**Referencia**", "**Ponderación**" y "**Nota**" está reservados de toda modificación. Esto es que no se tiene que anotar nada en él.
- **De antemano gracias.**

Lineamientos	Numeración
Principios de Calidad:	1-16
Usabilidad	1-4
Funcionalidad	5-9
Fiabilidad	10-13
Eficiencia	14-16

Lineamientos	Numeración
Diseño de navegación	17-22
Diseño de pantallas	23-32
De interfaz	33-40
De Interfaz Web	41-45
De utilización de color	46-49

# USUARIO 1

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:		Estimado		Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente		Elaboro:		20/10/2004		10/01/05	
Etapas:		Calidad		Supervisor:		Ing. Josefina Bárcenas López		Ing. José A. Domínguez	
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios	
	1	¿Comprende el sitio globalmente?	5	✓			5		
Nota:	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5		✓		3		
Nota:	3	Califique el logro visual	5	✓			5		
Nota:	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5	✓			2	Pero no funciona	
Nota:	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	✓			5		
Nota:	6	¿El sistema tiene capacidad de recuperación? (regresar, deshacer)	5	✓			5		
Nota:	7	¿Se cuenta con un sistema de búsqueda? (externas y/o internas)	5	✓			4		
Nota:	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	✓			4		
Nota:	9	¿El sistema proporciona un servicio bueno?	5		✓		2		
Nota:	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5	✓			4		
Nota:	11	¿Hay enlaces rotos o que no funcionen?	5	✓			2		
Nota:									

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	14	¿El tiempo de respuesta es razonable?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	15	¿El tiempo en que se despliegan las páginas es mucho?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	16	¿El sistema tarda en cargar imágenes o gráficos?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	17	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	18	¿Los enlaces son uniformes?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	19	¿La forma en que se representan los enlaces es uniforme?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	<input checked="" type="checkbox"/>					3
<b>Nota:</b>									
	23	¿Se mantiene una pantalla sencilla?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	24	¿Se maneja una presentación consistente?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	25	¿Los movimientos entre pantallas son fáciles?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	<input checked="" type="checkbox"/>					4



Nota:	43	¿El sistema tiene enlaces rotos o inservibles?	5	<input checked="" type="checkbox"/>				1
Nota:	44	¿Se recorre la pantalla repetida o innecesariamente?	5	<input checked="" type="checkbox"/>				5
Nota:	45	¿Las opciones de navegación son intuitivas?	5	<input checked="" type="checkbox"/>				4
Nota:	46	¿El número de colores utilizados es excesivo?	5	<input checked="" type="checkbox"/>				5
Nota:	47	¿Se utilizan cambios de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>				5
Nota:	48	¿Se usan los mismos colores y diseño general en las pantallas?	5	<input checked="" type="checkbox"/>				5
Nota:	49	¿Los juegos de colores son contrastantes?	5	<input checked="" type="checkbox"/>				5



## USUARIO 2

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:	Estimado	Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Elaboro:	20/10/2004	10/01/05	
Etapas:		Calidad			Supervisor:	Irma R. Carrión Bravo		
					Ing. Josefina Bárcenas López			
					Ing. José A. Domínguez			
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios
	1	¿Comprende el sitio globalmente?	5	<input checked="" type="checkbox"/>			4	
Nota:	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5	<input checked="" type="checkbox"/>			1	
Nota:	3	Califique el logro visual	5	<input checked="" type="checkbox"/>			4	
Nota:	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5	<input checked="" type="checkbox"/>			4	
Nota:	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	<input checked="" type="checkbox"/>			4	
Nota:	6	¿El sistema tiene capacidad de recuperación? (regresar, deshacer)	5	<input checked="" type="checkbox"/>			2	
Nota:	7	¿Se cuenta con un sistema de búsqueda?(externas y/o internas)	5	<input checked="" type="checkbox"/>			4	
Nota:	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	<input checked="" type="checkbox"/>			4	
Nota:	9	¿El sistema proporciona un servicio bueno?	5	<input checked="" type="checkbox"/>			5	
Nota:	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5	<input checked="" type="checkbox"/>			5	
Nota:	11	¿Hay enlaces rotos o que no funcionan?	5	<input checked="" type="checkbox"/>			2	
Nota:								

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	14	¿El tiempo de respuesta es razonable?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	15	¿El tiempo en que se despliegan las páginas es mucho?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>							
	16	¿El sistema tarda en cargar imágenes o gráficos?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>							
	17	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	18	¿Los enlaces son uniformes?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	19	¿La forma en que se representan los enlaces es uniforme?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5	<input checked="" type="checkbox"/>			3
<b>Nota:</b>							
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	<input checked="" type="checkbox"/>			3
<b>Nota:</b>							
	23	¿Se mantiene una pantalla sencilla?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>							
	24	¿Se maneja una presentación consistente?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>							
	25	¿Los movimientos entre pantallas son fáciles?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>							
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	<input checked="" type="checkbox"/>			2

Nota:	27	¿El sistema indica si se aceptó la entrada?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	28	¿Se puede percibir si la entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	29	¿Se logra percibir si la entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	30	¿Se indica si habrá un retraso en el procesamiento?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	31	¿Se muestra si la solicitud ha sido concluida?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	32	¿Se demuestra si el sistema es incapaz de conducir la petición?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	33	¿Encuentra familiaridad con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	34	¿Existe uniformidad en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	35	¿Hay cambios radicales en los componentes de la pantalla?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	36	¿El comportamiento del sistema es predecible?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	37	¿Hay confirmación de acciones destructivas?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	38	¿El sistema dispone un recurso de deshacer?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
Nota:	39	¿Se proporcionan ayudas?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
Nota:	40	¿El sistema es fácil de usar?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	41	¿Se cuenta con un mapa de sitio?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	42	¿En el sistema se presentan símbolos de "Página en construcción"?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5

<b>Nota:</b>	43	¿El sistema tiene enlaces rotos o inservibles?	5	<input checked="" type="checkbox"/>				2
<b>Nota:</b>	44	¿Se recorre la pantalla repetida o innecesariamente?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>	45	¿Las opciones de navegación son intuitivas?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>	46	¿El número de colores utilizados es excesivo?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>	47	¿Se utilizan cambios de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>				2
<b>Nota:</b>	48	¿Se usan los mismos colores y diseño general en las pantallas?	5	<input checked="" type="checkbox"/>				3
<b>Nota:</b>	49	¿Los juegos de colores son contrastantes?	5	<input checked="" type="checkbox"/>				4

### USUARIO 3

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:	20/10/2004	Estimado	Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente		Elaboro:	Irma R. Carrón Bravo		10/01/05	
Etapa:		Calidad		Supervisor:	Ing. Josefina Bárcenas López Ing. José A. Domínguez			
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios
	1	¿Comprende el sitio globalmente?	5	✓			4	
Nota:	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5	✓			5	
Nota:	3	Califique el logro visual	5		✓		5	
Nota:	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5	✓			5	
Nota:	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	✓			5	
Nota:	6	¿El sistema tiene capacidad de recuperación? (regresar, deshacer)	5	✓			5	
Nota:	7	¿Se cuenta con un sistema de búsqueda (externas y/o internas)	5	✓			4	
Nota:	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	✓			5	
Nota:	9	¿El sistema proporciona un servicio bueno?	5	✓			5	
Nota:	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5	✓			5	
Nota:	11	¿Hay enlaces rotos o que no funcionan?	5		✓		5	
Nota:								

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	14	¿El tiempo de respuesta es razonable?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	15	¿El tiempo en que se despliegan las páginas es mucho?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	16	¿El sistema tarda en cargar imágenes o gráficos?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	17	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	18	¿Los enlaces son uniformes?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	19	¿La forma en que se representan los enlaces es uniforme?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	23	¿Se mantiene una pantalla sencilla?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	24	¿Se maneja una presentación consistente?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	25	¿Los movimientos entre pantallas son fáciles?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	<input checked="" type="checkbox"/>				4

Nota:	27	¿El sistema indica si se aceptó la entrada?	5	<input checked="" type="checkbox"/>				5
Nota:	28	¿Se puede percibir si la entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>				4
Nota:	29	¿Se logra percibir si la entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>				4
Nota:	30	¿Se indica si habrá un retraso en el procesamiento?	5	<input checked="" type="checkbox"/>				1
Nota:	31	¿Se muestra si la solicitud ha sido concluida?	5	<input checked="" type="checkbox"/>				5
Nota:	32	¿Se demuestra si el sistema es incapaz de conducir la petición?	5	<input checked="" type="checkbox"/>				5
Nota:	33	¿Encuentra familiaridad con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>				5
Nota:	34	¿Existe uniformidad en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>				5
Nota:	35	¿Hay cambios radicales en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>				5
Nota:	36	¿El comportamiento del sistema es predecible?	5	<input checked="" type="checkbox"/>				5
Nota:	37	¿Hay confirmación de acciones destructivas?	5	<input checked="" type="checkbox"/>				5
Nota:	38	¿El sistema dispone un recurso de deshacer?	5	<input checked="" type="checkbox"/>				2
Nota:	39	¿Se proporcionan ayudas?	5	<input checked="" type="checkbox"/>				5
Nota:	40	¿El sistema es fácil de usar?	5	<input checked="" type="checkbox"/>				5
Nota:	41	¿Se cuenta con un mapa de sitio?	5	<input checked="" type="checkbox"/>				5
Nota:	42	¿En el sistema se presentan símbolos de "página en construcción"?	5	<input checked="" type="checkbox"/>				3

Nota:	43	¿El sistema tiene enlaces rotos o inservibles?	5	<input checked="" type="checkbox"/>		5
Nota:	44	¿Se recorre la pantalla repetida o innecesariamente?	5	<input checked="" type="checkbox"/>		5
Nota:	45	¿Las opciones de navegación son intuitivas?	5	<input checked="" type="checkbox"/>		4
Nota:	46	¿El número de colores utilizados es excesivo?	5	<input checked="" type="checkbox"/>		5
Nota:	47	¿Se utilizan cambios de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>		5
Nota:	48	¿Se usan los mismos colores y diseño general en las pantallas?	5	<input checked="" type="checkbox"/>		5
Nota:	49	¿Los juegos de colores son contrastantes?	5	<input checked="" type="checkbox"/>		5
Nota:						



### USUARIO 4

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:	Estimado	Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Elaboro:	20/10/2004	10/01/05	
Etapas:		Calidad			Supervisor:	Irma R. Carrión Bravo		
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios
	1	¿Comprende el sitio globalmente?	5	✓			5	
Nota:	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5	✓			5	
Nota:	3	Califique el logro visual	5	✓			5	Bueno
Nota:	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5		✓		3	
Nota:	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	✓			4	
Nota:	6	¿El sistema tiene capacidad de recuperación? (regresar, deshacer)	5	✓			4	
Nota:	7	¿Se cuenta con un sistema de búsqueda?(externas y/o internas)	5		✓		3	
Nota:	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	✓			5	
Nota:	9	¿El sistema proporciona un servicio bueno?	5	✓			5	
Nota:	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5		✓		3	
Nota:	11	¿Hay enlaces rotos o que no funcionan?	5		✓		5	
Nota:								

	12	¿Se dispone de alguna forma de recuperarse de errores?	5				✓			No percibí nada de este punto
<b>Nota:</b>										
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5				✓		3	
<b>Nota:</b>										
	14	¿El tiempo de respuesta es razonable?	5				✓		4	
<b>Nota:</b>										
	15	¿El tiempo en que se despliegan las páginas es mucho?	5				✓		5	
<b>Nota:</b>										
	16	¿El sistema tarda en cargar imágenes o gráficos?	5					✓		No realicé este evento
<b>Nota:</b>										
	17	¿Se cuenta con controles de acceso?	5				✓		5	
<b>Nota:</b>										
	18	¿Los enlaces son uniformes?	5				✓		5	
<b>Nota:</b>										
	19	¿La forma en que se representan los enlaces es uniforme?	5				✓		5	
<b>Nota:</b>										
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5				✓		3	Solo hay colores: azul, blanco y negro.
<b>Nota:</b>										
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5				✓		5	
<b>Nota:</b>										
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5				✓		5	
<b>Nota:</b>										
	23	¿Se mantiene una pantalla sencilla?	5				✓		5	
<b>Nota:</b>										
	24	¿Se maneja una presentación consistente?	5				✓		4	
<b>Nota:</b>										
	25	¿Los movimientos entre pantallas son fáciles?	5				✓		5	
<b>Nota:</b>										
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5				✓		3	Se hace monótono.

Nota:	27	¿El sistema indica si se aceptó la entrada?	5	<input checked="" type="checkbox"/>			4
Nota:	28	¿Se puede percibir si la entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>			5
Nota:	29	¿Se logra percibir si la entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>			4
Nota:	30	¿Se indica si habrá un retraso en el procesamiento?	5	<input checked="" type="checkbox"/>			3
Nota:	31	¿Se muestra si la solicitud ha sido concluida?	5	<input checked="" type="checkbox"/>			5
Nota:	32	¿Se demuestra si el sistema es incapaz de concluir la petición?	5			<input checked="" type="checkbox"/>	
Nota:	33	¿Encuentra familiaridad con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>			5
Nota:	34	¿Existe uniformidad en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>			4
Nota:	35	¿Hay cambios radicales en los componentes de la pantalla?	5			<input checked="" type="checkbox"/>	4
Nota:	36	¿El comportamiento del sistema es predecible?	5	<input checked="" type="checkbox"/>			4
Nota:	37	¿Hay confirmación de acciones destructivas?	5			<input checked="" type="checkbox"/>	3
Nota:	38	¿El sistema dispone un recurso de deshacer?	5			<input checked="" type="checkbox"/>	4
Nota:	39	¿Se proporcionan ayudas?	5	<input checked="" type="checkbox"/>			5
Nota:	40	¿El sistema es fácil de usar?	5	<input checked="" type="checkbox"/>			5
Nota:	41	¿Se cuenta con un mapa de sitio?	5			<input checked="" type="checkbox"/>	3
Nota:	42	¿En el sistema se presentan símbolos de "Página en construcción"?	5			<input checked="" type="checkbox"/>	5



**USUARIO 5**

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:		Estimado		Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Elaboro:		20/10/2004		10/01/05	
Etapas:		Calidad			Supervisor:		Ing. Josefina Bárcenas López		Ing. José A. Domínguez	
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios		
	1	¿Comprende el sitio globalmente?	5	✓			5			
Nota:	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5	✓			4			
Nota:	3	Califique el logro visual	5	✓			4			
Nota:	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5	✓			3			
Nota:	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	✓			5			
Nota:	6	¿El sistema tiene capacidad de recuperación? (resgatar, deshacer)	5	✓			5			
Nota:	7	¿Se cuenta con un sistema de búsqueda?(externas y/o internas)	5	✓			4			
Nota:	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	✓			5			
Nota:	9	¿El sistema proporciona un servicio bueno?	5	✓			4			
Nota:	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5	✓			3			
Nota:	11	¿Hay enlaces rotos o que no funcionan?	5	✓			3			
Nota:										

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	✓			4
<b>Nota:</b>							
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5		✓		0
<b>Nota:</b>							
	14	¿El tiempo de respuesta es razonable?	5	✓			5
<b>Nota:</b>							
	15	¿El tiempo en que se despliegan las páginas es mucho?	5		✓		5
<b>Nota:</b>							
	16	¿El sistema tarda en cargar imágenes o gráficos?	5		✓		5
<b>Nota:</b>							
	17	¿Se cuenta con controles de acceso?	5	✓			5
<b>Nota:</b>							
	18	¿Los enlaces son uniformes?	5	✓			5
<b>Nota:</b>							
	19	¿La forma en que se representan los enlaces es uniforme?	5	✓			5
<b>Nota:</b>							
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	✓			3
<b>Nota:</b>							
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5		✓		3
<b>Nota:</b>							
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	✓			3
<b>Nota:</b>							
	23	¿Se mantiene una pantalla sencilla?	5	✓			5
<b>Nota:</b>							
	24	¿Se maneja una presentación consistente?	5	✓			4
<b>Nota:</b>							
	25	¿Los movimientos entre pantallas son fáciles?	5	✓			5
<b>Nota:</b>							
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	✓			3

Nota:	27	¿El sistema indica si se aceptó la entrada?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	28	¿Se puede percibir si la entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	29	¿Se logra percibir si la entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	30	¿Se indica si habrá un retraso en el procesamiento?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	31	¿Se muestra si la solicitud ha sido concluida?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	32	¿Se demuestra si el sistema es incapaz de concluir la petición?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	33	¿Encuentra familiaridad con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	34	¿Existe uniformidad en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	35	¿Hay cambios radicales en los componentes de la pantalla?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	36	¿El comportamiento del sistema es predecible?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	37	¿Hay confirmación de acciones destructivas?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	38	¿El sistema dispone un recurso de deshacer?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	39	¿Se proporcionan ayudas?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	40	¿El sistema es fácil de usar?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	41	¿Se cuenta con un mapa de sitio?	5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
Nota:	42	¿En el sistema se presentan símbolos de "Página en construcción"?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5





**USUARIO 6**

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina		Fecha:	Estimado	Real		
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente		Elaboro:	20/10/2004	10/01/05		
Etapas:		Calidad		Supervisor:	Irma R. Carrón Bravo			
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios
	1	¿Comprende el sitio globalmente?	5	✓			4	
<b>Nota:</b>	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5	✓			4	
<b>Nota:</b>	3	Califique el logro visual	5	✓			4	
<b>Nota:</b>	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5	✓			3	
<b>Nota:</b>	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	✓			4	
<b>Nota:</b>	6	¿El sistema tiene capacidad de recuperación? (regresar, desahcer)	5	✓			4	
<b>Nota:</b>	7	¿Se cuenta con un sistema de búsqueda?(externas y/o internas)	5	✓			4	
<b>Nota:</b>	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	✓			5	
<b>Nota:</b>	9	¿El sistema proporciona un servicio bueno?	5	✓			4	
<b>Nota:</b>	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5		✓		2	En participación y mapa
<b>Nota:</b>	11	¿Hay enlaces rotos o que no funcionan?	5		✓		5	
<b>Nota:</b>								

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5	<input checked="" type="checkbox"/>					3
<b>Nota:</b>									
	14	¿El tiempo de respuesta es razonable?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	15	¿El tiempo en que se despliegan las páginas es mucho?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	16	¿El sistema tarda en cargar imágenes o gráficos?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	17	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	18	¿Los enlaces son uniformes?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	19	¿La forma en que se representan los enlaces es uniforme?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	23	¿Se mantiene una pantalla sencilla?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	24	¿Se maneja una presentación consistente?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	25	¿Los movimientos entre pantallas son fáciles?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	<input checked="" type="checkbox"/>					4



<b>Nota:</b>									
	43	¿El sistema tiene enlaces rotos o inservibles?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	44	¿Se recorre la pantalla repetida o innecesariamente?	5	<input checked="" type="checkbox"/>					3
<b>Nota:</b>									
	45	¿Las opciones de navegación son intuitivas?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	46	¿El número de colores utilizados es excesivo?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	47	¿Se utilizan cambios de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									
	48	¿Se usan los mismos colores y diseño general en las pantallas?	5	<input checked="" type="checkbox"/>					5
<b>Nota:</b>									
	49	¿Los juegos de colores son contrastantes?	5	<input checked="" type="checkbox"/>					4
<b>Nota:</b>									

# USUARIO 7

Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:		Estimado		Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Elaboro:		20/10/2004		10/01/05	
Etapas:		Calidad			Supervisor:		Ing. Josefina Bárcenas López		Ing. José A. Dominguez	
Referencia	No de Pregunta	Pregunta	Ponderación	Si	No	N/A	Calif.	Comentarios		
	1	¿Comprende el sitio globalmente?	5	<input checked="" type="checkbox"/>			5			
<b>Nota:</b>	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5		<input checked="" type="checkbox"/>		3			
<b>Nota:</b>	3	Califique el logro visual	5	<input checked="" type="checkbox"/>			3			
<b>Nota:</b>	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5		<input checked="" type="checkbox"/>		3			
<b>Nota:</b>	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5	<input checked="" type="checkbox"/>			5			
<b>Nota:</b>	6	¿El sistema tiene capacidad de recuperación? (regresar, deshacer)	5	<input checked="" type="checkbox"/>			4			
<b>Nota:</b>	7	¿Se cuenta con un sistema de búsqueda?(externas y/o internas)	5	<input checked="" type="checkbox"/>			4			
<b>Nota:</b>	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5	<input checked="" type="checkbox"/>			4			
<b>Nota:</b>	9	¿El sistema proporciona un servicio bueno?	5		<input checked="" type="checkbox"/>		3			
<b>Nota:</b>	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5		<input checked="" type="checkbox"/>		3			
<b>Nota:</b>	11	¿Hay enlaces rotos o que no funcionen?	5	<input checked="" type="checkbox"/>			3			
<b>Nota:</b>										

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
<b>Nota:</b>									
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
<b>Nota:</b>									
	14	¿El tiempo de respuesta es razonable?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
<b>Nota:</b>									
	15	¿El tiempo en que se despliegan las páginas es mucho?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
<b>Nota:</b>									
	16	¿El sistema tarda en cargar imágenes o gráficos?	5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	A las páginas que entré no tienen imágenes
<b>Nota:</b>									
	17	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
<b>Nota:</b>									
	18	¿Los enlaces son uniformes?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
<b>Nota:</b>									
	19	¿La forma en que se representan los enlaces es uniforme?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
<b>Nota:</b>									
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
<b>Nota:</b>									
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
<b>Nota:</b>									
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
<b>Nota:</b>									
	23	¿Se mantiene una pantalla sencilla?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
<b>Nota:</b>									
	24	¿Se maneja una presentación consistente?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
<b>Nota:</b>									
	25	¿Los movimientos entre pantallas son fáciles?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
<b>Nota:</b>									

	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	<input checked="" type="checkbox"/>		3	Colores neutros, le falta combinación y letra de mayor tamaño
<b>Nota:</b>	27	¿El sistema indica si se aceptó la entrada?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	28	¿Se puede percibir si la entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>		4	
<b>Nota:</b>	29	¿Se logra percibir si la entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	30	¿Se indica si habrá un retraso en el procesamiento?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	31	¿Se muestra si la solicitud ha sido concluida?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	32	¿Se demuestra si el sistema es incapaz de concluir la petición?	5	<input checked="" type="checkbox"/>		4	
<b>Nota:</b>	33	¿Encuentra familiaridad con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	34	¿Existe uniformidad en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>		4	Hay errores de texto, palabras mal escritas.
<b>Nota:</b>	35	¿Hay cambios radicales en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>		5	
<b>Nota:</b>	36	¿El comportamiento del sistema es predecible?	5	<input checked="" type="checkbox"/>		4	
<b>Nota:</b>	37	¿Hay confirmación de acciones destructivas?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	38	¿El sistema dispone un recurso de deshacer?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	39	¿Se proporcionan ayudas?	5	<input checked="" type="checkbox"/>		3	
<b>Nota:</b>	40	¿El sistema es fácil de usar?	5	<input checked="" type="checkbox"/>		4	
<b>Nota:</b>	41	¿Se cuenta con un mapa de sitio?	5	<input checked="" type="checkbox"/>		3	

<b>Nota:</b>									
42	¿En el sistema se presentan símbolos de "Página en construcción"?	5							5
<b>Nota:</b>									
43	¿El sistema tiene enlaces rotos o inservibles?	5							3
<b>Nota:</b>									
44	¿Se recorre la pantalla repetida o innecesariamente?	5							4
<b>Nota:</b>									
45	¿Las opciones de navegación son intuitivas?	5							5
<b>Nota:</b>									
46	¿El número de colores utilizados es excesivo?	5							5
<b>Nota:</b>									
47	¿Se utilizan cambios de color para mostrar el cambio en el estado del sistema?	5							3
<b>Nota:</b>									
48	¿Se usan los mismos colores y diseño general en las pantallas?	5							4
<b>Nota:</b>									
49	¿Los juegos de colores son contrastantes?	5							4
<b>Nota:</b>									



**USUARIO 8**

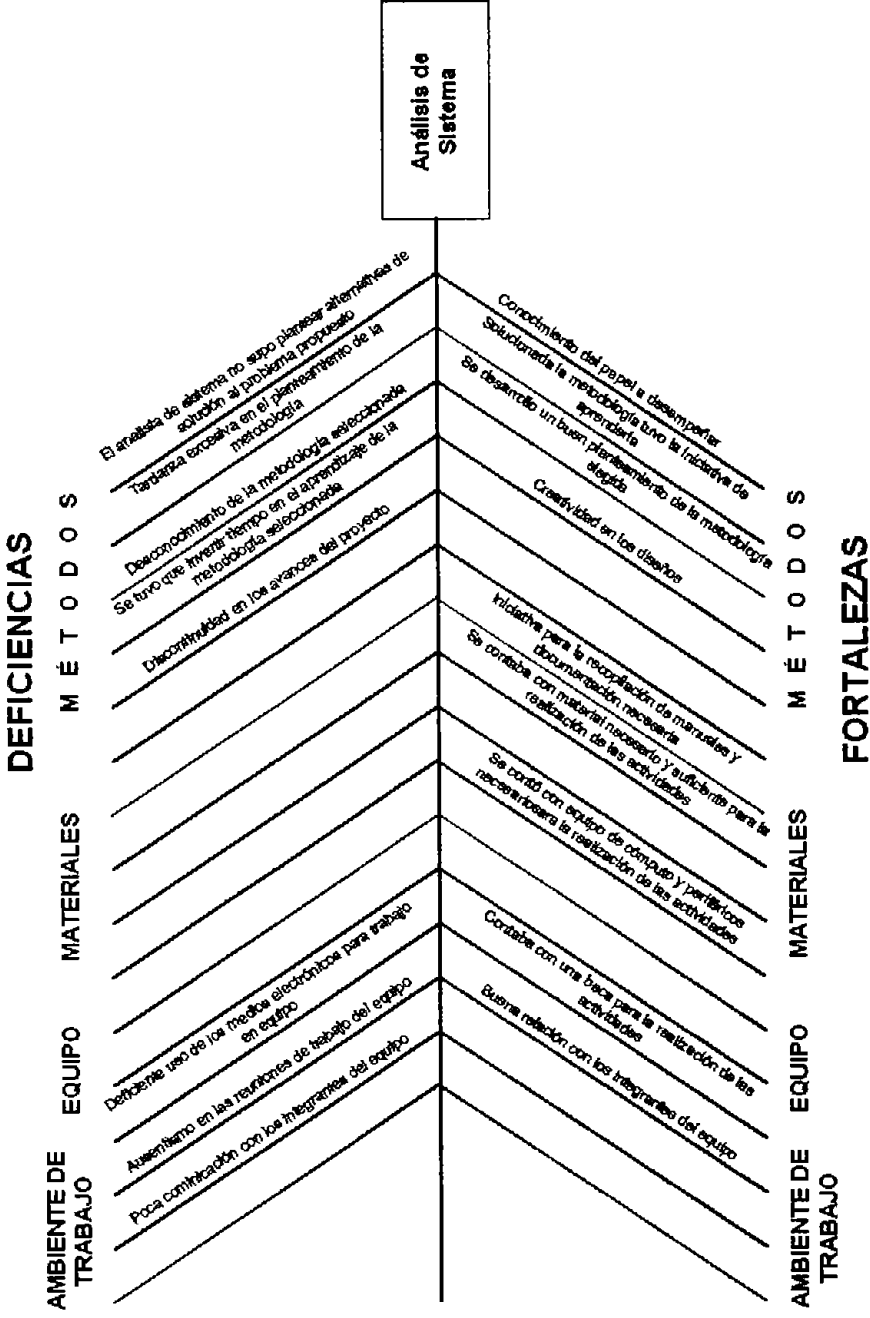
Nombre de la Empresa:		Laboratorio de Interacción Humano Máquina			Fecha:	Estimado	Real	
Nombre del Cuestionario:		Cuestionario Para el Análisis de la Voz del Cliente			Elaboro:	20/10/2004	10/01/05	
Etapas:		Calidad			Supervisor:	Irma R. Carrión Bravo		
Referencia	No de Pregunta	Pregunta	Ponderación	SI	No	N/A	Calif.	Comentarios
	1	¿Comprende el sitio globalmente?	5		✓		3	
Nota:								
	2	¿Hay algún servicio de ayuda y/o retroalimentación en línea?	5		✓		3	
Nota:								
	3	Califique el logro visual	5		✓		5	
Nota:								
	4	¿Se proporciona algún servicio especial? (ayuda, búsqueda)	5		✓		4	
Nota:								
	5	¿El sistema realiza las tareas para las cuales fue desarrollada?	5		✓		4	
Nota:								
	6	¿El sistema tiene capacidad de recuperación? (regresar, deshacer)	5		✓		4	
Nota:								
	7	¿Se cuenta con un sistema de búsqueda?(externas y/o internas)	5		✓		4	
Nota:								
	8	¿Se tiene un sistema de navegación? (menús, enlaces)	5		✓		5	
Nota:								
	9	¿El sistema proporciona un servicio bueno?	5		✓		5	
Nota:								
	10	¿El sistema permite eliminar errores a través de verificación o algún sistema parecido? (Entradas de datos erróneas, o caracteres no válidos)	5		✓		3	
Nota:								
	11	¿Hay enlaces rotos o que no funcionan?	5		✓		5	
Nota:								

	12	¿Se dispone de alguna forma de recuperarse de errores?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	13	¿Se pueden recuperar las entradas ya registradas? (recordar login o password olvidados)	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	14	¿El tiempo de respuesta es razonable?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	15	¿El tiempo en que se despliegan las páginas es mucho?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	16	¿El sistema tarda en cargar imágenes o gráficos?	5	<input checked="" type="checkbox"/>				No Vi esto
<b>Nota:</b>								
	17	¿Se cuenta con controles de acceso?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	18	¿Los enlaces son uniformes?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	19	¿La forma en que se representan los enlaces es uniforme?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	20	¿Los enlaces tienen diferentes colores o dinámicas que reflejen sus diferentes estados?	5	<input checked="" type="checkbox"/>				4
<b>Nota:</b>								
	21	¿Los enlaces indican si se ha elegido una opción de navegación?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	22	¿Se proporciona una indicación de los enlaces por los que se ha navegado?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	23	¿Se mantiene una pantalla sencilla?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	24	¿Se maneja una presentación consistente?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	25	¿Los movimientos entre pantallas son fáciles?	5	<input checked="" type="checkbox"/>				5
<b>Nota:</b>								
	26	¿El tipo de letra, imágenes y el color hacen las pantallas atractivas?	5	<input checked="" type="checkbox"/>				4

Nota:	27	¿El sistema indica si se aceptó la entrada?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	28	¿Se puede percibir si la entrada se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	29	¿Se logra percibir si la entrada no se encuentra en forma correcta?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	30	¿Se indica si habrá un retraso en el procesamiento?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	31	¿Se muestra si la solicitud ha sido concluida?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	32	¿Se demuestra si el sistema es incapaz de concluir la petición?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	33	¿Encuentra familiaridad con otros sistemas ya utilizados?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	34	¿Existe uniformidad en los componentes de la pantalla?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	35	¿Hay cambios radicales en los componentes de la pantalla?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	36	¿El comportamiento del sistema es predecible?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	37	¿Hay confirmación de acciones destructivas?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	38	¿El sistema dispone un recurso de deshacer?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
Nota:	39	¿Se proporcionan ayudas?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	40	¿El sistema es fácil de usar?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	41	¿Se cuenta con un mapa de sitio?	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
Nota:	42	¿En el sistema se presentan símbolos de "Página en construcción"?	5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5

<b>Nota:</b>	43	¿El sistema tiene enlaces rotos o inservibles?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>	44	¿Se recorre la pantalla repetida o innecesariamente?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>	45	¿Las opciones de navegación son intuitivas?	5	<input checked="" type="checkbox"/>			4
<b>Nota:</b>	46	¿El número de colores utilizados es excesivo?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>	47	¿Se utilizan cambios de color para mostrar el cambio en el estado del sistema?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>	48	¿Se usan los mismos colores y diseño general en las pantallas?	5	<input checked="" type="checkbox"/>			5
<b>Nota:</b>	49	¿Los juegos de colores son contrastantes?	5	<input checked="" type="checkbox"/>			5

# ANEXO 5: DEFICIENCIAS Y FORTALEZAS PRODUCTIVAS TÉCNICA DE LAS 5 M'S



**DEFICIENCIAS**

**AMBIENTE DE TRABAJO**

**EQUIPO**

**MATERIALES**

**MÉTODOS**

Ausentismo en las reuniones de trabajo del equipo  
Poca comunicación con los integrantes del equipo

Avance lento durante el análisis comparativo entre AOO y Arriba Estructurado  
Dificultad al decidir el uso de una metodología

**Diseño de Sistema Programación e Implementación**

Buena relación con el líder de proyecto y los directores  
Estable uso de los medios electrónicos para trabajo en equipo

Herramientas de desarrollo de punta  
Manejo excelente en el manejo de las herramientas de desarrollo y diseño  
Buena elección en las herramientas de programación  
Después de seleccionar la metodología se pusieron  
El diseñador de sistema tiene capacidades adicionales a las de programación, mismas que fueron aprovechadas para completar el diseño de la interfaz  
El software necesario se obtuvo de Internet  
Los manuales para aprender las herramientas necesarias se obtuvieron de Internet  
Se contó con el equipo que sería para el portal  
Se contó con equipo de cómputo y perifericos necesarios para la realización de las actividades

**AMBIENTE DE TRABAJO**

**EQUIPO**

**MATERIALES**

**MÉTODOS**

**FORTALEZAS**

**DEFICIENCIAS**

**AMBIENTE DE TRABAJO**

**EQUIPO**

**MATERIALES**

**MÉTODOS**

- Falta de claridad en la presentación de la investigación realizada
- Por la falta de comprensión del tema no hubo avances respecto a ideas para el proyecto
- Discontinuidad en los avances para el proyecto
- Dependencia no justificada de los demás integrantes del proyecto
- Falta de claridad del papel a desempeñar por parte del responsable de seguridad
- Duplicidad de actividades para la recopilación de documentos de referencia
- Asistencia en las reuniones de trabajo del equipo
- Poca contribución con los integrantes del equipo
- Deficiente uso de las mejores alternativas para trabajo

**Calidad y Seguridad**

- Buena investigación bajo estricta en tiempo de principios de calidad
- Buena investigación de Benchmarking
- Buena aplicación de los principios de Benchmarking
- Herramientas utilizadas de punta
- Aplicación de calidad en tiempo
- Visión clara de la implementación de la calidad en el sistema
- Se contó con material necesario y suficiente para la realización de las actividades
- Se contó con equipo de cómputo y periféricos necesarios para la realización de las actividades
- Buena relación con el líder de proyecto y los directores de trabajo
- Contaba con una beca para la realización de las actividades

**AMBIENTE DE TRABAJO**

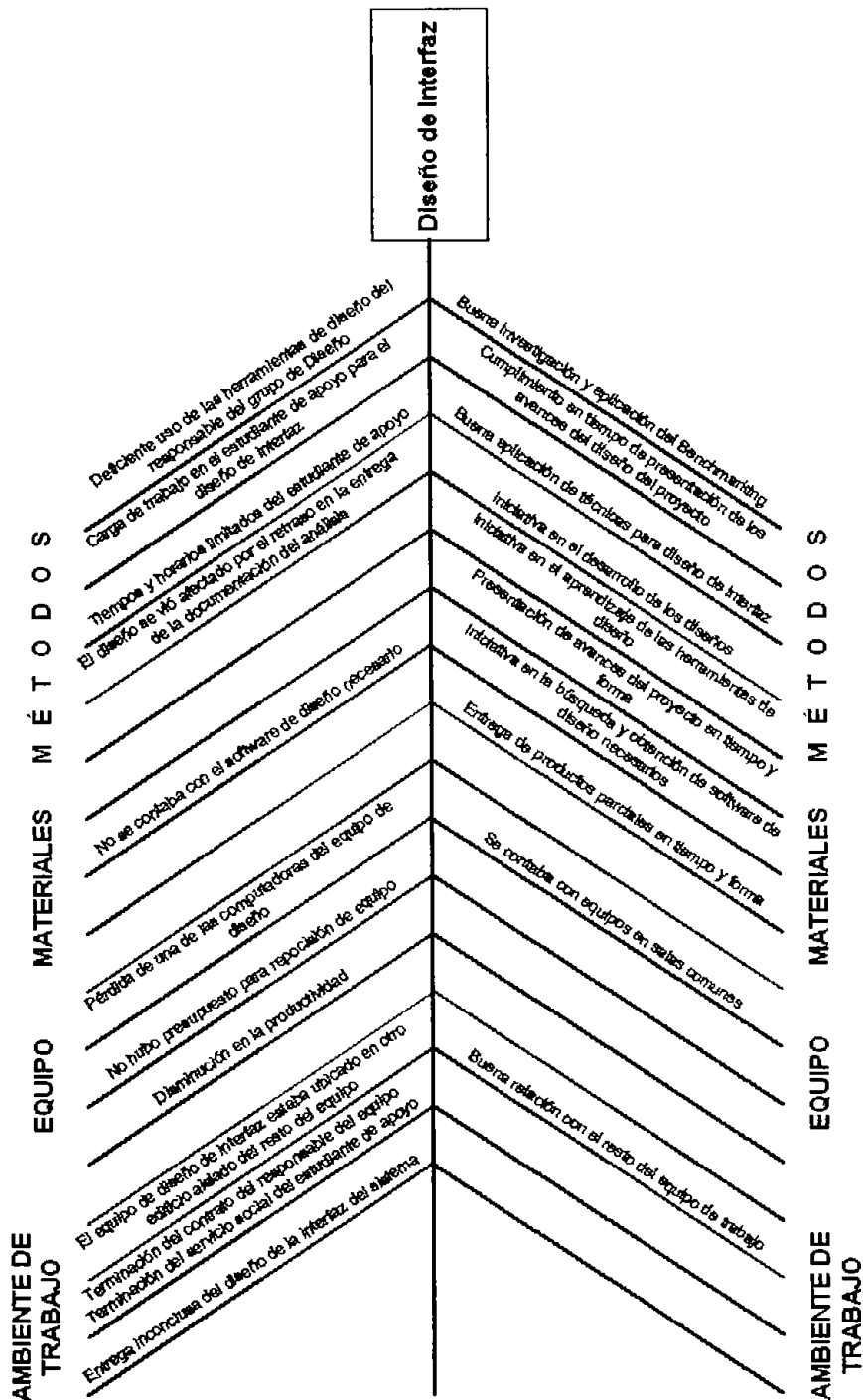
**EQUIPO**

**MATERIALES**

**MÉTODOS**

**FORTALEZAS**

# DEFICIENCIAS





# ANEXO 6: ANÁLISIS DE RIESGOS

**RECURSO CRÍTICO:**  
*Información*

- Actor Humano
- Acceso De Red

	Valoración Impacto										Probabilidad			En caso de logro			
	Reputac	Financie	Product	Usabilidad	Funcion	Fabrida	Eficient	Rapidez	Confabi	Valor	Alto	Medio	Nada	Aceptar	Detener	Mitigar	
Interno	Accidental	Pérdida/Interrupción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Modificación	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Intercepción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Robo	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Interno	Deliberado	Pérdida/Interrupción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Modificación	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Intercepción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Robo	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Externo	Accidental	Pérdida/Interrupción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Modificación	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Intercepción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Robo	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Externo	Deliberado	Pérdida/Interrupción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Modificación	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Intercepción	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		Robo	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

**RECURSO CRÍTICO:**  
**Información**

- Actor Humano
- Acceso Físico

	Valoración Impacto										Probabilidad				En caso de logro		
	Reputación	Financiero	Productividad	Usabilidad	Funcionalidad	Fabilidad	Endeuda	Rapidez	Confiablez	Valor	Mucho	Algo	Nada	Aceptar	Detener	Mitigar	
																	Confidencialidad
Accidental	Pérdida/Interrupción	*	*	*	*	*	*	*	*	*	♦				*		
	Modificación	*	*	*	*	*	*	*	*	*	♦				*		
	Intercepción	*	*	*	*	*	*	*	*	*	♦				*		
	Robo																
	Publicación																
Deliberado	Pérdida/Interrupción	*	*	*	*	*	*	*	*	*	♦	A			*		
	Modificación	*	*	*	*	*	*	*	*	*	♦	A			*		
	Intercepción	*	*	*	*	*	*	*	*	*	♦	A			*		
	Robo	*	*	*	*	*	*	*	*	*	♦	A			*		
	Publicación	*	*	*	*	*	*	*	*	*	♦	A			*		
Accidental	Pérdida/Interrupción																
	Modificación																
	Intercepción																
	Robo																
	Publicación																
Deliberado	Pérdida/Interrupción																
	Modificación																
	Intercepción																
	Robo																
	Publicación																

Interno

Externo

Acceso Físico  
Información

**RECURSO CRÍTICO:**  
**Información**

- Problemas de Sistema

	Valoración Impacto										Probabilidad			En caso de logro			
	Reputación	Financiero	Productividad	Usabilidad	Funcionalidad	Fidelidad	Eficiencia	Rapidez	Confiablez	Confabilidad	Valor	Confidencialidad			Aceptar	Detener	Mitigar
												Mucho	Algo	Nada			
Software Defectuoso	Pérdida/Interrupción	+		+	+	+	+	+	+	+		+					
	Modificación	+		+	+	+	+	+	+	+	A	+					
	Intercepción	+		+	+	+	+	+	+	+	A	+					
	Robo	+		+	+	+	+	+	+	+	A	+					
Código Malicioso	Pérdida/Interrupción	+		+	+	+	+	+	+	+		+					
	Modificación	+		+	+	+	+	+	+	+	A	+					
	Intercepción	+		+	+	+	+	+	+	+	A	+					
	Robo	+		+	+	+	+	+	+	+	A	+					
Caídas De Sistema	Pérdida/Interrupción	+		+	+	+	+	+	+	+		+					
	Modificación	+		+	+	+	+	+	+	+	A	+					
	Intercepción	+		+	+	+	+	+	+	+	A	+					
	Robo	+		+	+	+	+	+	+	+	A	+					
Inestabilidad En LAN	Pérdida/Interrupción	+		+	+	+	+	+	+	+		+					
	Modificación	+		+	+	+	+	+	+	+	A	+					
	Intercepción	+		+	+	+	+	+	+	+	A	+					
	Robo	+		+	+	+	+	+	+	+	A	+					

**RECURSO CRITICO:  
Información**

- Otros Problemas

	Valoración Impacto										Probabilidad			En caso de logro		
	Reputación	Financiero	Productividad	Usabilidad	Funcionalidad	Fabilidad	Eficiencia	Rapidez	Confidencial	Valor	Mucho	Algo	Nada	Aceptar	Detener	Mitigar
Desastres Naturales	Pérdida/Interrupción															
	Modificación															
	Intercepción															
	Robo															
Servicio Red Irregular O Indisponible	Publicación															
	Pérdida/Interrupción															
	Modificación															
	Intercepción															
Alimentación Eléctrica	Robo															
	Publicación															
	Pérdida/Interrupción															
	Modificación															





**RECURSO CRITICO:**  
**Sistema**

- Problemas de Sistema

Sistema	Valoración Impacto										Probabilidad			En caso de logro		
	Reputación	Financiero	Productividad	Usabilidad	Funcionalidad	Fiabilidad	Eficiencia	Rapidez	Confidencialidad	Valor	Mucho	Algo	Nada	Aceptar	Detener	Mitigar
Software Defectuoso	Reputación	*		+	+	+	+	+	+	+	+	+	+			
	Financiero			+	+	+	+	+	+	+	+	+	+			
	Productividad			+	+	+	+	+	+	+	+	+	+			
	Usabilidad			+	+	+	+	+	+	+	+	+	+			
Código Malicioso	Reputación	*		+	+	+	+	+	+	+	+	+	+			
	Financiero			+	+	+	+	+	+	+	+	+	+			
	Productividad			+	+	+	+	+	+	+	+	+	+			
	Usabilidad			+	+	+	+	+	+	+	+	+	+			
Caídas De Sistema	Reputación	*		+	+	+	+	+	+	+	+	+	+			
	Financiero			+	+	+	+	+	+	+	+	+	+			
	Productividad			+	+	+	+	+	+	+	+	+	+			
	Usabilidad			+	+	+	+	+	+	+	+	+	+			
Inestabilidad En LAN	Reputación	*		+	+	+	+	+	+	+	+	+	+			
	Financiero			+	+	+	+	+	+	+	+	+	+			
	Productividad			+	+	+	+	+	+	+	+	+	+			
	Usabilidad			+	+	+	+	+	+	+	+	+	+			

**RECURSO CRITICO:**  
**Sistema**

- Otros Problemas

	Valoración Impacto										Probabilidad			En caso de logro		
	Reputación	Financiero	Productividad	Usabilidad	Funcionalidad	Fabilidad	Eficiencia	Rapidez	Confabilidad	Confidencialidad			Aceptar	Detener	Mitigar	
										Valor	Mucho	Algo				Nada
Desastres Naturales	Pérdida/Interrupción	+	+	+	+	+	+	+	+		A	◆				*
	Modificación															
	Intercepción															
	Robo															
Servicio Red Irregular O Indisponible	Publicación															
	Pérdida/Interrupción	+	+	+	+	+	+	+	+		A	◆				*
	Modificación															
	Intercepción															
Alimentación Eléctrica	Robo															
	Publicación															
	Pérdida/Interrupción	+	+	+	+	+	+	+	+		A	◆				*
	Modificación															
Sistema	Intercepción															
	Robo															
	Publicación															
	Modificación															







**RECURSO CRITICO:**  
**Servicios y Aplicaciones**

- Problemas de Sistema

Publicación	Valoración Impacto										Probabilidad			En caso de logro		
	Reputación	Finandero	Productividad	Usabilidad	Funcionalidad	Habilidad	Eficiencia	Rapidez	Confiabledad	Valor	Confidencialidad			Aceptar	Detener	Mitigar
											Mucho	Algo	Nada			
Software Defectuoso	Pérdida/Interrupción	+		+	+	+	+	+	+	+	+	+	+			
	Modificación	+		+	+	+	+	+	+	A	+	+				
	Intercepción	+		+	+	+	+	+	+	A	+	+				
	Robo	+		+	+	+	+	+	+	A	+	+				
Código Malicioso	Pérdida/Interrupción	+		+	+	+	+	+	+	+	+	+	+			
	Modificación	+		+	+	+	+	+	+	A	+	+				
	Intercepción	+		+	+	+	+	+	+	A	+	+				
	Robo	+		+	+	+	+	+	+	A	+	+				
Caídas De Sistema	Pérdida/Interrupción	+		+	+	+	+	+	+	+	+	+	+			
	Modificación	+		+	+	+	+	+	+	A	+	+				
	Intercepción	+		+	+	+	+	+	+	A	+	+				
	Robo	+		+	+	+	+	+	+	A	+	+				
Inestabilidad Fx IAN	Pérdida/Interrupción	+		+	+	+	+	+	+	+	+	+	+			
	Modificación	+		+	+	+	+	+	+	A	+	+				
	Intercepción	+		+	+	+	+	+	+	A	+	+				
	Robo	+		+	+	+	+	+	+	A	+	+				

