UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

"IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO PARA SISTEMAS TIPO LINUX"

P R O Y E C T O

QUE PARA OBTENER EL TÍTULO DE:
LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T A:

VANEGAS LOMAS VERÓNICA

Director de Tesis: M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ

MÉXICO **2005**

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito: "Implementación de un sistema de monitoreo para sistemas tipo linux"

realizado por Verónica Vanegas Lomas

con número de cuenta 9420881-6 , quien cubrió los créditos de la carrera de: Lic. en Ciencias de la Computación.

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	M en C. María Guadalupe Elena Ibargüengoitia González	
Propietario	Dra. Hanna Oktaba.	
Propietario	M. en I.O. María de Luz Gasca Soto.	
Suplente	Mat. Salvador López Mendoza.	
Suplente	M. en C. Gustavo Adolfo Arellano Sandoval.	

Consejo Departamental de Matemáticas

Dr. Francisco Hernández Quiroz.

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL

DEDICATORIAS

Nunca confundas el conocimiento con la sabiduría. Uno te ayuda a ganarte la vida; el otro te ayuda a construirla.

A mis padres:

Por que sin el apoyo de los dos no hubiera llegado tan lejos. Gracias por la paciencia que me tuvieron y por el impulso que me dieron cuando me llego a faltar la fuerza para continuar.

A mis hermanos:

Gracias por el apoyo que me brindaron, por su confianza y por estar ahí cuando los he necesitado.

A mis sobrinos:

Por el cariño incondicional que me brindan.

A mis tíos:

Muchas gracias por poyarnos tanto a pesar de estar tan lejos.
A mis primos y amigos de la facultad, que me apoyaron en los mementos difíciles.

A Carlos:

Gracias por tu apoyo, paciencia y comprensión por estar ahí cuando te he necesitado. Por creer en mi, muchas gracias por todo lo que me has brindado.

AGRADECIMIENTOS

A Dios:

Por haberme permitido llegar hasta este momento.

A la UNAM:

Por el conocimiento proporcionado durante toda mi estancia en ella, que me ha abierto muchas puertas.

A la Facultad de Ciencias:

Por haberme brindado conocimiento que me ayudo a concluir con mis estudios.

A LA DGSCA:

En especial a Liz por haberme permitido estar en el plan de becarios brindándome la oportunidad de seguir ampliando mis conocimientos. Y a mis compañeros del plan de becarios.

A MI ASESORA:

Por haber dedicado tiempo y esfuerzo para realización de mi tesis.

A LA UPN:

Por apoyarme para terminar este trabajo en especial a marcy.
También quiero agradecer a mis amigos del trabajo Azael, Ricardo y Erick.

A SONIA:

Por su amistad incondicional, sus consejos y sobre todo por su apoyo, muchas gracias.

ÍNDICE

INTRODUCCIÓN

OBJETIVO GENERAL

Implementar un sistema de monitoreo, que facilite al administrador de sistemas la tarea de detectar vulnerabilidades en la seguridad de un sistema.

JUSTIFICACIÓN DEL TEMA

Dado el aumento de equipos de cómputo que utilizan sistemas Unix/Linux para proporcionar servicios, crece cada vez más el número de servidores que son sujetos de acceso no autorizado, por lo que se requiere de una herramienta para realizar un monitoreo de los elementos básicos del sistema y con ello poder detectar un posible ataque a cualquiera de estos elementos.

DESCRIPCIÓN DEL TRABAJO

En este proyecto se presenta un sistema de monitoreo basado en técnicas de ingeniería de software y con la filosofía del software libre, esperando que sea de utilidad para los administradores de sistemas, así como para cualquier persona que esté interesada en el desarrollo de aplicaciones con los lenguajes de programación perl y perl/tk.

En la primera parte de este trabajo se realizó un resumen sobre las fases del desarrollo de software del libro [Braude] "Ingeniería de software, una perspectiva orientada a objetos" esto con el fin de tener la información necesaria, para entender mejor la aplicación de un proceso completo que permita desarrollar software de calidad. En segunda parte se presenta el desarrollo de la aplicación, siguiendo las fases del desarrollo de software estudiado en los capítulos I, II, III, IV, V, VI, VII y VIII.

CAPÍTULO I

EL PROCESO DE DESARROLLO DE SOFTWARE

La ingeniería de software propone un proceso para construir aplicaciones de tamaño o alcance prácticos, en las que predomina el esfuerzo para el desarrollo del software y que satisfacen los requerimientos de funcionalidad y desempeño.

El alcance de las aplicaciones de software se ha expandido sin precedente desde la invención de las computadoras. La ingeniería de software debe apoyar en la construcción de todos estos tipos de aplicaciones.

La meta de todo proyecto de software es producir un producto de software de calidad. También existe preocupación por el proceso mediante el cual se generan productos de manera efectiva.

Existen cinco expectativas importantes de la ingeniería de software.

1. Predeterminar metas de calidad cuantitativas.
2. Mantener todo el trabajo visible.
3. Diseñar sólo lo establecido en los requerimientos, programar únicamente lo establecido en el diseño y probar solamente requerimientos y diseño.
4. Medir y lograr las metas de calidad.
5. Reunir datos para proyectos subsecuentes.

El producto de la ingeniería de software consiste en mucho más que un código. Incluye artefactos como planes, informes y gráficas.

Los ingenieros de software desempeñan una variedad de roles o papeles con responsabilidades asignadas tales como, implementación de código, documentación en cada uno de las fases, etcétera.

El proceso bajo el que se desarrollan los proyectos es un factor importante para administrar su complejidad. Existen varios procesos alternativos, entre los cuales él mas antiguo es el proceso en cascada.

1.1 Proceso en cascada

El modelo clásico del proceso de desarrollo de software es el modelo en cascada. Este es una secuencia única de actividades que consiste en el análisis de requerimientos, diseño, implementación, integración, pruebas y mantenimiento[1].

[1] Ingeniería de software , Una perspectiva orientada a objetos autor Fraude pag 24

- Análisis de requerimientos consiste en analizar las necesidades del producto recabadas previamente, su salida es texto.
- Diseño describe la estructura interna del producto, se representa con diagramas y texto.
- Implementación es la programación, el producto de esta etapa es el código en cualquier nivel.
- Integración es el proceso de ensamblar las partes para complementar el producto.
- Pruebas unitarias y del sistema, marca la distinción entre probar las partes de una aplicación y el todo.
- Mantenimiento es donde se repara y modifica la aplicación para que continúe siendo útil.

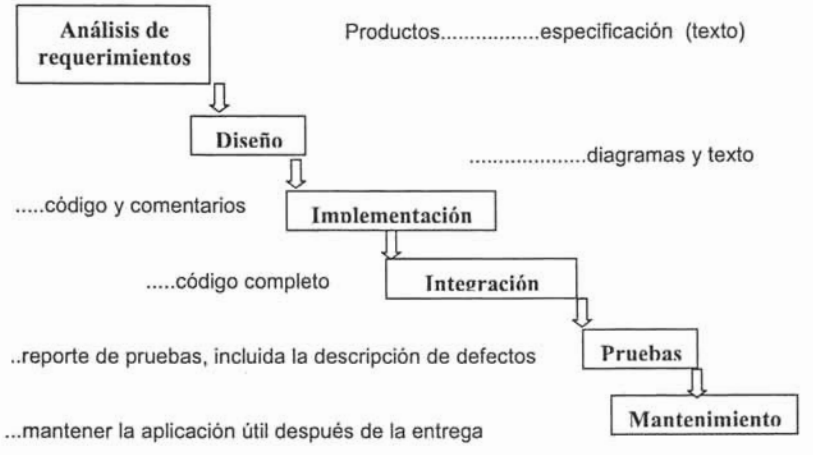


Figura 1 Proceso en cascada autor Braude editorial Alfaomega pag 26

El proceso en cascada es la base o punto de referencia para la mayoría de los procesos y siempre debe considerarse como un proceso alternativo. Las etapas de este proceso se muestra en la Figura 1.

1.2 Proceso en espiral

El proceso en espiral reconoce la necesidad de pasar por la secuencia, del análisis de requerimientos, diseño, implementación y pruebas más de una vez[2].

[2] Ingeniería de software , Una perspectiva orientada a objetos autor Fraude pag 27

Las razones son las siguientes:

- Eliminar los riesgos.
- Construir una versión parcial del producto que se puede mostrar al cliente para obtener una retroalimentación.
- Evitar la integración de una base de código grande, toda a la vez.

La idea es construir cada versión sobre el resultado de la anterior. Este proceso repetitivo forma una especie de trayectoria en espiral, como se ilustra en la Figura 2.

El proceso en espiral se ajusta al avance de los proyectos típicos, requiere una administración mucho más cuidadosa, esto se debe a que la documentación debe ser consistente, siempre que el proceso termine una iteración completa.

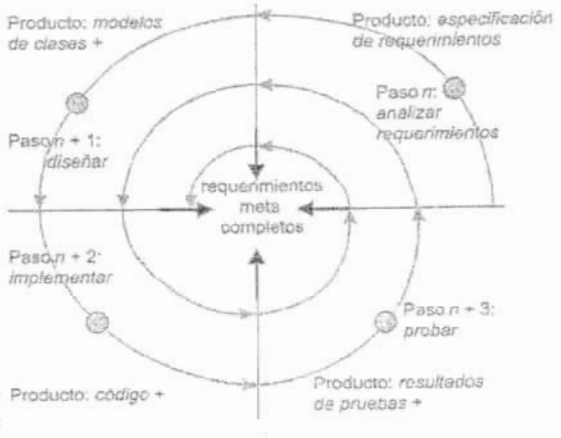


Figura 2 Proceso en espiral autor Braude editorial Alfaomega pag 27

1.3 Proceso por incrementos

Este modelo de proceso es más adecuado en las últimas etapas del proyecto, cuando el producto está en mantenimiento o cuando el producto propuesto es muy similar a otro desarrollado antes. Para realizar un desarrollo por incrementos, lo normal es elegir un intervalo, todo el proyecto se actualiza en cada intervalo. En la figura Figura 3 se muestra este proceso.

Este proceso es muy difícil de coordinar, porque cada documento puede haber cambiado más de una vez, en un solo intervalo.

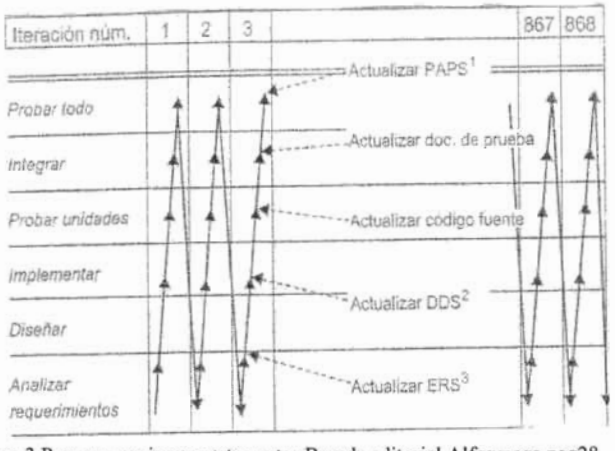


Figura 3 Proceso por incrementos autor Braude editorial Alfaomega pag28

1.4 Proceso unificado de desarrollo de software (PUDS)

Debido a que los enfoques iterativos repiten todas las partes del proceso en cascada, puede ser complicado describirlos. El PUDS es un proceso iterativo que intenta resolver este problema con una clasificación de iteraciones en cuatro grupos.

- Iteraciones de concepción: iteración preliminar con los interesados:
  - Cliente preliminar
  - Usuarios
  - Inversionistas financieros
- Iteraciones de elaboración: finalización de qué se desea y necesita; establecer la base de la arquitectura.
- Iteraciones de construcción: dan como resultado la capacidad operativa.
- Iteraciones de transición: terminar la liberación del producto.

El PUSD no muestra la etapa de "integración" especifica, esto se debe a la creencia expresada por Booch [Booch-UML] que las aplicaciones OO pueden y deben emplear la integración continua.

En otras palabras al añadir partes nuevas, se integra toda la aplicación y se elimina la necesidad de una etapa de integración designada.

1.5 Documentación

La documentación es la base que alimenta la ingeniería de software. Esto incluye la documentación separada del código, al igual que la documentación asociada con él. Sin la documentación completa de un fragmento de código es imposible interpretarlo, por lo tanto tiene muy poco valor.

Cuando se insertan comentarios y se dan nombres más expresivos, el resultado aunque mejora, puede ser que deje la idea equivocada de su significado y contexto, las consecuencias pueden ser desastrosas. Sin embargo, cuando se ve un código bien documentado, su significado es mucho más claro.

La documentación adquiere significado no sólo de su texto, también de su contexto. Una buena documentación prepara a un nuevo ingeniero para que pueda entender el proyecto en un tiempo razonable. Un proyecto es un conjunto de artefactos bien construido, coordinado, que incluye documentación, resultados de pruebas y código.

1.6 Estándares de documentación

Los estándares hacen posible la interoperabilidad; en otras palabras, una idea o artefacto donde artefacto es cualquier tipo de dato, código o información producida o usada por una aplicación, se puede llevar a otra. Los estándares mejoran la comunicación entre los ingenieros de software.

A continuación se mencionan algunos estándares.

- **Calidad**

  Una función de calidad, consiste en un código que:

  - Satisface los requerimientos establecidos con claridad.
  - Verifica sus entradas; reacciona de manera predecible a entradas ilegales.
  - Se ha inspeccionado de manera íntegra por ingenieros que no son el autor.
  - Se ha probado de modo exhaustivo en varias formas independientes.
  - Está bien documentado.
  - Tiene una tasa de error confiable conocida, si los tiene.

  Un diseño de calidad casi siempre

  - Se extiende, es fácil mejorarlo para proporcionar funcionalidad adicional.
  - Evoluciona, se puede adaptar con facilidad a requerimientos diferentes.
  - Se mueve, se aplica a varios entornos.
  - Es general, se aplica a varias situaciones diferentes.

La meta es especificar los estándares de aceptación y crear productos que satisfagan estas especificaciones, se debe saber como cuantificar la calidad, especificar metas y controlar el avance hacia estas.

➢ Métricas

La cuantificación es una parte esencial de la ingeniería. Los ingenieros de software usan métricas como el número de líneas de código, número de defectos encontrados en cierto tiempo o número de funciones por clase, etcétera.

Las métricas que siempre se incluyen son:

- Cantidad de trabajo realizado, medido físicamente (como líneas de código).
- Tiempo que toma realizar el trabajo.
- Tasa de defectos (defectos por 1000 líneas de código, defectos por página de documento).

Los valores previstos o deseados para las métricas se pronostican antes de realizar el esfuerzo, y después se comparan los resultados, de los valores obtenidos con lo pronosticado.

1.7 Proceso de aseguramiento de calidad (QA)

La principal responsabilidad de la calidad de un artefacto radica en la persona que lo crea. La función de QA[5] (aseguramiento de calidad) incluye revisiones, inspecciones y pruebas. Es ideal que el administrador del proceso de QA esté involucrado en garantizar que se usa un proceso sólido y que la documentación se mantiene actualizada.

Técnicas de caja negra y caja blanca

Las técnicas de caja negra de QA manejan aplicaciones o aparte de ellas, que ya están construidas. Estas técnicas verifican si el software cumple con sus requerimientos.

Las técnicas de caja blanca de QA se aplican a componentes que forman la unidad que se está probando y se revisa que haga lo que deben hacer.

1.8 Introducción a la inspección

Una inspección es una técnica de caja blanca para asegurar la calidad. Consiste en examinar las partes del proyecto para encontrar defectos.

El principio de inspección se puede extender a tres reglas:

1. Las inspecciones excluyen de modo específico la reparación de defectos.
2. Un grupo de ingenieros de software debe realizar las inspecciones.
3. Para moderar costos, una persona puede desempeñar dos roles.
   - El moderador es responsable de ver que se lleve a cabo la inspección de modo apropiado.
   - El autor es responsable del trabajo en sí y reparar los defectos encontrados.
   - El líder es responsable de conducir al equipo a revisar el producto de una forma adecuada e integral.
4. Preparación completa. Se requiere que los participantes en inspecciones se preparen al mismo nivel que el desarrollador.

[5] Notación tomada del libro "Ingeniería de software, Una perspectiva orientada a objetos" autor Braude

Las inspecciones deben aplicarse lo más pronto posible en el proceso de desarrollo de software, inspeccionar los requerimientos, planes de administración y configuración del proyecto.

Pasos para ejecutar una inspección:

1. El proceso de inspección comienza con la planeación, que incluye decidir las métricas de inspección que se recolectarán e identificar las herramientas con las que se registrara y analizar los datos.
2. Conforme avanza el proyecto, el líder del proyecto deben determinar qué grupo de ingenieros debe inspeccionar qué fragmentos del trabajo.
3. Si es necesario, se debe organizar una junta de visión general para explicar la unidad sometida a inspección.
4. Etapa de preparación. En esta etapa los inspectores revisan el trabajo con todo detalle.
5. Una vez que todos los participantes están preparados, se lleva acabo la junta de inspección.
6. El autor es capaz de reparar todos los defectos, ésta es la etapa de retrabajo.
7. Si los defectos se deben a malos entendidos o errores conceptuales se debe convocar a una junta separada del análisis en las que se discutan estas causas.
8. La junta de seguimiento. En ella el moderador y el autor confirman que los defectos se repararon.
9. El grupo se reúne para revisar el proceso de inspección y decide como puede mejorarse este.

Los proyectos cambian de dos formas al avanzar hacia la terminación. La primera es mediante la acumulación de partes nuevas. La segunda consiste en las versiones sucesivas de esas partes. La **"administración de la configuración"** revisa y organiza esas partes.

Nuestra primera obligación al evaluar un software es saber exactamente dónde se localiza las partes del proyecto y cuáles van juntas. Las "partes", consisten en mucho más que sólo el código fuente, incluye módulos y toda la documentación.

CAPÍTULO II

ADMINISTRACIÓN DEL PROYECTO

La administración de proyectos consiste en gestionar la generación de un producto dentro del tiempo dado y los limites de costos. La administración del proyecto involucra no sólo la organización técnica y las habilidades organizacionales, sino también la habilidad de coordinar personas.

Componentes de la administración de proyectos

- Estructura (elementos organizacionales involucrados).
- Proceso administrativo (responsabilidades y supervisión de los participantes).
- Proceso de desarrollo (métodos, herramientas, lenguajes, documentación y apoyo).
- Programa (tiempos en los que deben realizarse las porciones del trabajo).

2.1 Variables principales: costo, capacidad y programa

Quienes planean un proyecto pueden definir costos, calidad y fecha de entrega. La manera en que estos tres factores pueden controlarse depende del proyecto. Los costos pueden estar fijos de antemano, muchas veces se puede dar una cierta flexibilidad para que estos cambien.

Las capacidades tampoco son fijas ya que el cliente puede estar de acuerdo con eliminar requerimientos y esto ayuda a reducir el tiempo de terminación de un proyecto. Las metas de calidad también pueden variar, cuando estas no cumplen con lo establecido por el cliente, este puede quedar insatisfecho, cuando las metas son muy altas, el costo de encontrar el detalle más pequeño, lleva a un consecuente retraso en el proyecto.

Autores como Humphrey[Hu7] aconsejan a los ingenieros de software no prometer fechas de entrega para los productos sin antes verificar con cuidado lo razonable que esas fechas pueden ser.

Una buena práctica administrativa es crear agendas de las reuniones y darles seguimiento.

Manera de especificar una agenda

1. Llegar a un acuerdo sobre la agenda y la asignación de tiempos.
2. Elegir voluntarios para
   - Registrar las decisiones tomadas.
   - Vigilar el tiempo y apresurar a los participantes.
3. Informar el progreso del proyecto.
4. Presentar los artefactos preliminares.
5. Analizar la mitigación del riesgo.
6. Definir métricas y proponer mejoras al proceso.

La experiencia muestra que el número de ingenieros con los que interactúa cada desarrollador es por lo regular entre tres y siete. Los estudios formales acerca del efecto del tamaño de equipo sobre el desempeño del mismo son raros.

En un extremo, el desarrollador trabajaba de manera individual sin interacción habitual con otros. Aunque no gaste tiempo en comunicación ese aislamiento suele repercutir en malos entendidos respecto a lo que se espera del desarrollador.

El ingrediente principal requerido para producir software es la gente. Las aptitudes técnicas de los ingenieros con las que cuentan, deben ser aprovechadas en el momento adecuado, en el tiempo correcto. Esto requiere una combinación de trabajo en equipo y liderazgo.

Los factores que a la larga ocasionan que un proyecto fracase aparecen como riesgos cuando se reconocen con prontitud y al reconocerlos tal vez pueda prevenirse el fracaso mediante la acción adecuada.

Tipos de riesgos	Atributos
Riesgos que pueden evitarse	• Probabilidad de ocurrencia en el proyecto • Impacto en el proyecto
Riesgos que no pueden evitarse	• Probabilidad de ocurrencia en el proyecto • Impacto en el proyecto

Si los riesgos del primer tipo se identifican con suficiente prontitud, su mitigación convierte un proyecto con posibilidades de fracasado en uno muy posiblemente exitoso. También es de gran beneficio reconocer el riesgo del segundo tipo.

Un proyecto puede detenerse antes de malgastar recursos o es posible cambiar el enfoque o agregar personal para mitigar el riesgo. Los equipos efectivos adoptan una metodología de riesgo donde los riesgos son buscados todo el tiempo.

Actividades de la administración del riesgo

1. Identificar el riesgo
2. Calcular su severidad, clasificarlo.
3. Mitigación.

Estas actividades deben llevarse a cabo desde el principio del proyecto, y continuar de manera disciplinada.

La identificación de los riesgos consiste en escribir todas las inquietudes o preocupaciones de quienes están relacionados con el proyecto, después motivar continuamente a los integrantes del equipo a pensar en más inquietudes.

El líder del proyecto tiene la mayor responsabilidad de mitigar los riesgos para el éxito de un proyecto.

Existen tres maneras de mitigar un riesgo.

- Una es hacer cambios en los requerimientos del proyecto para evitar el aspecto que causa el riesgo (**evasión**).
- [1]Reorganizar el proyecto de manera que algo o alguien externo al proyecto asuma el riesgo (**transferencia**).
- Otra manera es desarrollar técnicas y diseños que resuelvan el problema (**aceptación**).

2.2 Elección de herramientas de desarrollo y soporte

Debe identificarse el lenguaje o lenguajes para el desarrollo al inicio del proyecto. En ocasiones esta decisión es directa, como cuando una organización impone un lenguaje o cuando el lenguaje es el único capaz de desarrollar los requerimientos.

Los proyectos requieren apoyo para los administradores del sistema de la red, de las bases de datos. El administrador del proyecto debe asegurar que estas personas están disponibles.

Con la información y el trabajo realizado hasta el momento se puede desarrollar la primera versión de un calendario donde se puede mostrar la distribución de tiempos.

[1] Tesis de Maestría, "Integración del contexto técnico y tecnológico al proceso de desarrollo para la generación de software de calidad" de Gustavo Adolfo Arrellano Sandoval pag 71

CAPÍTULO III

ANÁLISIS DE REQUERIMIENTOS

Los requerimientos detallados constituyen el único lugar donde se escribe la naturaleza exacta de la aplicación. Los ingenieros de software necesitan una base para el diseño e implementación de la aplicación.

El análisis de requerimientos se divide en dos niveles. El primer nivel documenta los deseos y necesidades del cliente y se expresa en lenguaje claro para él. Los resultados suelen llamarse "requerimientos del cliente" o "requerimientos C".

El segundo nivel documenta los requerimientos de manera específica y estructurada. Éstos se llaman "requerimientos del desarrollador" o "requerimientos D". También se conoce como "requerimientos específicos", "especificaciones funcionales".

Los requerimientos D son una lista completa de las propiedades específicas y la funcionalidad que debe tener la aplicación, expresada con todo detalle.

Cada uno de estos requerimientos se enumera, etiqueta y se revisa durante toda la implementación.

En general, se supone que los desarrolladores leerán los requerimientos D. Los clientes también están interesados en ellos y casi siempre pueden comprender y comentar muchos de ellos.

Secuencia común de actividades para reunir y documentar los requerimientos D

1. Seleccionar la organización de requerimientos D.
2. Crear diagramas de actividad a partir de los casos de uso.
3. Obtener los requerimientos D a partir de requerimientos C.
4. Describir planes de prueba.
5. Inspeccionar.
6. Validar con cliente.
7. Liberar.

Los requerimientos D se describen a partir de los requerimientos C. En el caso ideal se comienzan a escribir las pruebas de cada uno de los requerimientos al mismo tiempo que éstos. Aunque los requerimientos D están escritos principalmente para desarrolladores, tanto los requerimientos como sus pruebas se revisan con el cliente. Después los requerimientos D deben inspeccionarse y liberarse.

3.1 Tipos de Requerimientos D

➢ Requerimientos funcionales

Los requerimientos funcionales especifican los servicios que debe proporcionar la aplicación.

➢ Requerimientos no funcionales

- Requerimientos de desempeño

Los requerimientos de desempeño especifican las restriciones de tiempo que debe observar la aplicación. Los requerimientos de desempeño son una parte crítica de las aplicaciones en tiempo real donde las acciones deben terminar dentro de límites de tiempo especificado.

- Confiabilidad y disponibilidad

Estos requerimientos especifican la confiabilidad en términos cuantificados. Este tipo de requerimientos reconoce que es poco probable que las aplicaciones sean perfectas por lo que circunscribe su grado de imperfección. La disponibilidad, tiene una estrecha relación con la confiabilidad, cuantifica el grado en el que la aplicación debe estar disponible para los usuarios.

- Manejo de errores

Esta categoría de requerimientos explica cómo debe responder la aplicación a los errores en su entorno. El manejo de errores se refiere a las acciones que debe realizar la aplicación en situaciones excepcionales por parte de los usuarios.

Sin embargo, este tipo de requerimientos de errores debe aplicarse en forma selectiva por que el objetivo es producir aplicaciones libres de defectos, y no cubrir los errores con códigos interminables de manejo de errores. La verificación de errores en la aplicación misma es adecuada para un mínimo de partes críticas de la aplicación.

➢ Requerimientos de interfaz

Los requerimientos de interfaz describen el formato con el que la aplicación se comunica con su entorno.

- Restricciones

Las restricciones de diseño o implementación describen los límites o condiciones para diseñar o implementar la aplicación. Estos requerimientos no pretenden sustituir el proceso del diseño, sólo especifican las condiciones que el cliente impone al proyecto, el entorno u otras circunstancias e incluye la exactitud.

- **Requerimientos inversos**

  Los requerimientos inversos establecen qué no debe hacer el software. Es lógico que haya un número infinito de requerimientos inversos, se seleccionan los que aclaran los requerimientos verdaderos y los que eliminan posibles malos entendidos.

3.2 Rastreo de requerimientos funcionales

Los requerimientos D deben ser completos y consistentes. Cada uno debe poderse rastrear en el diseño y la implementación, probarse en cuanto a su validez e implementarse de acuerdo con la prioridad racional.

Sin un rastreo limpio de cada requerimiento desde el diseño de la aplicación hasta el código real que la implementa, es muy difícil asegurar que la aplicación todavía cumpla con el. Cuando los requerimientos cambian, esto es aun más difícil.

La capacidad de hacer corresponder cada requerimiento con su parte relevante del diseño e implementación se llama rastreabilidad. Una manera de ayudar a lograr esto es mapear cada requerimiento D funcional con una función especifica del lenguaje a usar.

Conforme avanza el proyecto, el documento de requerimientos debe permanecer consistente con el diseño y la implementación. Cuando es difícil rastrear los requerimientos en el diseño y el código los desarrolladores tienden a evitar actualizar el documento de requerimientos al hacer cambios al mismo código fuente debido al gran esfuerzo que implica.

En último caso, el deterioro de este tipo en los documentos da como resultado un gasto mucho mayor en el desarrollo y mantenimiento.

Los documentos se relacionan de manera estricta y sencilla unos con otros y la administración determina que la documentación es un requerimiento de desempeño. Dicho de otra manera, el sistema debe hacer la correspondencia entre los requerimientos D, el diseño y el código debe ser muy claro y concreto.

3.3 Rastreo de requerimientos no funcionales

Una meta de la etapa de diseño es aislar cada requerimiento no funcional en un elemento de diseño separado. En el caso de los requerimientos de desempeño se intenta aislar las unidades de procesamiento más lentas.

Para validar los requerimientos no funcionales cada uno se liga a un plan de pruebas, de preferencia en el momento de escribir el requerimiento.

Debe ser posible probar un requerimiento cuando este se ha implementado de manera apropiada. Los requerimientos que se pueden probar se llaman comprobables. Los requerimientos no comprobables tienen poco valor práctico.

A menos que un requerimiento D se escriba con claridad y sin ambigüedad, no será posible determinar si se ha implementado bien. En general es difícil llevar a la práctica toda la funcionalidad planeada de una aplicación, y dentro del presupuesto. Puede variar una o más de las características de capacidades, programa de tiempos, nivel de calidad y costo.

Asi no puede cambiarse la programación, del presupuesto y la calidad, la única alternativa que varía es la capacidad, es decir, reducir el requerimiento que se va a implementar. Este proceso se realiza de manera planeada. Una técnica es asignar prioridades a los requerimientos específicos.

Jerarquizar todos los requerimientos casi siempre es una perdida de tiempo, en su lugar, muchas organizaciones clasifican los requerimientos en tres categorías.
Se llaman "esenciales", "deseables" y "opcionales". El proyecto debe incluir los requerimientos "esenciales".

Asignar prioridades a los requerimientos tiene impacto en el diseño por que con frecuencia los requerimientos deseables y aplicables indican hacia donde se dirige la aplicación.

Para cada requerimiento se pregunta qué pasaría si ocurriera en las circunstancias equivocadas. Una falta de condiciones de error en las especificaciones de los requerimientos resalta de manera especial cuando se prueba ya que quien realiza la prueba fuerza las condiciones de error y debe saber que salida del requerimiento se espera.

Un conjunto de requerimientos D es consistente si no hay contradicciones entre ellos. Cuando el número de requerimientos D crece, la consistencia puede disminuir.

La organización orientada a objetos de los requerimientos ayuda a evitar inconsistencias mediante la agrupación de los requerimientos D en clases, sin embargo esto no es una garantía de consistencia y por ello las inspecciones de los requerimientos las verifican.

3.4 Métodos para organizar los requerimientos

Los requerimientos D se pueden organizar de acuerdo con varios esquemas entre ellos están los siguientes:

- Por características observables de la aplicación.
- Por casos de uso: la idea es que la mayoría de los requerimientos son parte de un caso de uso.
- Por jerarquía de función: descomponiendo la aplicación en un conjunto de funciones de alto nivel.
- Por estados indicando los requerimientos específicos aplicados a cada estado.

El método para organizar los requerimientos D con frecuencia se relaciona con la arquitectura probable de la aplicación. Si el diseño debe orientarse a objetos la organización por casos de uso o por clases debe considerarse debido a que facilita la rastreabilidad.

CAPÍTULO IV

ARQUITECTURA DE SOFTWARE

Arquitectura de software es equivalente a "diseño en el más alto nivel". La especificación clara de la arquitectura de software, es importante para todas las aplicaciones, y es indispensable en trabajos de desarrollo con más de una persona.

Esto se debe a que las aplicaciones grandes deben diseñarse e implementarse en partes (módulos) y después ensamblarse. Los ingenieros encargados de la tarea de desarrollar la arquitectura son generalmente los mas experimentados.

Las aplicaciones requieren de manera invariable, componentes tanto de software como de hardware. La ingeniería de software es el proceso de diseño y análisis que desglosa una aplicación en hardware y software.

4.1 Metas de la selección de la arquitectura

Para un proyecto de desarrollo de software, pueden existir varias arquitecturas adecuadas, para decidir cuál es la mejor depende de las metas de proyecto.

Las metas de diseño mas importantes son:

- Extensión: Facilitar la adición de nuevas características.
- Cambio: Facilitar los cambios en los requerimientos.
- Sencillez: Hacer de fácil compresión e implementación.
- Eficiencia: Lograr alta velocidad: ejecución y/o compilación.

El problema principal de los sistemas de software es la complejidad, no el número de líneas de código en sí. Una manera de evitar la complejidad es descomponer el problema para que tenga las características de un programa mas pequeño.

La descomposición (modularización) del problema tiene una importancia crítica y es uno de los mayores retos.

4.2 Metas de descomposición del software

Hay dos medidas a considerar al hacer la descomposición.

- Cohesión dentro de un módulo es el grado de comunicación entre los elementos del módulo.
- Acoplamiento es el grado en que los módulos se comunican con otros módulos.

La modularización efectiva se logra al maximizar la cohesión y minimizar el acoplamiento. Esto hace posible descomponer tareas complejas en otras más sencillas.

Es más sencillo modificar las arquitecturas de software con bajo acoplamiento y alta cohesión, ya que los cambios tienden a tener efectos locales sobre ellos.

La diferencia entre proyectos de gran escala y pequeña escala es la cantidad de módulos o paquetes anidados o inmersos. Los proyectos de gran escala organizan cada paquete de alto nivel en subpaquetes.

La descomposición perfecta es una meta valiosa, pero difícil de lograr. El software no es el único tipo de ingeniería en la que es difícil lograr módulos limpios y claros. La estructura de estos subpaquetes es la arquitectura de la aplicación

Para definir la arquitectura se debe:

1. Desarrollar un modelo de la aplicación a un nivel alto con una estructura jerárquica.
2. Descomponer los componentes requeridos.
    - Buscar una cohesión alta y acoplamiento bajo.
3. Repetir este proceso para los componentes.

CAPÍTULO V

DISEÑO DETALLADO

El diseño detallado es la actividad técnica que sigue a la selección de la arquitectura. Su meta es preparar por completo el proyecto para su implementación. Con base a los requerimientos, los ingenieros seleccionan una arquitectura.

En el diseño de software se verifica que las clases y los métodos que especifican el diseño detallado sean capaces de representar los casos de uso requeridos.

Para describir los módulos se puede se usar seudocódigo o diagramas de actividad. A continuación se describe cada uno indicando tanto sus ventajas como desventajas.

Seudocódigo

Un seudocódigo es un medio para expresar un algoritmo como texto sin tener que especificar detalles del lenguaje de programación.

Ventajas

- Es fácil de entender.
- Es suficientemente preciso para expresar los algoritmos.
- Se puede inspeccionar la correctez de los algoritmos.
- Se obtiene una taza de defectos en el seudocódigo.

Desventajas

- Crea un nivel adicional de documentación que se debe mantener.
- Introduce posibilidades de errores al trasladar el código.

5.1 Diagramas de actividad

El diagrama de actividad de UML es un diagrama de flujo del proceso que se usa para modelar el comportamiento del sistema. Los diagramas de actividad se pueden usar para modelar un caso de uso, una clase o un método.

Ventajas

- Ofrece una herramienta gráfica para modelar el proceso de un caso de uso.
- Se pueden usar para listar los pasos de un caso de uso.
- Es provechoso para entender el comportamiento de alto nivel de la ejecución de un sistema.

Desventajas

- La realización de diagramas podría ser tedioso.

Una razón para esta falta de rehusó de software es la organización inadecuada de arquitecturas, diseños y códigos existentes. Dado el gran número de métodos empaquetados en cada clase, la funcionalidad que se requiere con frecuencia está incluida en clases que se pueden reutilizar.

5.2 Diagramas de secuencia

Los diagramas de secuencia son construidos a partir de los casos de uso y proporciona las clases involucradas con los métodos requeridos para ejecutar las secuencias.

Manera de refinar los modelos para el diseño detallado

1. Comenzar con los diagramas de secuencia construidos para los requerimientos detallados de la arquitectura correspondiente a los casos de uso.
2. Introducir casos de uso adicionales, si es necesario, para describir cómo interactúan las partes del diseño con el resto de la aplicación.
3. Proporcionar diagramas de secuencia con detalles completos.

Los diagramas de clases detallados deben incluir todos los nombres de atributos, sus tipos, operaciones, visibilidad y tipos de resultados.

Una vez identificadas las funciones que se deben programar, se debe escribir el algoritmo que se usará de manera que sea el código fuente. La ventaja de esto es que los ingenieros de software pueden inspeccionar el algoritmo por separado, sin las complejidades de la programación, para detectar los defectos antes de que se conviertan en defectos de código.

CAPÍTULO VI

IMPLEMENTACIÓN

La "implementación" se refiere a la programación, su propósito es satisfacer los requerimientos de la manera que especifica el diseño detallado.

Pasos para preparar la implementación

1. Confirmar los diseños detallados que deben implementarse.
2. Preparar la medición del tiempo dedicado.
3. Prepara una forma para el registrar de los defectos.
4. Comprender los estándares requeridos.
5. Estimar el tamaño y el tiempo con base en sus datos anteriores.
6. Planear el trabajo en segmentos de mas menos 100 LOC (líneas de código).

El seudocódigo del documento de diseño detallado se puede convertir en código con comentarios. Se desarrolla un plan de pruebas para cada unidad. Una vez que se implementa un modelo, se puede usar una herramienta de ingeniería inversa para generar los aspectos del diseño detallado.

La ingeniería inversa sería aconsejable después de la implementación inicial por que el código fuente tiende a estar más actualizado que el nivel más bajo del diseño detallado. También puede ser útil para el código heredado cuando el diseño está mal documentado. En términos generales, la ingeniería inversa debe ser un recurso sólo si en verdad es necesaria.

6.1 Implementación en el proceso unificado de desarrollo de software

El proceso unificado de desarrollo de software (PUDS) da nombres a los grupos de iteraciones. La mayoría de las iteraciones involucra cierta implementación e iteraciones de construcción.

Aunque las partes del diseño deben corresponder lo más fielmente posible a las del sistema de archivos físicos, tal vez no sea práctico un simple mapeo. Varias clases pueden corresponder a un archivo y los artefactos de implementación pueden incluir archivos fuente, archivos objetos y versiones comprimidas.

El modelo de implementación muestra la organización de los artefactos físicos programados y la localización de los elementos de diseño en ellos. El modelo de implementación consiste en subsistemas anidados. Éstos contienen componentes como archivos e interfaces de implementación.

6.2 Lenguajes de programación

Existe una gran variedad de lenguajes de programación que van de los especializados a los generales.

La imagen popular de la programación como el acto de someter material tecleado a un compilador es sólo una pequeña parte. La meta real de la ingeniería de software es crear el código correcto, pero los compiladores sólo pueden verificar la sintaxis y generar el código objeto.

Una gran parte de la programación se dirige al manejo de errores. Para disminuirlo es necesario un enfoque disciplinado.

La meta real es la prevención de errores y no su corrección. Utilizar un proceso bien definido, inspeccionar las etapas, es la primera línea de defensa esencial. Una segunda línea de defensa para manejar datos potenciales ilegales es interactuar con la fuente de datos hasta que la entrada cambie a una legal antes de continuar con el proceso.

Esto es posible para una gran parte de la programación de interfaces de usuario, donde con frecuencia se puede asegurar que sólo se permiten entradas legales.

Un defecto es un error, pero un valor predeterminado arbitrario no especificado de manera explícita en los requerimientos es un encubrimiento.

6.3 Estándares de programación

El uso de estándares mejora la disciplina, lo legible y lo portable de un programa. Los ingenieros de software tienden a ser emocionales en cuanto a sus convenciones favoritas de nombrado de variables y generalmente el consenso es imposible. De cualquier manera, las convenciones son necesarias.

El siguiente es un ejemplo de convenciones de nombres.

Inicie los nombres de clase con C mayúscula.

Por ejemplo: **Ccliente**

6.4 Herramientas y entornos para programación

Los entornos de desarrollo interactivos (IDE) tienen un uso amplio para permitir que los programadores produzcan más código en menos tiempo. Incluyen características de "arrastrar y soltar" para formar los componentes de la interfaz gráfica, representaciones gráficas de los directorios, depuradores, ayudas automáticas y otros mas.

La historia de las herramientas en otras ramas de la ingeniería de software (como CAD/CAM) sugiere que las herramientas de programación tendrán mejoras significativas continuas que seguirán apoyando las habilidades de programación y reducirán las tareas penosas y mecánicas.

Las "líneas de código" constituyen una medida útil, aunque no perfecta, porque el número de líneas de código no garantiza que el proyecto tendrá la funcionalidad requerida.

CAPÍTULO VII

PRUEBAS

Las pruebas consisten en probar las partes estructurales de una aplicación en desarrollo. Al probar no se puede demostrar que una aplicación no tiene defectos, como pueden hacer las pruebas de que es correcto. Las pruebas sólo pueden mostrar la presencia de defectos. A menudo hacer pruebas se malinterpreta en esencia como un proceso de establecer la confianza, como en "probar para asegurar que es correcto".

Un propósito importante de las pruebas es casi opuesto al de establecer la confianza. El propósito no es demostrar que una aplicación es satisfactoria, sino determinar con firmeza en qué parte no lo es.

El tiempo dedicado a las pruebas implica un gasto considerable, y se intenta obtener el máximo beneficio de ese gasto. Entonces, el propósito de probar es encontrar el mayor número de defectos, con el más alto nivel de severidad posible.

Las pruebas son responsables de más de la mitad del tiempo dedicado a los proyectos. La recompensa por encontrar un defecto pronto en el proceso, es de al menos un ahorro de diez veces comparado con detectarlo en la etapa de integración o peor aun después de la entrega.

Cuando un ingeniero de software desarrolla un código se forma una visión de lo que debe hacer ese código y, al mismo tiempo, desarrolla circunstancias típicas en las que debe ejecutarse el código. Sin duda puede suponerse que el código mostrará algunos problemas en esas circunstancias particulares. De manera consiente o no, éstas constituyen los casos de prueba del desarrollador. Así, cuando un individuo prueba su propio código, tiende a ocultar justo eso que debe descubrir.

Las "pruebas de unidades" son el primer tipo de prueba que se aplica. El siguiente nivel consiste en las pruebas de integración. Éstos validan la funcionalidad global de cada etapa de la aplicación parcial. Por último, las pruebas del sistema y de aceptación validan el producto final.

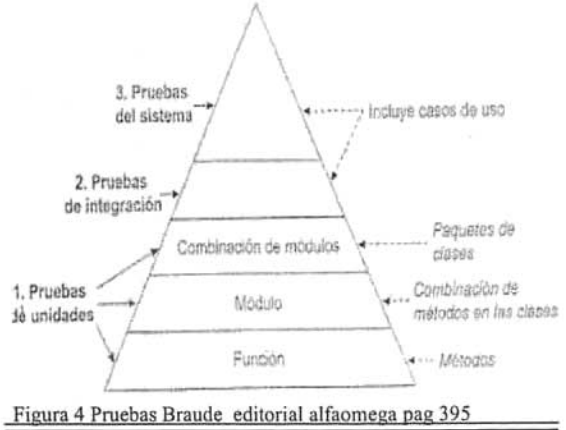


Figura 4 Pruebas Braude editorial alfaomega pag 395

La Figura 4, ilustra este tipo de pruebas y sus relaciones. Las unidades a las que se aplican las "pruebas de unidades" son los bloques de construcción de la aplicación.

Una prueba de unidad es un complemento de la inspección y del uso de la formalidad correcta. En términos del proceso de desarrollo de software unificado, las pruebas unitarias se llevan a cabo durante las iteraciones de elaboración y también en las primeras iteraciones de construcción.

7.1 Tipos de pruebas

➢ Pruebas de caja negra, blanca y gris

Cuando el único interés es si una aplicación, o parte de ella, proporciona la salida adecuada, se prueba que cumple cada requerimiento al usar la entrada apropiada. Esto se llama prueba de caja negra porque no se toma en cuenta el interior de la caja (la aplicación).

Las pruebas de caja negra pueden ser suficientes si se puede asegurar que agotan todas las combinaciones de entrada. Esto probaría al cliente que todos los requerimientos se satisfacen.

Las pruebas de caja blanca prueban las líneas de falla más comunes en la aplicación . Para realizar pruebas de caja blanca, primero se desglosa el diseño de la aplicación en busca de trayectorias y otras particiones de control y datos.

Las pruebas de "caja gris" consideran el trabajo interior de la aplicación o unidad bajo prueba, pero solo en un grado limitado, también pueden incluir aspectos de caja negra. La Figura 5 ilustra este tipo de pruebas.

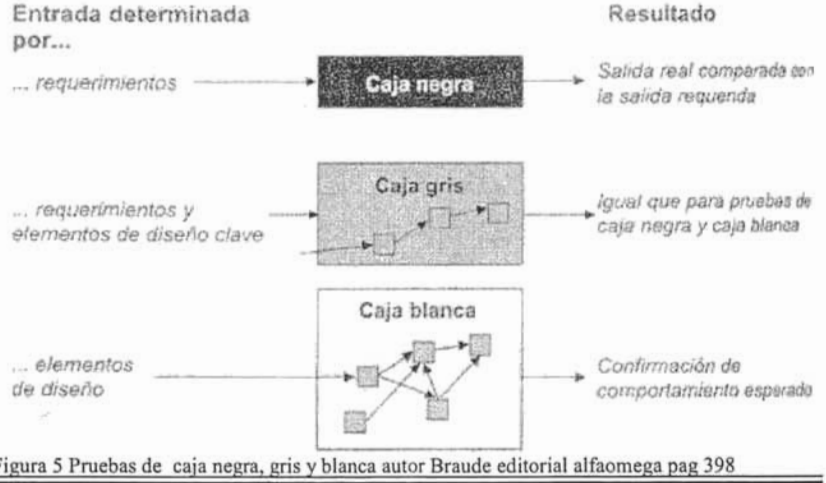


Figura 5 Pruebas de caja negra, gris y blanca autor Braude editorial alfaomega pag 398

7.2 Planeación de pruebas de unidades

Se requiere un enfoque sistemático para la ejecución de los planes de pruebas porque el número de unidades potenciales que se debe probar suele ser grande. Es fácil establecer que "cada parte del trabajo debe probarse" sin embargo, esto tiene poco significado porque se asigna sólo una mínima cantidad de recursos a la etapa de pruebas.

Pasos para el plan de pruebas de unidades

1. Decida la filosofía de las pruebas de unidades.
2. Decida qué, dónde y cómo documentar.
3. Determine el grado de las pruebas de unidades.
4. Decida cómo y dónde obtener los datos para las pruebas.
5. Estime los recursos requeridos.
6. Registre tiempo, conteo de defectos, tipo y fuente.

CAPÍTULO VIII

INTEGRACIÓN

Debido a que la mayoría de las aplicaciones son complejas, deben construirse con partes que primero se desarrollan por separado. La "integración" se refiere al proceso de ensamble de las partes desarrolladas por separado en las etapas previas en el proceso de desarrollo de software.

La etapa de integración del proceso en cascada suele producir sorpresas desagradables debido a la incompatibilidad de las partes que se integran. Por ello, el proceso unificado de desarrollo de software, en particular, intenta evitar la integración "masiva" es decir la unión de todas las partes en una sola etapa, mediante la integración continua con múltiples iteraciones.

Cuando se aplica a la integración, la verificación se reduce a confirmar que se están uniendo justo las componentes que se planeó ensamblar, justo en la forma que se planeó ensamblarlas. Esa verificación se puede realizar mediante la inspección de los productos de la integración.

Una vez hecha la integración total o parcial del código para el sistema, es posible probar las partes en el contexto de todo el sistema en lugar de aislarlas. Para enfocar las pruebas en las partes designadas se debe idear una entrada apropiada.

Aunque el proceso de construcción típico tiene la desventaja de trabajar con unidades incompletas, posee la ventaja de ejercer la integración antes en el proceso de desarrollo. Esto ayuda a eliminar los riesgos al evitar la integración "masiva".

Se simplifica con la incorporación de implementaciones de casos de uso completos en cada construcción en lugar de sólo partes de los casos de uso. Crear casos de uso relativamente pequeños desde el principio facilita su ajuste en las construcciones.

Como las interfaces de usuario tendrán que construirse y probarse en algún momento, es preferible construir interfaces temporales para usarlas durante las pruebas de integración.

8.1 Procesos de pruebas

Los casos de uso son una fuente ideal de casos de prueba para las pruebas de integración. La idea es que los casos de uso se construyan sobre los que ya están integrados para formar pruebas cada vez más representativas del uso de la aplicación.

Por lo general, las construcciones consisten en el código de varios desarrolladores y es común encontrar muchos problemas cuando se integra el código para crear la construcción. Por ello, se intenta comenzar la integración y las pruebas de integración pronto en el proceso de desarrollo de software, para ejecutar el código en su contexto final.

Pruebas de integración sugeridas por el proceso unificado de desarrollo de software (PUDS).

- Modelo de casos de uso: conjunto de casos de uso que describen el uso típico de la aplicación y los diagramas de secuencia que los describen con detalle.
- Casos de prueba: los datos de entrada para cada prueba.
- Procedimientos de prueba: la manera en que se deben establecer y ejecutar las pruebas, y evaluar los resultados.
- Evaluación de pruebas: resumen, detalles y efectos de los defectos encontrados.
- Plan de pruebas: plan global para realizar las pruebas.
- Componentes de las pruebas: código fuente para las pruebas en sí, y para el código de la aplicación que se debe probar.
- Defectos: informes de los defectos descubiertos como resultado de este proceso, clasificado por la severidad y tipo.

El proceso unificado de desarrollo de software (PUDS) implica los papeles del ingeniero de pruebas, el ingeniero de componentes y el que prueba el sistema como se muestra en la Figura 6.

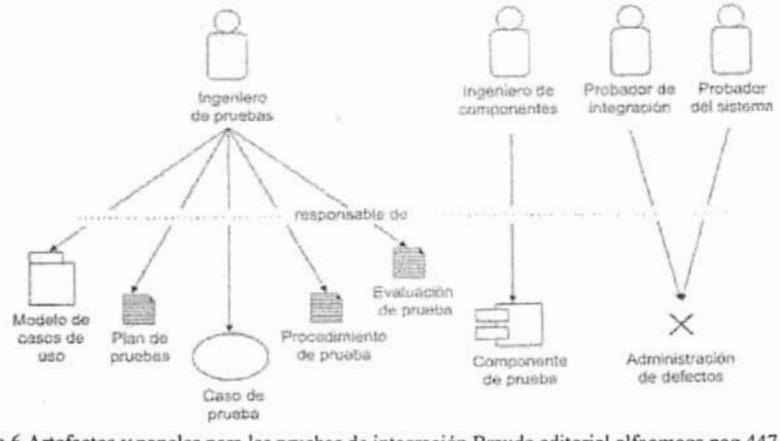


Figura 6 Artefactos y papeles para las pruebas de integración Braude editorial alfaomega pag 447

8.2 Pruebas de interfaz

Muchas fallas de aplicaciones se deben a problemas con las interfaces entre las componentes. En el caso de la ejecución de un proyecto, algunos grupos encuentran más sencillo comunicarse entre ellos que con otros grupos, de manera que es fácil interpretar mal las interfaces de diseño y programación.

Una vez desarrollados los módulos, se pueden probar las interfaces. Esto se hace generando tráfico entre las interfaces, casi siempre en la forma de llamadas de las funciones.

Para probar que las interfaces trabajan como se desea, los métodos de interfaces se pueden llamar en forma secuencial, con diferentes combinaciones. Las secuencias elegidas deben ejecutar muchas combinaciones de métodos de interfaces, pero deben tener sentido en el contexto.

8.3 Pruebas del sistema

La prueba del sistema es la culminación de las pruebas de integración. Consiste en pruebas de caja negra que validan la aplicación completa revisando lo establecido en los requerimientos. Siempre que es posible, las pruebas del sistema se realizan mientras la aplicación se ejecuta en su entorno requerido. Sin embargo, en ocasiones habrá que conformarse con pruebas de ejecuciones del sistema en un entorno o configuración que no es equivalente a la del cliente.

Dado que las pruebas del sistema aseguran que los requerimientos se cumplan, deben validar de modo sistemático cada requerimiento.

El proceso de desarrollo de software unificado intenta organizar la mayoría de los requerimientos mediante casos de uso, en cuyo caso las pruebas son más sencillas.

8.4 Pruebas de usabilidad

Una buena interfaz puede mejorar mucho el valor de una aplicación. La prueba de usabilidad valida la aceptación de la aplicación por los usuarios.

La tarea principal de las pruebas de usabilidad es asegurar que la aplicación satisface los requerimientos establecidos si es útil a los usuarios.

Medidas de usabilidad

- Accesibilidad
  Facilidad con la que entran, navegan y salen los usuarios.
- Rapidez de respuesta
  Qué tan rápido permite la aplicación al usuario lograr sus metas especificas.
- Eficiencia
  Qué tan cortos son los pasos requeridos para la funcionalidad elegida.
- Compresión
  La facilidad con que se entiende y usa el producto mediante la documentación y la ayuda.

8.5 Pruebas de aceptación

La organización desarrolladora y el cliente son las partes de un contrato. Cuando termina un trabajo, un miembro del equipo de desarrollo obtiene una declaración definitiva del cliente donde indique que la aplicación está entregada.

Las pruebas de aceptación están diseñadas para asegurar al cliente que se construyó la aplicación estipulada.

SISTEMA DE MONITOREO (MDS)

CAPÍTULO IX

ADMINISTRACION DEL PROYECTO MDS

9.1 Comprender el contenido, alcance y tiempo del proyecto

La meta del proyecto es programar un conjunto de herramientas que permitan monitorear los elementos básicos de un sistema en plataforma tipo UNIX.

Los elementos básicos del sistema son:

- Servicios abiertos.
- Usuarios con **UID 0**.
- Archivos con **SUID y/o SGID**.
- Monitoreo del sistema de archivos.
- Respaldos.
-

Cada uno de estos módulos será controlado por un programa maestro. Esta herramienta permitirá a los administradores monitorear su sistema, para prevenir un ataque tanto interno como externo, ya que los sistemas de información en la actualidad se encuentran expuestos a distintas amenazas que ponen en riesgo el buen funcionamiento del mismo. Por tal motivo, la seguridad en cómputo se ha vuelto un elemento tan importante que debe ser considerado en la planeación de un sistema o servicio.

El tiempo estimado para la realización de esta aplicación es de 6 meses. El proceso con el cual será desarrollado es el descrito por Braude "Ingeniería de Software una perspectiva orientada a objetos" el cual se describió en los primeros ocho capítulos de este trabajo.

Esta tabla muestra las fechas de inicio y termino en las que se realizaran cada uno de los documentos para el desarrollo del proyecto.

Documentos	Dic	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago
Administración del proyecto	5-11								
Requerimientos		6-14							
Arquitectura de software			2-18						
Diseño detallado				2	27				
Implementación						3	22		
Pruebas								5-23	
Integración								29	10

9.2 Identificación y administración de riesgos en el desarrollo

Un riesgo es algo que puede ocurrir en el curso de un proyecto que, en el peor de los casos lo afectaría de manera negativa y significativa.

Mitigar los riesgos

Mitigar los riesgos es el proceso mediante el cual los riesgos se reducen o incluso se anulan.

Los riesgos identificados en el proyecto y su contención se muestran en la siguiente tabla:

RIESGO	CONTENCIÓN
Pérdida de la información del proyecto que se encuentra almacenada en la computadora debido a una falla en el sistema.	Para mitigar el riesgo se almacenará la información del proyecto en unidades de almacenamiento secundarias. Con lo que se contará con un respaldo total del proyecto en caso de un accidente o perdida de la información.
Que el desarrollo de la interfaz no se termine lo que ocasionaría que el proyecto no se concluyera en su totalidad.	Se recopilará toda la información posible sobre el desarrollo de interfaces con perl/tk para que la interfaz se termine y así concluir el proyecto.
Que el sistema operativo donde el proyecto será desarrollado no cuente con las herramientas necesarias para el desarrollo. Estas herramientas son: ➢ Compilador de perl. ➢ Tk para el desarrollo de la interfaz.	Este riesgo se puede evitar revisando que el sistema operativo cuente con el compilador de perl y con perl/tk para el desarrollo de la interfaz. Si no cuenta con estas dos herramientas se puede instalar el rpm que viene en los discos de instalación de Red-Hat estos son de distribución gratuita.
Que el aprendizaje del lenguaje de programación perl/tk con el que se desarrollara la interfaz tome más tiempo del estimado.	El riesgo se puede evitar estudiando constantemente y a fondo el lenguaje para el entendimiento completo de éste.

RIESGO	CONTENCIÓN
Que los documentos no se entreguen en la fecha indicada ocasionando un retraso en el desarrollo del proyecto.	Se cumplirá con la fecha de entrega establecida para cada documento, evitando así el retraso del mismo.
No disponer de una máquina que facilite el desarrollo del proyecto.	Se buscará oportunamente una máquina, este equipo se puede encontrar en un laboratorio, de la escuela o en algún cubículo para desarrollo de aplicaciones.

CAPÍTULO X

REQUERIMIENTOS DEL PROYECTO MDS

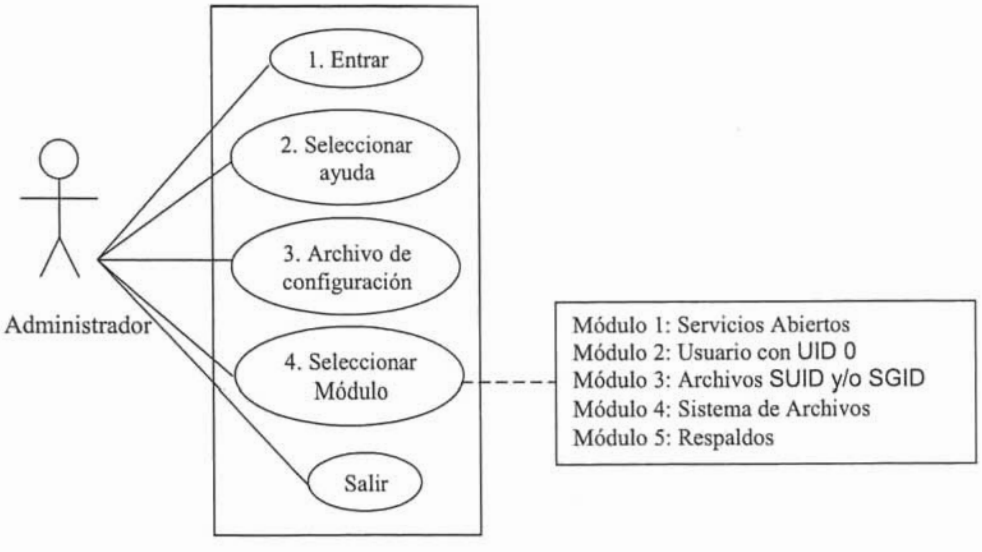
Se necesita una base para el diseño e implementación. Esta base consiste en los requerimientos detallados.

10.1 Requerimientos funcionales

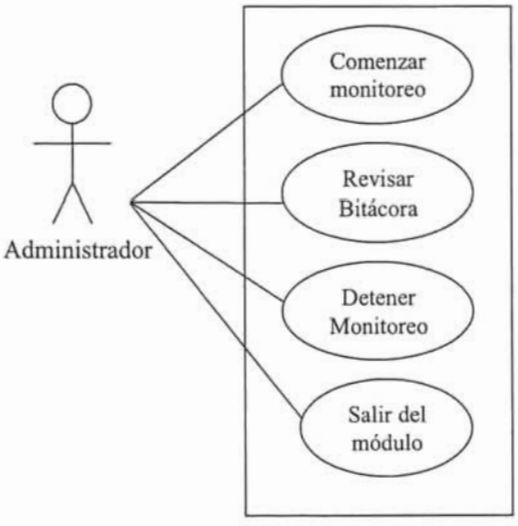
Se requiere que el administrador del sistema monitorea los elementos básicos de un sistema en plataforma tipo LINUX. Cuando exista algún cambio en cualquiera de los módulos esté será reportado por correo electrónico al administrador del sistema, la información que se mandará por correo será leída del archivo de configuración. Este archivo podrá ser modificado por el administrador del sistema.

Se requiere contará con un módulo de ayuda que proporcione información acerca del funcionamiento de cada uno de los módulos.

Caso de uso 1
Entrada al sistema MDS

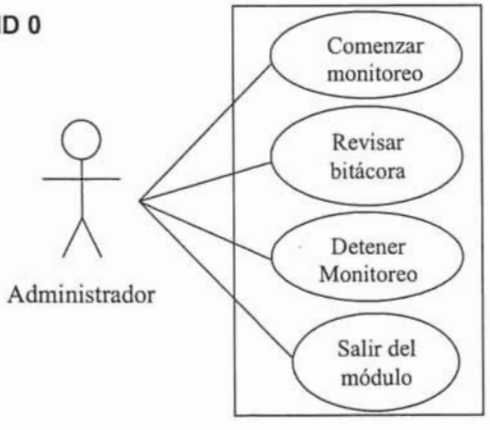


Caso de uso 4.1
Módulo de Servicios Abiertos



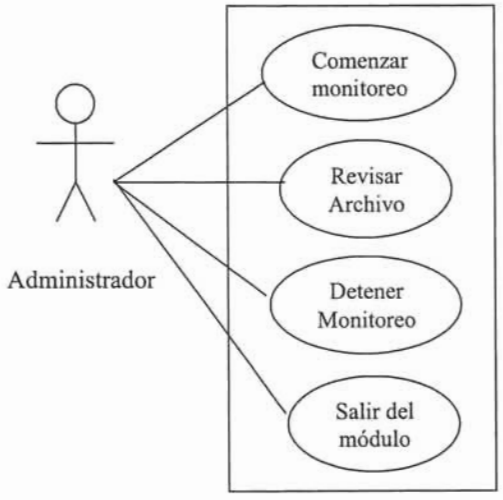
Por su parte el módulo de servicios abiertos, debe reportar cualquier cambio que exista por correo electrónico al administrador del sistema, indicado si un servicio ha sido abierto y registrar el cambio en la bitácora, esta operación debe ser realizada en un minuto.

Caso de uso 4.2
Módulo Usuarios con UID 0



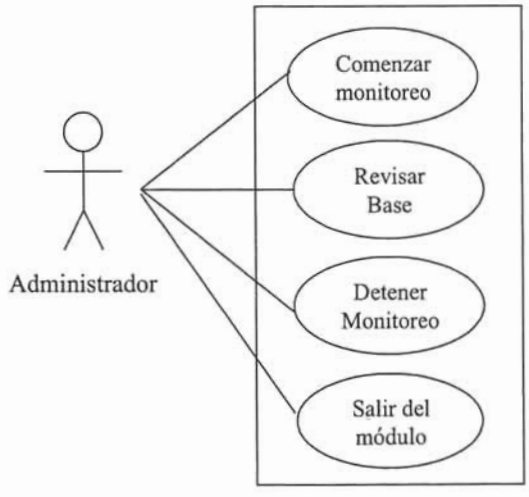
El módulo de usuarios con **UID 0**, debe mantener un control de la tabla **/etc/passwd** donde solamente debe existir un usuario con **UID 0** que es root. En caso contrario, debe enviar un aviso urgente por correo y registrar este cambio en la bitácora. Esta operación debe realizar en menos de un minuto.

Caso de uso 4.3
Módulo Archivos con SUID y/o SGID



Para el módulo archivos con **SUID y/o SGID** debe buscar los archivos que tengan el bit **SUID y/o SGID** y almacenarlos en una base. Cuando exista una variación en éstos notificarlo por correo, esta operación debe ser ejecutada en menos de un minuto.

Caso de uso 4.4
Módulo Sistemas de Archivos

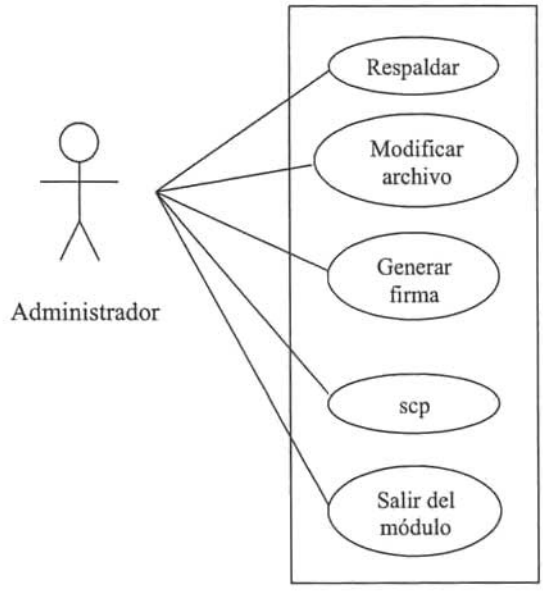


En cuanto al módulo de monitoreo del sistema de archivos, debe controlar las características de los archivos del sistema creando una base con los siguientes elementos:

- Archivos con **SUID y/o SGID**.
- Número de enlaces por archivos.
- Dueño del archivo.
- Grupo del archivo.
- Firma **MD5.**
- Número de i-nodo.
- Permisos.
- Tamaño.

Cuando exista algún cambio en cualquiera de los elementos debe reportarlo por correo y esta operación debe realizarse en menos de un minuto.

Caso de uso 4.5
Módulo de Respaldos



Para el módulo de respaldos, se debe de contar con un archivo de configuración que contendrá lo siguiente.

- Directorios a respaldar.
- El nombre o identificador del usuario con el cual se cifrará el archivo.
- Equipo remoto a donde se enviarán los archivos respaldados y firmados.

Los archivos de respaldados deben ser enviados, a través de un canal seguro **(scp)** al equipo definido en el archivo de configuración.

10.2 Requerimientos no funcionales

Desempeño

- La aplicación está dividida en un módulo maestro y 5 módulos de monitoreo el módulo maestro es el encargado de llamar a los demás módulos y el tiempo de respuesta de cada módulo no debe ser mayor a un minuto.

Confiabilidad y Disponibilidad

- La aplicación no debe tener más de tres fallos en la búsqueda de intrusos en el sistema.

- Esta aplicación debe estar disponible sólo para el administrador del sistema y no permitir el uso a personas sin autorización las cuales podrían hacer un mal uso de ella, ocasionando un daño al sistema.

Manejo de errores

- Los errores generados por la aplicación cuando está sea ejecutada, deben ser redireccionados a un archivo con el nombre de **errores.log** donde serán almacenados.

Requerimientos de interfaz

- La aplicación mandará el correo con el comando mail, para avisar al administrador del sistema de algún cambio ocurrido en algunos de los módulos de monitoreo.

El correo tendrá el siguiente formato.

Asunto: nombre del módulo.
Generado por: nombre del usuario que genero el reporte.
Fecha: fecha en la que se genero el reporte
Nombre de la maquina: nombre del equipo donde se realizó el monitoreo.
Descripción: resumen del cambio encontrado en el momento del monitoreo.

Restricciones

Las restricciones de diseño son:

- La aplicación debe ser desarrollada con el lenguaje de programación perl versión 5.0.
- La interfaz de la aplicación debe ser implementada en perl/tk
- La aplicación debe correr en plataformas tipo UNIX.
- El sistema debe contener una ayuda que será visible en una ventana.
- Los programas deben incluir documentación interna.
- Se debe entregar un documento en latex con la documentación externa del proyecto

10.3 Requerimientos inversos

La aplicación no debe crear archivos con permisos inadecuados, ya que provocarían el acceso al sistema de algún intruso.

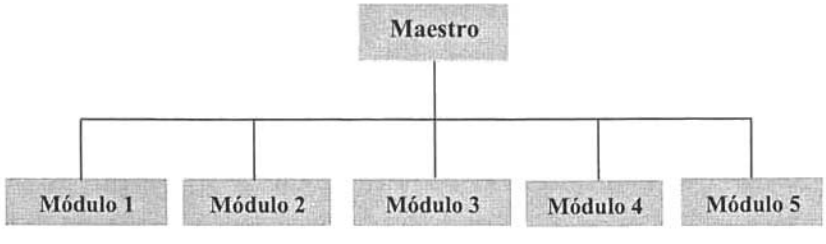
CAPÍTULO XI

ARQUITECTURA DE SOFTWARE DEL PROYECTO MDS

Para un proyecto de desarrollo de software dado pueden existir varias arquitecturas adecuadas para elegir, decidir cuál es la mejor depende de las metas. Suele ser difícil satisfacer todas las metas, ya que un diseño que satisface una puede no satisfacer otra. Para esto se asigna prioridades.

La estructura que se eligió para este proyecto es la siguiente:

Se dividido en módulos, ya que cualquier cambio realizado en alguno de ellos no afectaría a los demás con lo que obtendremos una alta cohesión entre los módulos y baja acoplamiento ya que el cambio sólo afectaría al módulo donde se realizaron los cambio.



Donde Módulo 1: Servicios abiertos.
Módulo 2: Usuarios con UID 0.
Módulo 3: Archivos con SUID y/o SGID.
Módulo 4: Sistema de archivos.
Módulo 5: Respaldos.

Esta arquitectura cumple con las metas descritas en el Capítulo VI .

Extensión

El proyecto cuenta con un módulo maestro que llama a cada uno de los módulo de monitoreo esto permite agregar otro módulo de monitoreo sin modificar el código fuente.
Otro módulo que podría ser agregado es el monitoreo del espacio ocupado en cada uno de los sistemas de archivos. Previniendo así la caída del sistema.

Cambio

En este caso se desea que el diseño permita alteraciones en los requerimientos del proyecto.
Uno de los cambios que se podrían realizar en los requerimientos del proyecto es el siguiente:

Cuando exista algún cambio en los elementos de monitoreo de alguno de los módulos se mostraría un mensaje en la pantalla avisando el cambio del módulo de monitoreo en lugar de enviar el aviso por correo, otro cambio podría ser que el archivo de bitácora pudiera ser especificado en el archivo de configuración por el administrador del sistema.

Sencillez

La sencillez es una meta de diseño en todas las circunstancias. La sencillez de este proyecto se basa en la especificación clara y concreta de cada módulo de monitoreo y del módulo maestro.

Eficiencia

La eficiencia del proyecto, se basará principalmente en que el tiempo de ejecución sea menos de un minuto, esto porque los avisos de alerta, necesitan ser enviados lo antes posible al administrador. Y que esté pueda ser ejecutado en cualquier sistema en plataforma tipo UNIX.

11.1 Selección del lenguaje de programación

El lenguaje seleccionado para el desarrollado del proyecto es perl, la interfaz será desarrollada con el lenguaje de programación perl/tk.

Las razones por las que se eligió el lenguaje de programación perl son las siguientes:

- Es un lenguaje de programación con un conjunto de funciones muy variado.
- Es un lenguaje bastante práctico para generar reportes y extraer información.

Es útil para tareas como:

- Administración de sistemas.
- Acceso a bases de datos.
- Programación de CGI´s para Internet.
- Programación cliente servidor.

Reúne características de otros lenguajes como C, shell, awk, etcétera. Se encuentra disponible en plataformas UNIX, DOS.

Perl/tk se seleccionó porque permite escribir programas para el desarrollo de interfaces graficas de Usuarios (GUI) en plataformas tipo UNIX y es una extensión del lenguaje perl.

CAPÍTULO XII

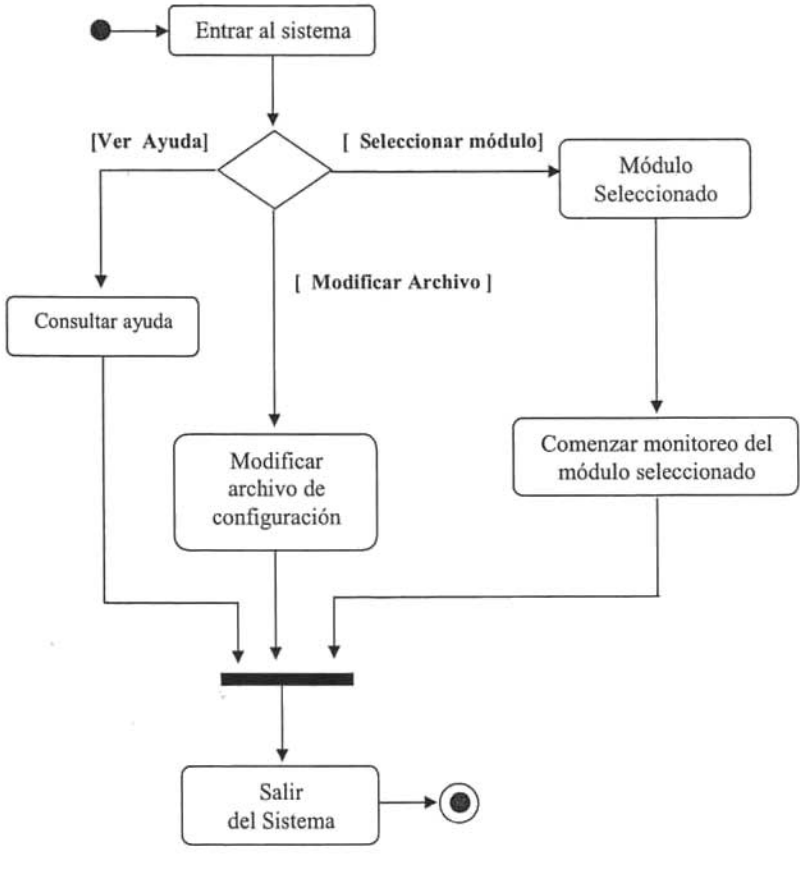
DISEÑO DETALLADO DEL PROYECTO MDS

El diseño detallado será desarrollado con base en la relación entre los casos de uso, los requerimientos y la arquitectura. Esto nos permitirá preparar el proyecto para su implementación.

Para describirlo se usará seudocódigo y diagramas de actividad. A continuación se describe cada uno, tanto sus ventajas como desventajas.

Según lo visto en el Capitulo V, se decidió usar diagramas de actividad, para el diseño detallado escribiendo en la parte inferior de cada diagrama una pequeña descripción de éste.

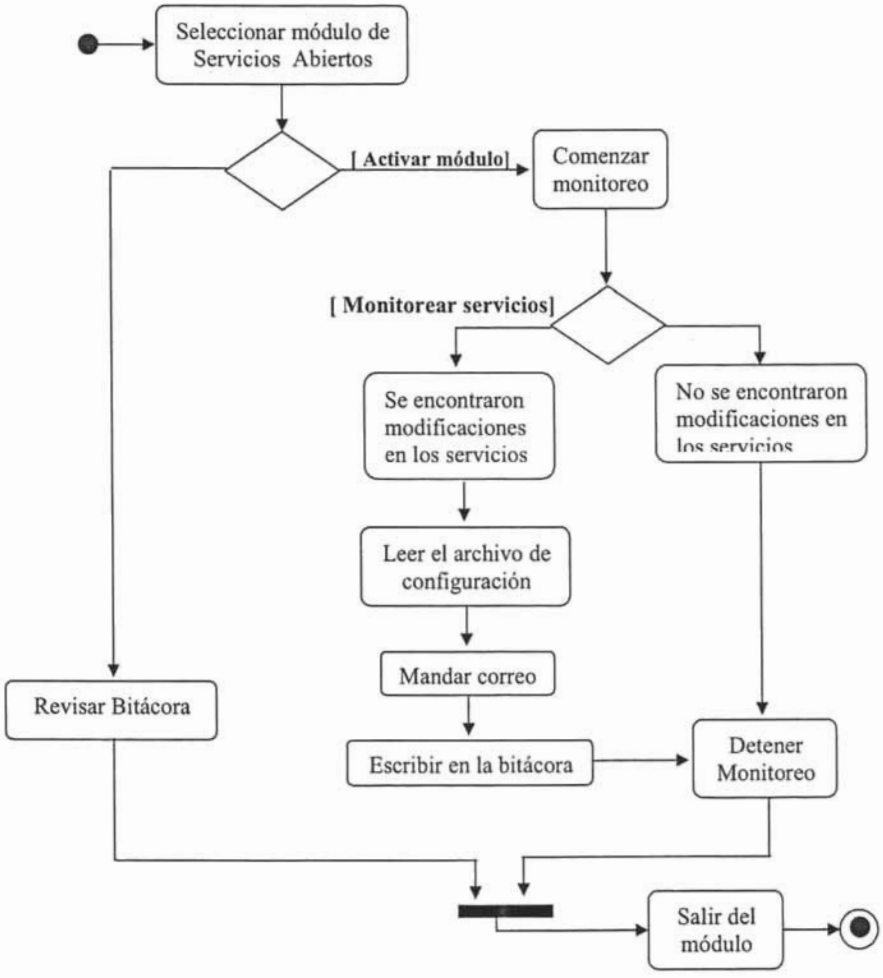
12.1 Diagramas de actividad para el sistema



Si el módulo seleccionado es el de **SERVICIOS ABIERTOS** se llama a la función encargada de crear la ventana para este módulo. Si el botón seleccionado en esta ventana es **MONITOREO** se buscan los puertos que se han abierto o cerrado, si es encontrado algún cambio se obtiene el nombre del equipo, puerto, servicio, fecha y hora en el que se realizó tal cambio.

Esta información será enviada a la dirección electrónica especificada en el archivo de configuración. También el cambio será registrado en la bitácora. Después de comenzar el monitoreo, si el botón seleccionado es **DETENER** se cancelará la revisión de los puertos. Por el contrario si el botón seleccionado es **BITACORA** se muestra una ventana con la información del monitoreo realizado.

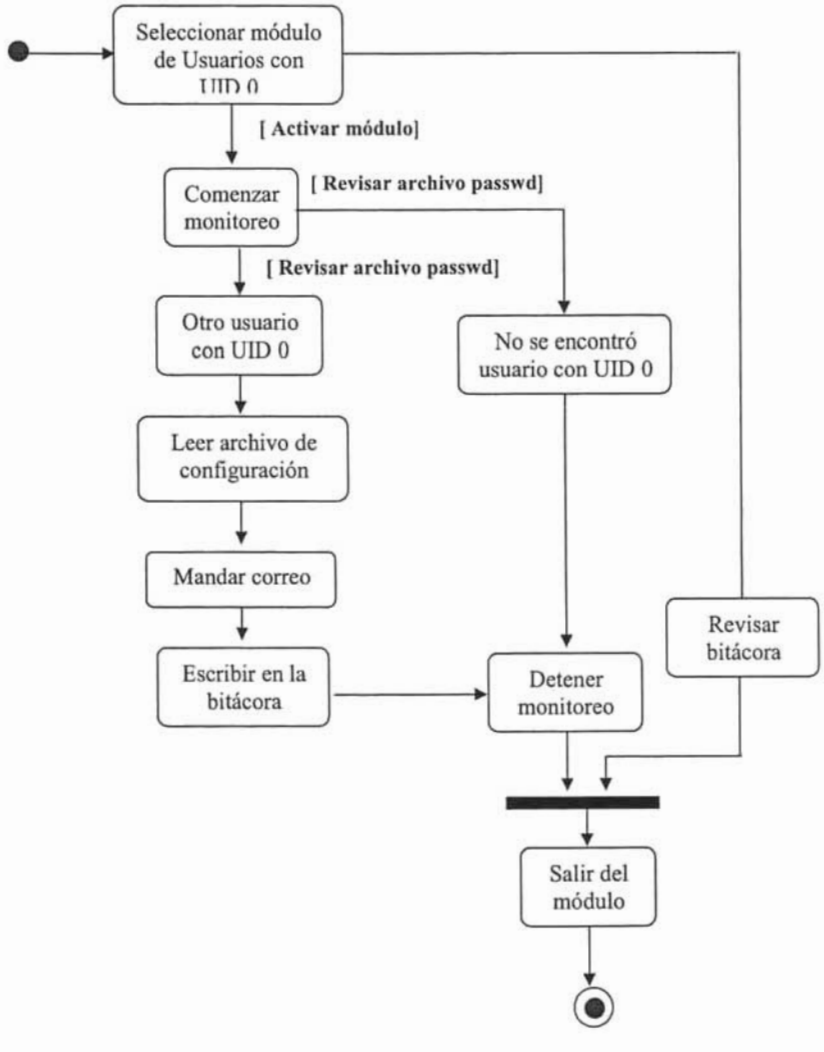
Módulo Servicios Abiertos



Si el módulo seleccionado es **usuario UID 0** se mostrara la ventana para ese módulo. Si en este módulo es seleccionada el botón **MONITOREO**, verifica que no exista otro usuario con UID 0 mas que root, en caso contrario mandara por correo, la linea en la que el campo UID sea 0 y leerá del archivo de configuración la dirección electrónica a la que se mandará el reporte.

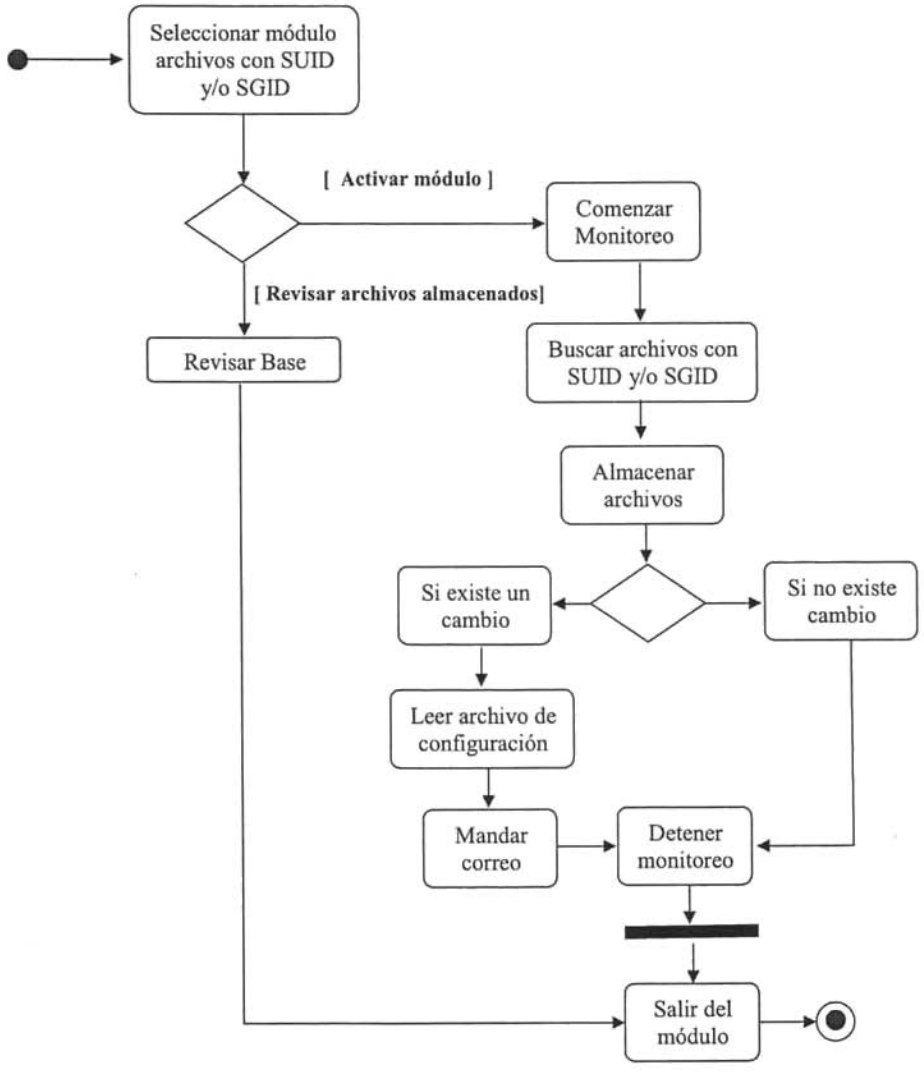
Si no se encuentra ningún cambio y es seleccionada la opción de **BITACORA** se mostrará el monitoreo realizado. Si selecciona la opción de **DETENER** se detendrá el monitorear de este módulo.

Módulo Usuarios con UID 0



Cuando el módulo seleccionado sea **ARCHIVOS SUID Y/O SGID** se mostrará la ventana de este módulo. Si el botón seleccionado es **MONITOREO**, se realizara una búsqueda de los archivos que tengan el bit **SUID Y/O SGID** si existe algún cambio en ellos se mandará un correo al administrador notificándole este cambio y se creará un registro con estos archivos. Cuando el botón **DETENER** sea seleccionado, se suspenderá la búsqueda de estos archivos. Si la opción seleccionada es **BASE**, se mostrarán los archivos en los que se encontró un cambio.

Módulo: Archivos con SUID y/o SGID



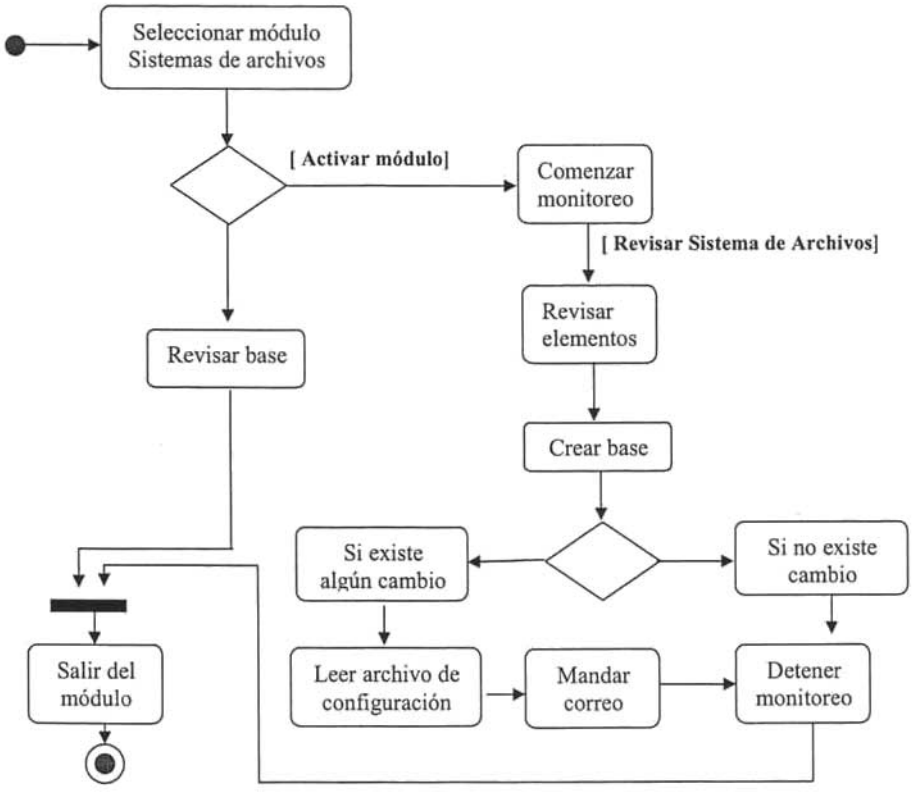
Si el módulo seleccionado es **SISTEMA DE ARCHIVOS** será mostrada una ventana con las opciones correspondientes para este módulo.

Cuando el botón **MONITOREO** sea seleccionado, se revisará en el sistema de archivos los siguientes elementos:

- Archivos con SUID y/o SGID.
- Número de enlaces por archivos.
- Dueño del archivo.
- Grupo del archivo.
- Firma MD5.
- Número de i-nodo.
- Permisos.
- Tamaño.

Si existe algún cambio en alguno de estos elementos, se notificará el cambio por correo y se creará una base con todos estos elementos. Si después el botón seleccionado es **DETENER**, la revisión de los elementos dejará de efectuarse. Cuando la opción sea **BASE**, se mostrará los elementos encontrados.

Módulo Sistemas de Archivo

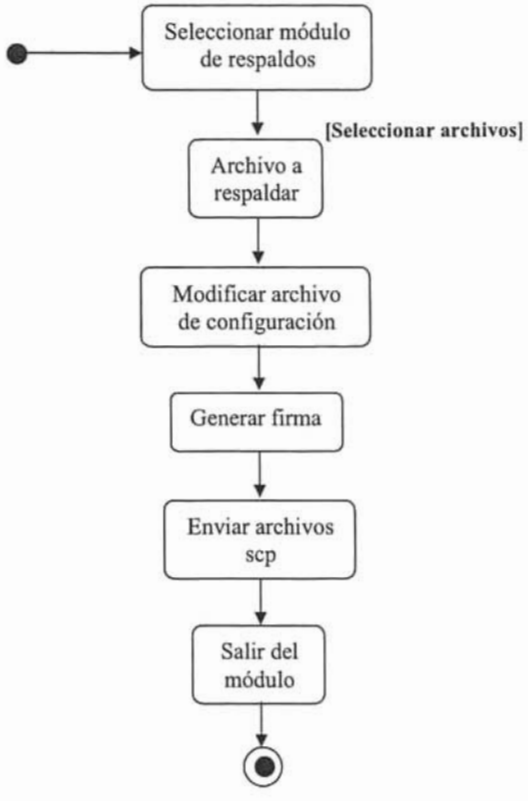


Cuando el módulo seleccionado sea el de **respaldos** se mostrará la ventana de este módulo.
Si la opción seleccionada es **RESPALDAR** se mostrará una ventana, donde se solicitara al administrador escribir la ruta del archivo a respaldar.

Si la opción seleccionada no fue esa si no la de **ARCHIVO DE CONFIGURACION**, se abrirá una ventana desde la cual se podrá realizar modificaciones a este archivo. Si el botón seleccionado fue el de **MD5**, entonces se firmarán los archivos.

Cuando la opción seleccionada es **SCP**, se enviarán los archivos a la dirección de la máquina que está especificada en el archivo de configuración.

Módulo de respaldos



CAPÍTULO XIII

IMPLEMENTACIÓN DEL PROYECTO MDS

La meta de la implementación del sistema es basarnos en el diseño detallado, y hacer una programación correcta, para promover su mantenimiento. Para esto aplicaremos convenciones de nombres, e inspecciones al código para clasificar la severidad de los defectos.

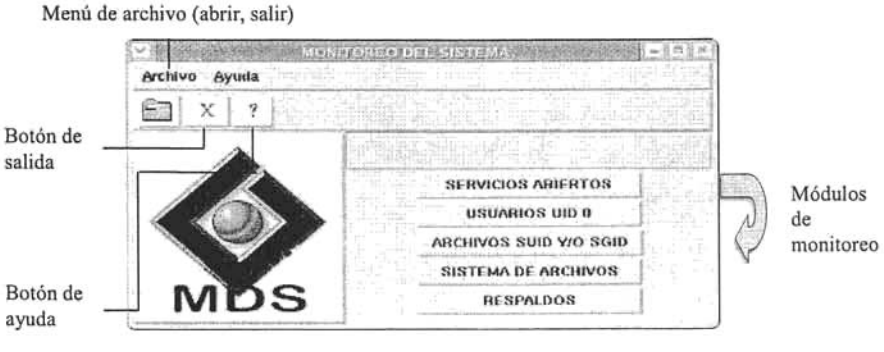
A continuación se listan dichas convenciones:

- Los nombres de las funciones iniciarán con la primera letra en mayúscula como en Ayuda.
- Cuando un botón sea creado, su nombre comenzará con un B mayúscula como en B_cerrar.
- El nombre para las ventanas, comenzará con minúscula y terminará con un v por ejemplo v_principal.
- Los frames comenzarán con un F mayúscula al principio del nombre como en F_principal.
- Las imágenes comenzarán con el prefijo img como en img_ciencias.
- Las etiquetas comenzarán con una E mayúscula al principio de su nombre como en E_sistema.
- El nombre de las variables comenzarán con una I y terminarán en minúscula como en I_fecha.
- Cuando sea creado un Canvas su nombre iniciará con un C por ejemplo C_datos.

13.1 Inspección del código

Módulo maestro

La ventana del módulo maestro que se muestra en la figura anterior cuenta con una barra de menú en está, se encuentran las opciones de **Archivo** que contiene las opciones siguientes (Abrir, Salir) y de Ayuda que contiene lo siguiente (Contenido, Acerca de).



Debajo de esta barra de menú se tiene la barra de herramientas con los botones de editor, salir y ayuda con sus respectivos iconos en cada botón.

Cuando el puntero es posicionado sobre alguno de los botones aparece un letrero amarrillo indicándole al usuario el nombre del botón

Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección del código	5 minutos

Descripción: El nombre de la ventana no cumplía con la convención de nombres.
Severidad: Media

22/06/04	2	interfaz	implementación	inspección del código	3 minutos

Descripción: El nombre del botón no cumplía con la convención de nombres.
Severidad: Media

22/06/04	3	interfaz	implementación	inspección del código	4 minutos

Descripción: No se estaba llamando a la función correcta en el botón de servicios abiertos.
Severidad: Media

Módulo de servicios abiertos

La ventana de servicios abiertos esta conformada por la imagen a la izquierda y cuatro botones a la derecha con las opciones de monitoreo, detener y bitácora. Se separo el botón de salida con una etique, para que le diera una mejor presentación. La implementación de los otros módulos puede ser consultada en el anexo A.

Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	3 minutos

Descripción: El nombre de la función para el módulo 1 no cumplía con la convención de nombres.
Severidad: Media

22/06/04	2	ejecución	implementación	inspección de código	8 minutos

Descripción: La función del modulo de servicios abiertos obtenía los campos requeridos del comando nmap.
Severidad: Media

22/06/04	3	ejecución	implementación	inspección de código	8 minutos

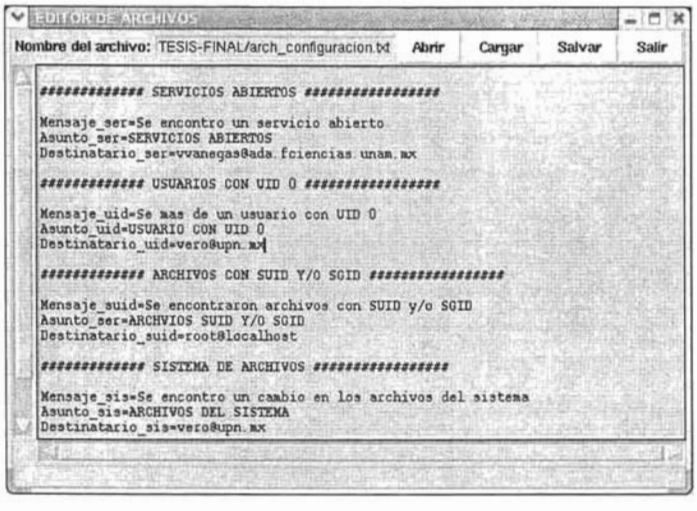
Descripción: El archivo donde se guarda la información de los servicios abiertos no es renombrado por lo que la aplicación manda el correo mas de una vez.
Severidad: Alta

22/06/04	4	interfaz	implementación	inspección de código	3 minutos

Descripción: El nombre del botón de bitácora no cumple con la convención de nombres.
Severidad: Media

Editor de archivos

La ventana del editor de archivos tiene una caja de texto, en la que se puede escribir el nombre del archivo a editar, si este archivo no existe se muestra un mensaje al usuario, indicándole que dicho archivo no existe.



Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	15 minutos

Descripción: La función de guardar_arch no escribía en el archivo, la información que era escrita en el editor de archivos.
Severidad: Alta

22/06/04	2	ejecución	implementación	inspección de código	20 minutos

Descripción: No se contaba con un función que muestre un mensaje al usuario cuando un archivo no existía.
Severidad: Alta

22/06/04	3	ejecución	implementación	inspección de código	5 minutos

Descripción: Las variables no cumplen con la convención de nombres.
Severidad: Alta

Bitácora

En la ventana de bitácora se muestra la información del monitoreo realizado en dicho módulo.

Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	3 minutos

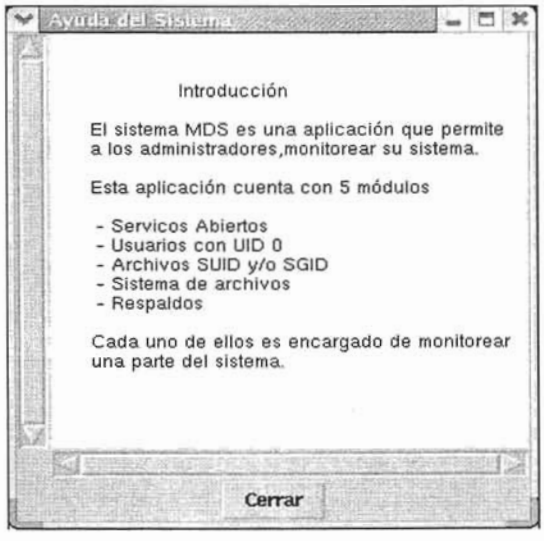
Descripción: El nombre de la variable **text** no cumple con la convención de nombres.
Severidad: Media

22/06/04	2	ejecución	implementación	inspección de código	3 minutos

Descripción: El botón de salir no cumple con la convención de nombres.
Severidad: Media

Ayuda

En la venta de ayuda se muestra la información acerca del sistema del sistema.



Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	3 minutos

Descripción: El nombre de la función Ayuda no cumplía con la convención de nombres ya que comenzaba con minúscula.
Severidad: Media

22/06/04	2	ejecución	implementación	inspección de código	3 minutos

Descripción: El botón de salir no cumple con la convención de nombre.
Severidad: Media

22/06/04	3	ejecución	implementación	inspección de código	2 minutos

Descripción: El nombre para la imagen no cumple con la convención de nombres.
Severidad: Media

Acerca de MDS

En la ventana de acerca de MDS tiene el logotipo de la aplicación y dos botones, el botón de créditos, cuando esté es seleccionado despliega otra ventana, que muestra el nombre del autor y el correo electrónico.

Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	2 minutos

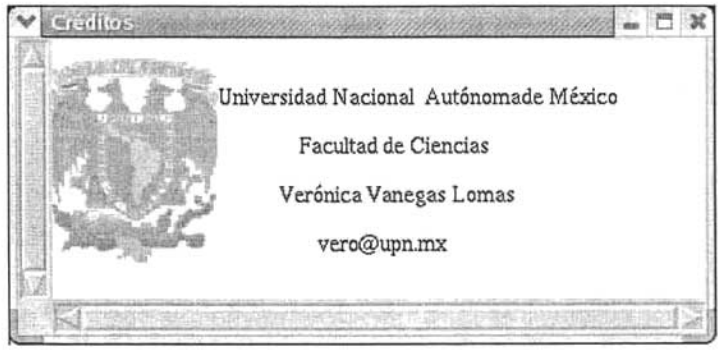
Descripción: El nombre del botón de créditos no cumplía con la convención de nombres.
Severidad: Media

22/06/04	2	ejecución	implementación	inspección de código	3 minutos

Descripción: El botón de cerrar no cumple con la convención de nombres.
Severidad: Media

Créditos

En la ventana de créditos, se muestra la información del autor.



Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	2 minutos

Descripción: El nombre de la imagen no cumple con la convención de nombres.
Severidad: Media

22/06/04	2	ejecución	implementación	inspección de código	3 minutos

Descripción: El nombre de la venta no cumple con la convención de nombres.
Severidad: Media

CAPÍTULO XIV

PRUEBAS DEL PROYECTO MDS

La meta para la prueba del sistema es encontrar el mayor número de defectos. Con esto podremos tener un ahorro de tiempo, si se detectan antes de la integración.

Estas pruebas se llevarán a cabo siguiendo el método de la caja negra, que consiste en ver si una aplicación o parte de ella proporciona la salida adecuada, al usar una entrada apropiada.

Para esto realizaremos los siguientes pasos:

1. Cada uno de los módulos serán probados por el programador.
2. La documentación de la prueba debe mencionar el nombre del módulo al que se le hizo la prueba, los datos de entrada y salida si los hay.
3. La parte que más será probada es la del monitoreo, esta se realizará hasta encontrar las 3 primeras fallas y las demás solo se realizarán una vez.
4. Los datos de entrada para las pruebas en los casos que sean necesarios serán proporcionados por el propio programador.
5. Se registrara el tiempo dedicado a las pruebas, así como el número de defectos, su tipo y fuente.
   Estos nos ayudarán a evaluar el estado en que se encuentra el sistema.

14.1 Prueba aplicada a: Módulo maestro

	Si	No
1.1 Salir del sistema desde el botón de acceso directo	pasa	
1.2 Salir del sistema desde la opción del menú desplegable	pasa	
1.3 Abrir el editor de archivos desde el botón de acceso directo	pasa	
1.4 Abrir el editor de archivos desde la opción del menú desplegable	pasa	
1.5 Mostrar la ayuda desde la opción del menú desplegable.	pasa	
1.6 Mostrar la ayuda desde el botón de acceso directo.	pasa	
1.7 Mostrar la información del sistema desde la opción del menú desplegable.	pasa	
1.8 Abrir la ventana con la información de los créditos del Desarrollador del sistema.	pasa	

Num. de defectos	Tipo	Fuente	Tiempo
0	----	-----	5 minutos

14.2 Prueba aplicada a: Servicios abiertos

	Si	No
2.1 Obtiene el puerto y el servicio.	pasa	
2.2 El archivo de bitácora tiene los permisos adecuados.		no pasa
2.3 Manda el correo solo una vez.		no pasa
2.4 Lee los datos correctos del archivo de configuración.	pasa	
2.5 Detiene el monitoreo cuando se le indica.		no pasa

Num. de defectos	Tipo	Fuente	Tiempo
3	alta	implementación	5 minutos

14.3 Prueba aplicada a: Usuarios con UID 0

	Si	No
3.1 Muestra la información de la bitácora	pasa	
3.2 Manda el correo a la dirección electrónica correcta.	pasa	
3.3 Detiene el monitoreo cuando se le indica	pasa	
3.4 Los permisos del archivo de bitácora son correctos		no pasa
3.5 El botón de salir cierra el módulo	pasa	

Num. de defectos	Tipo	Fuente	Tiempo
1	alto	implementación	5 minutos

14.4 Prueba aplicada a: Archivos con SUID y/o SGID

	Si	No
4.1 Crea la base con los archivos encontrados.	pasa	
4.2 Manda el correo a la dirección electrónica correcta.	pasa	
4.3 Muestra la información de la base cuando es solicitada.	pasa	
4.4 Se detiene el monitoreo cuando es solicitado.	pasa	
4.5 Los permisos del archivo de la base son los correctos.		no pasa
4.6 El botón de salir cierra el módulo	pasa	

Num. de defectos	Tipo	Fuente	Tiempo
1	alto	requerimientos	5 minutos

14.5 Prueba aplica a: Sistema de archivos

	Si	No
5.1 Muestra la información de la base	pasa	
5.2 Se almacenan todos los elementos en la base	pasa	
5.3 Se detiene el monitoreo cuando se indica	pasa	
5.4 El archivo de la base tiene los permisos adecuados	pasa	
5.5 El botón de salir cierra el módulo	pasa	

.

Num. de defectos	Tipo	Fuente	Tiempo
0	----	-----	5 minutos

14.6 Prueba aplicada a: Respaldos

	Si	No
6.1 El respaldo de los archivo se hace de manera correcta.	pasa	
6.2 La firma de los archivos se realiza de manera adecuada.	pasa	
6.3 Los archivos respaldados se envían a la maquina indicada	pasa	
6.4 Se detiene el monitoreo cuando se le indica		no pasa
6.5 Lee la información del archivo de configuración.	pasa	
7.6 Los archivos y la firma se envían en menos de 3 minutos.		no pasa

Num. de defectos	Tipo	Fuente	Tiempo
2	alta	implementación	5 minutos

CAPÍTULO XV

INTEGRACIÓN DEL PROYECTO MDS

Durante el proceso de integración, se añadió al código de cada botón creado en el módulo maestro la función que se encarga de llamar a su módulo respectivo.

Las pruebas realizadas en esta etapa fueron de tipo caja negra, probando contra requerimientos no funcionales.

15.1 Pruebas del sistema

1. La aplicación no debe permitir, que un usuario no autorizado pueda utilizar este sistema.
2. La respuesta de cada módulo no debe ser mayor a un minuto.
3. La aplicación debe crear un archivo llamado **errores.log**, donde serán almacenados los errores generados por la aplicación.
4. El correo debe ser enviado con el comando mail, y debe cumplir con el siguiente formato.

Asunto: nombre del módulo.
Generado por: nombre del usuario que genero el reporte.
Fecha: Fecha en la que se genero el reporte
Nombre de la maquina: nombre del equipo donde se realizo el monitoreo.
Descripción: resumen del cambio efectuado en el módulo.

Prueba	Resultado	Referencia
1	Pasa	requerimientos
2	Pasa	requerimientos
3	no pasa	requerimientos
4	Pasa	requerimientos

Las pruebas realizadas en la integración fueron mínimas ya que en la etapa de pruebas todos los defectos encontrados en cada uno de los módulos fueron corregidos.

CONCLUSIONES

En años recientes, se ha desarrollado una gran variedad de herramientas con el único objeto de ayudar a los administradores del sistema a proteger sus computadoras que cuentan con el sistema operativo tipo UNIX.

Algunas de estas herramientas son Tripwire que se encarga de comprobar la integridad de archivos y directorios ayuda a los administradores a monitorear alguna posible modificación en algunos de los archivos, esta herramienta es de distribución libre, esta disponible para Linux, Debian y Windows. Otra herramienta es Snort es un sistema de detección de intrusos de red, capaz de realizar análisis de trafico en tiempo real y registro de paquetes en redes con IP, es una herramienta de distribución libre y esta disponible para Linux, Debian y Windows.

Así como estas herramientas hay muchas en el mercado que nos proporcionan una ayuda para el monitoreo de nuestro sistema, pero no encontré una que monitoreara los elementos básicos de un sistema, esto fue lo que me hizo pensar en el desarrollo de un sistema que se encargara de está tarea.

El desarrollo del sistema de monitoreo, basado en la ingeniería de software, me ayudó a obtener un software de calidad. La etapa más importante fue la de requerimientos ya que la recopilación correcta de estos es un proceso difícil ya que los artefactos generados en esta fase del proceso de desarrollo de software son dedicados en esencia a los clientes de la aplicación, pero también son de interés para los ingenieros de software que construyen o mantienen el software.

La interfaz para este sistema jugó un papel importante ya que un software no podría ser competitivo si no cuenta con ella.

Perl/tk es un potente lenguaje de programación para la realización de interfaces gráficas y con ayuda de perl, la programación es más sencilla.

Es importante destacar que a lo largo del desarrollo del sistema, se obtuvieron nuevos conocimientos, así como se reafirmaron los que ya se tenían.

Con las pruebas que se han realizado al sistema de monitoreo se puede ver que se ha cumplido con el objetivo planteado.

ANEXO A

Módulo usuarios con UID 0

Registro de defectos

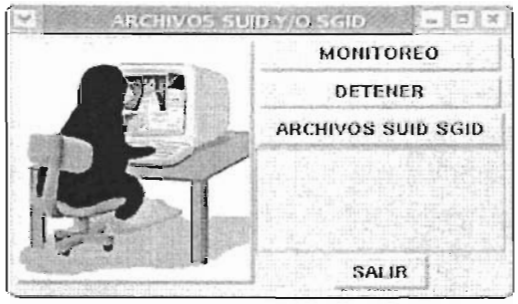
Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	3 minutos

Descripción: El nombre de la imagen para el módulo 2 no cumplía con la convención del nombre para imágenes.
Severidad: Media

22/06/04	2	ejecución	implementación	inspección de código	2 minutos

Descripción: El nombre de la variable nombre_maq no cumplía con la convención de nombres.
Severidad: Media

Módulo archivos con SUID y/o SGID



Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	5 minutos

Descripción: La sintaxis del comando **cut** estaba mal escrita por lo tanto no obtenía el campo deseado.
Severidad: Alta

22/06/04	2	ejecución	implementación	inspección de código	10 minutos

Descripción: El nombre del descriptor de archivo no era el correcto. Por lo tanto no escribía en el archivo de bitácora.
Severidad: Alta

22/06/04	3	ejecución	implementación	inspección de código	8 minutos

Descripción: El nombre de la variable de comparación del **while** no es el correcto por lo que nunca entra en este ciclo.
Severidad: Alta

Módulo sistema de archivos

Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	3 minutos

Descripción: El nombre del botón para salir del modulo 4 no cumple con la convención de nombres.
Severidad: Media

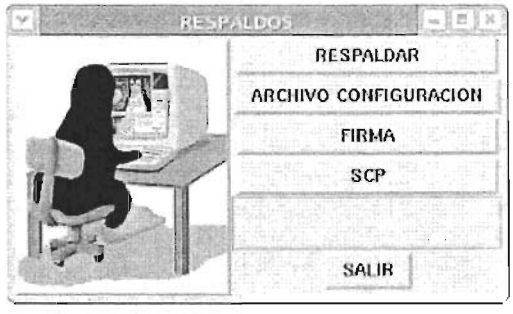
22/06/04	2	ejecución	implementación	inspección de código	8 minutos

Descripción: En el botón de monitoreo no se manda a llamar a la función correspondiente.
Severidad: Alta

22/06/04	3	ejecución	implementación	inspección de código	3 minutos

Descripción: El botón de base no cumple con la convención de nombres
Severidad: Media

Módulo de Respaldos



Registro de defectos

Fecha	Número	Tipo	Etapa incluida	Etapa eliminada	Tiempo de reparación
22/06/04	1	interfaz	implementación	inspección de código	3 minutos

Descripción: El botón de firma no cumple con la convención de nombres.
Severidad: Media

22/06/04 2 ejecución implementación inspección de código 3 minutos

Descripción: El nombre de la etiqueta no cumple con la convención de nombres.
Severidad: Media

22/06/04 3 interfaz implementación inspección de código 15 minutos

Descripción: La función para mandar los archivos con scp no está obteniendo del archivo de configuración el nombre de la máquina a donde se va a transferir el archivo.
Severidad: Alta

22/06/04 4 ejecución implementación inspección de código 10 minutos

Descripción: La sintaxis del comando **tar** está mal, por lo que no hace bien los respaldos.
Severidad: Alta

ANEXO B

```
#!/usr/bin/perl
#Monitoreo de sistemas
# Programa: menu2.txt
#Programo: Vanegas Lomas Veronica
#Fecha: 9-julio-2004
#Los # son comentarios

use Tk;
use Tk::Balloon;
use Tk::Dialog;

$Ihost_name=`hostname`;
$Inombre_maq = `hostname`;
$Iusuario=`whoami`;
$Ifecha=`date`;

#Crea la ventana principal
my $principalv = MainWindow->new();

#Se crea el frame en la parte superior de la ventana.
$principalv->configure(-title => "MONITOREO DEL SISTEMA");
$Fmenu = $principalv->Frame(-relief      => "raised",
                      -height      => '6c',
                      -width       => '6c',
                      -borderwidth =>  2);

$Barchivo = $Fmenu->Menubutton(-text      =>"Archivo",
                        -underline => 0);

$menu = $Barchivo->Menu();
$Barchivo->configure(-menu  => $menu);
$menu->command(-command     => \&Editor_c,
             -label       => "Abrir",
             -accelerator => "Ctrl-A",
             -underline   => 0);

#Se coloca un separador entre los botones.
$menu->separator;
$menu->command(-label       => "Salir",
             -command     => \&Salir,
             -accelerator => "Ctrl-S",
             -underline => 0);

$principalv->bind('<Control-Key-s>' => \&Salir);
$principalv->bind('<Control-Key-a>' => \&Editor_c);

$Bayuda = $Fmenu->Menubutton(-text      =>"Ayuda",
                       -underline => 0);
```

```
$ayuda = $Bayuda->Menu();
$ayuda->command(-command      => \&ayuda,
            -label        => "Contenido",
            -accelerator  => "Ctrl-C",
            -underline    => 0);

$ayuda->separator;
$ayuda->command(-command     => \&Acerca_de,
            -label       => "Acerca de",
            -accelerator => "Ctrl-D",
            -underline   => 0);

#Acceso a las opciones con teclas
$principalv->bind('<Control-Key-c>' => \&ayuda);
$principalv->bind('<Control-Key-d>' => \&Acerca_de);

$Bayuda->configure(-menu=>$ayuda);
$Barchivo->pack(-side=>"left");

$Bayuda->pack(-side=>"left");
$Fmenu->pack(-side => "top",
           -fill => "x");

#Se crea el segundo frame para los botones de acceso directo
$Fbarra = $principalv->Frame(-relief      => "raised",
                       -height      => '6c',
                       -width       => '6c',
                       -borderwidth => 2);

#Boton para el editor
$Babrir=$Fbarra->Button(-command    =>\&Editor_c,
                  -background =>'white');

$img_aceptar=$Babrir->Photo(-file => 'accept.gif');
$Babrir->configure(image=>$img_aceptar);
$Babrir->pack(-side => 'left');
$bolloon3=$Fbarra->Balloon(-background =>'yellow');
$bolloon3->attach($Babrir, -balloonmsg =>"Editor");

#Boton para salir.
$Bsalir_s=$Fbarra->Button(-text       => 'X',
                      -foreground     => 'red',
                      -font           => "{Times New Roman} 16 {normal}"
,
                      -background     => 'white',
                      -command        => \&Salir );

$Bsalir_s->pack(-side => 'left');
$bolloon=$Fbarra->Balloon(-background =>'yellow');
$bolloon->attach($Bsalir_s,
                  -balloonmsg =>"Salir");
```

```
#Boton de ayuda
$ayuda=$Fbarra->Button(-text       => '?',
                       -foreground  => 'blue',
                       -font        => "{Times New Roman} 16 {normal}",
                       -background  => 'white',
                       -command     => \&ayuda );


$ayuda->pack(-side  => 'left');
$bolloon2=$Fbarra->Balloon(-background =>'yellow');
$bolloon2->attach($ayuda, -balloonmsg =>"Ayuda");
$Fbarra->pack(-side =>"top",
              -fill =>"x");

$Eicono=$principalv->Button();
$img_logo =$Eicono->Photo(-file => 'mds.gif');
$Eicono->configure(image=>$img_logo);
$Eicono->pack(-side => 'left');

$Estatus = $principalv->Label(-relief      => groove,
                              -borderwidth => 2,
                              -anchor      => "w",
                              -padx        => 150,
                              -pady        => 10);

$Estatus->pack(-side =>"top",
               -fill =>"x");

#Boton del primer modulo
my $Bmodulo1 = $principalv->Button(-text    => 'SERVICIOS ABIERTOS',
                                   -command => \&Modulo1);
$Bmodulo1->configure(-width => 23);
$Bmodulo1->pack(-side => 'top',
                -fill => 'y');

#Boton para el segundo modulo
my $Bmodulo2 = $principalv->Button(-text     => 'USUARIOS UID 0',
                                   -command  => \&Modulo2);
$Bmodulo2->configure(-width => 23);
$Bmodulo2->pack(-side => 'top');

#Boton para el tercer modulo
my $Bmodulo3 = $principalv->Button(-text     => 'ARCHIVOS SUID Y/O SGID',
                                   -command  => \&Modulo3);
$Bmodulo3->pack(-side => 'top');

#Boton para el cuarto modulo
my $Bmodulo4 = $principalv->Button(-text    => 'SISTEMA DE ARCHIVOS',
                                   -command  => \&modulo4);

$Bmodulo4->configure(-width => 23);
$Bmodulo4->pack(side => 'top');
```

```
#Boton para el quinto modulo
my $Bmodulo5 = $principalv->Button( -text    => 'RESPALDOS',
                                    -command => \&modulo5);
$Bmodulo5->configure(-width => 23);
$Bmodulo5->pack(-side => 'top');

MainLoop;

sub Modulo1{

   `nmap -sT localhost|grep open >nmap1`;

    my $modulo1_v = MainWindow->new();
    $modulo1_v->configure(-title => 'SERVICIOS ABIERTOS');
    $E_imagen_m1=$modulo1_v->Button();
    $img_servicios =$E_imagen_m1->Photo(-file => 'wstux.gif');


    $E_imagen_m1->configure(image=>$img_servicio);
    $E_imagen_m1->pack(-side => 'left');
    $E_separador = $modulo1_v->Label(-text        => "",
                                     -relief      => groove,
                                     -borderwidth => 2,
                                     -anchor      => "w",
                                     -padx        =>  5,
                                     -pady        => 20);

    $E_separador->pack(-side => "top",
                       -fill => "x");

    my $Bmonitoreo_ser= $modulo1_v->Button(-text    => 'MONITOREO',
                                           -command  => \&monitorearSA);

    $Bmonitoreo_ser->pack(-side => 'top');
    my $Bdetener=$modulo1_v->Button(-text    => 'DETENER',
                                    -command => \&Detener_monitoreo);

    $Bdetener->configure(-width => 11);
    $Bdetener->pack(-side=>'top');

    my $Bbitacora= $modulo1_v->Button(-text    => 'BITACORA',
                                      -command => \&Abrir_bit);

    $Bbitacora->configure(-width =>11);
    $Bbitacora->pack(-side => 'top' );

    $E_separador2 = $Window->Label(-text        => "",
                                   -relief      => groove,
                                   -borderwidth => 2,
                                   -anchor      => "w",
                                   -padx        => 5,
                                   -pady        => 0);

   $E_separador2->pack(-side => "top",
                       -fill => "x");
```

```
   my $Bsalir = $modulo1_v->Button(-text    => 'SALIR',
                                   -command => sub{ $modulo1_v->destroy;});

    $Bsalir->pack(-side => 'top');


#Funcion que se encarga de  monitorear los puertos abiertos
sub monitorearSA{

       open(PUERTO,">>BITACORA");

       foreach(1..4){

         `nmap -sT localhost|grep open >nmap2`;
         `diff nmap1 nmap2>resul`;
         `more resul`;



         $lineas=`cat resul|grep open| wc -l `;

         $salir="no";

         while ($salir eq "no")
         {

             if ($lineas == '1'){

               print "Estas son las lineas$lineas\n";
               $puerto=`cat resul |tail -1|tr -s " " ":"|cut -d: -f2`;
               $ser=`cat resul |tail -1|tr -s " " ":"|cut -d: -f4`;

               open (CORREOS, "|mail -s servicios vero\@upn.mx");

               print CORREOS "================================== ";
               print CORREOS "\nREPORTE DE MONITOREO DE SERVICIOS
ABIERTOS";
               print CORREOS "\n====================================";
               print CORREOS "\n\nReporte generado por:\t ";
               print CORREOS "$usuario";
               print CORREOS "\Reporte creado el:\t";
               print CORREOS "$fecha\n";
               print CORREOS "-------------------------------------- ";
               print CORREOS "\n\nNombre del host:\t$nombre_maq";
               print CORREOS "\n\nSe encontro el puerto";
               print CORREOS " ";
               print CORREOS $puerto;
               print CORREOS " ";
               print CORREOS "con el servicio $ser\n";
               print  PUERTO "Puerto abierto:$puerto servicio:$ser\n";

             `cp nmap2 nmap1`;

             }
             else
             {
```

```
            print PUERTO "\n\nNombre del host:$host_name";
            print PUERTO "\nNo se encontro ningun puerto abierto\n";
            close(PUERTO);
          }

          $salir="si";
        }

        print "$salir\n";
        print "Espera terminada\n";

      }
      close(CORREOS);

  }



    sub Abrir_bit
    {
    my $bitacora_v = MainWindow->new();
    $bitacora_v->configure(-title => 'BITACORA');
    $maintext = $bitacora_v->Scrolled( 'Text' );
    open(SOURCE, "BITACORA") or die "no se puede abrir";
    tie(*TEXT, 'Tk::Text', $maintext);
    print TEXT <SOURCE>;
    close(SOURCE);
    $maintext->pack();
     my $salirb_bit = $bitacora_v->Button(-text    => CERRAR,
                                          -command => sub { $bitacora_v-
>destroy;});
    $salirb_bit->pack (-side => 'top' );
    }

}
sub Modulo2{

    my $modulo2_v = MainWindow->new();
    $modulo2_v->configure(-title => 'USUARIOS CON UID 0',
                          -width => 30);

    $E_m2=$modulo2_v->Button();
    $img_m2 =$E_m2->Photo(-file => 'usuarios.gif');
    $E_m2->configure(image=>$img_m2);
    $E_m2->pack(-side => 'top');

    my $Bmonitoreo=$modulo2_v ->Button(-text   => 'MONITEOREO',
                                       -command => \&Monitoreo_m2);
    $Bmonitoreo->configure(-width => 12);
    $Bmonitoreo->pack(-side => 'top');

    my $Bdetener=$modulo2_v ->Button(-text => 'BITACORA',
                                     -command => \&bitacora_m2);
    $Bdetener->configure(-width => 12);
    $Bdetener->pack(-side => 'top');
```

```
    $E_separa = $modulo2_v->Label(-relief      => groove,
                                  -borderwidth => 1,
                                  -anchor      => "w",
                                  -padx        => 115,
                                  -pady        => 1);

    $E_espera->pack(-side=>"top", -fill=>"x");
    my $Bsalir = $modulo2_v -> Button(-text => 'SALIR',
                                      -command => sub{ $Win->destroy;});
    $Bsalir->pack(-side => 'top');

}


sub bitacora_m2{

    my $bitacora_v = MainWindow->new();
    $bitacora_v->configure(-title => 'BITACORA');

    $Itext = $bitacora_v->Scrolled( 'Text' );
    open(SOURCE, "bitacoraUID") or die "no se puede abrir";
    tie(*TEXT, 'Tk::Text', $Itext);
    print TEXT <SOURCE>;
    close(SOURCE);
    $Itext->pack();

    my $Bsalir_m2 = $bitacora_v->Button(-text    => CERRAR,
                                -command => sub { $bitacora_v->destroy;});

   $Bsalir_m2->pack (-side => 'top' );
}


sub Monitoreo_m2{

      open(PASS,"<archivo");

      while(<PASS>)
      {
          #LE quita el salto de linea y obtiene los elementos del archivo
          #/etc/passwd.

          chop;

          ($login,$p,$uid,$guid,$shell,$des,$home)=split(':');

          if ($login !~/root/ && $uid=~/^0/){

            open (CORREO, "|mail -s usuarios vero\@upn.mx");
```

```
            print CORREO "========================================== ";
            print CORREO "\nREPORTE DE MONITOREO DE USUARIOS CON UID 0";
            print CORREO "\n==========================================";
            print CORREO "\n\nReporte generado por:\t ";
            print CORREO "$usuario";
            print CORREO "\Reporte creado el:\t";
            print CORREO "$fecha\n";
            print CORREO "------------------------------------------ ";
            print CORREO "\n\nNombre del host:\t$nombre_maq";
            print CORREO "Archivo utilizado:\t/etc/passwd";
            print CORREO "\n\nEl usuario";
            print CORREO " ";
            print CORREO $login;
            print CORREO " ";
            print CORREO "tiene UID 0 \n";
            print CORREO "La linea es

                        $login:$p:$uid:$guid:$shell:$des:$home";


          }


        else
          {
             open(BIT ,">bitacoraUID");

            $nombre= "Nombre de host:";
            print BIT $nombre;
            print BIT $nombre_maq;
            print BIT "El monitoreo se realizo\n";
            print BIT $fecha;
            print BIT "\nNo se encontro ningun usuario con UID 0\n";
            print BIT "----------------------------------------\n";

            close(BIT);
          }

       }

       close(CORREO);

    }


sub modulo3{

    #Obtiene los archivos con SUID y/o GUID

    `find .  -perm -4000  -type f -print > suidSGI`;
    `find .  -perm -2000  -type f -print >> suidSGI`;



    my $modulo3_v = MainWindow->new();
```

```
    $modulo3_v->configure(-title => 'ARCHIVOS SUID Y/O SGID');
    $E_imagen = $modulo3_v->Button();
    $img_archivos = $E_imagen->Photo(-file => 'wstux.gif');
    $E_imagen->configure(image=>$img_archivos);
    $lab->pack(-side => 'left');
    my $Bmonitoreo=$modulo3_v->Button(-text    => 'MONITOREO',
                         -command => \&Monitorea_m3);
    $Bmonitoreo->configure(-width =>19);
    $Bmonitoreo->pack(-side => 'top');
    my $Barchivos=$modulo3_v->Button(-text => 'ARCHIVOS SUID SGID');
    $Barchivos->pack(-side => 'top');
    $status = $Win->Label(-text       => "",
                          -relief      => sunken,
                          -borderwidth => 2,
                           -anchor     => "w",
                           -padx       => 15,
                           -pady       => 36);

    $E_separa->pack(-side => "top",
                    -fill => "x");
    my $Bsalir = $modulo3_v -> Button(-text => 'SALIR', command => sub
                          { $modulo3_v->destroy;});
    $Bsalir->pack(-side => 'top');

}


#Funcion que busca los archivos con SUID y/o SGID
sub Monitorea_m3
{

 $Icorreo=`cat arch_configuracion.txt | grep Destinatario_suid|cut  -d= -
f2 `;

    open(Bit,">>BITACORA_ARC");

    foreach(1..2)
    {
      `find . -perm -4000  -type f -print  > gui`;
      `find . -perm -2000  -type f -print  >> gui`;
      `diff suidSGI gui >ngui`;

      $Iarchivos=`cat ngui |grep / |wc -l`;
      $Isalir_arch="no";


while($Isalir_arch eq "no")
      {
          if($Iarchivos >= '1')
          {

            open(CORREO_ARCH, "|mail -s archivos root\@localhost");
            print CORREO_ARCH "========================================= ";
```

```
            print CORREO_ARCH "\nREPORTE DE MONITOREO DE ARCHIVOS SUID
SGID";
            print CORREO_ARCH "\n=====================================";
            print CORREO_ARCH "\n\nReporte generado por:\t ";
            print CORREO_ARCH "$usuario";
            print CORREO_ARCH "\Reporte creado el:\t";
            print CORREO_ARCH "$fecha\n";
            print CORREO_ARCH "-------------------------------------- ";
            print CORREO_ARCH "\n\nNombre del host:\t$nombre_maq";
            print CORREO_ARCH "\n\nSe encontro un cambio en base de
archivos SUID y/o SGID";
            print CORREO_ARCH " ";
            print CORREOl "El archivo es: ";
            print CORREO "tiene UID 0 \n";

          }

          else
          {
            print Bit "No se encontraron cambios";
          }
      }


}


sub modulo4{

    #Se crea la ventana con su titulo.
    my $modulo4_v = MainWindow->new();

    #Crea los botones para este modulo
    $modulo4_v->configure(-title => 'SISTEMA DE ARCHVIOS');
    $Bimagen=$modulo4_v->Button();
    $img_m4 =$Bimagen->Photo(-file => 'wstux.gif');
    $img_m4->configure(image=>$icon);
    $img_m4->pack(-side => 'left');

    my $Bmonitoreo_m4=$modulo4_v->Button(-text =>'MONITOREO');
    $Bmonitoreo_m4->pack(-side =>'top');
    my $Bbase=$modulo4_v->Button(-text => 'BASE');
    $Bbase->configure(-width => 12);
    $Bbase->pack(-side => 'top');

    $E_separa_m4 = $modulo4_v->Label(-text        =>"",
                                     -relief      =>sunken,
                                     -borderwidth =>2,
                                     -anchor      =>"w",
                                     -padx        =>15,
                                     -pady        =>36);

    $E_separa_m4->pack(-side => "top",
                       -fill => "x");
```

```
     my $Bsalir = $modulo4_v -> Button(-text    => 'Salir',
                                       -command => sub
                                      { $modulo4_v->destroy;});
    $Bsalir->pack(-side => 'top');

}


sub modulo5{

    my $modulo5_v = MainWindow->new();
    $B_imagen=$modulo5_v->Button();
    #Imagen para el módulo 5
    $img_m5 =$B_imagen->Photo(-file => 'wstux.gif');
    $B_imagen->configure(image=>$img_m5);
    $B_imagen->pack(-side => 'left');

    #Botones para el modulo 5
    $modulo5_v->configure(-title => 'RESPALDOS');
    my $Brespaldar=$modulo5_v->Button(-text => 'RESPALDAR');
    $Brespaldar->configure(-width => 23);
    $Brespaldar->pack(-side => 'top');

    my $Barchivo_con=$modulo5_v->Button(-text =>'ARCHIVO CONFIGURACION');
    $archivo_con->pack(-side => 'top');
    my $Bfirma=$modulo5_v->Button(-text => 'FIRMA');
    $Bfirma->configure(-width => 23);
    $Bfirma->pack(-side => 'top');
    my $Bscp=$modulo5_v->Button(-text => 'SCP');
    $Bscp->configure(-width => 23);
    $Bscp ->pack(-side => 'top');

    $E_seprar = $modulo5_v->Label(-text         => "",
                                  -relief       => groove,
                                  -borderwidth  => 2,
                                  -anchor       => "w",
                                  -padx         =>  5,
                                   -pady        =>  10);

    $E_separa->pack(-side => "top",
                    -fill => "x");

    my $Bsalir = $modulo5_v->Button(-text     => 'SALIR',
                                -command  => sub { $modulo5_v ->destroy;});

   $Bsalir->pack(-side => 'top');

}
```

```
#Función para el editor de archivos.

sub Editor_c {


    $editorarch_v=MainWindow->new;
    $editorarch_v->configure(-title => "EDITOR DE ARCHIVOS");

     #Se crea el frame para los botones del editor.
    $F_editor=$editorarch_v->Frame->pack(-side => 'top',
                                         -fill => 'x');



    $F_editor->Label(-text => "Nombre del archivo:")


                   ->pack(-side   => 'left',
                          -anchor => 'w');

    $F_editor->Entry(-textvariable=>\$nombre_arch)->
                            pack(-side   => 'left',
                                 -anchor => 'w',
                                 -fill   => 'x',
                                 -expand => 1);

     #Se crean los botones para el editor
    $F_editor->Button(-text        => "Salir",
                      -background  => 'white',
                       -command    =>
                      sub { $editorarch_v->destroy;})
                           ->pack(-side => 'right');


    $F_editor->Button(-text       => "Salvar",
                      -background => 'white',
                      -command    => \&Salvar_arch)

                       ->pack(-side   => 'right',
                              -anchor => 'e');

    $F_editor->Button(-text         => "Cargar",
                      -background   => 'white',
                      -command      => \&Cargar_arch)
                      ->pack(-side   => 'right',
                             -anchor  => 'e');

    $F_editor->Button(-text        =>"Abrir",
                      -background  => 'white',
                      -command     => \&Abrir_arch)
                      ->pack(-side   => 'right',
                             -anchor => 'e');
```

```
$editorarch_v->Label(-textvariable => \$info,
                     -relief       => 'ridge')
                        ->pack(-side => 'bottom',
                               -fill => 'x');


   $Itexto=$editorarch_v->Scrolled("Text")->pack(-side    => 'bottom',
                                                 -fill    => 'both',
                                                 -expand => 1);


    #Función para abrir los archivos.

    sub Abrir_arch{

      if($nombre_arch eq '')
      {

       my  $pregunta=$editorarch_v->Dialog(-title => 'No se puede abrir',
                                           -text  => 'Debes escribir el
nombre del archivo',
                                  -default_button => 'Continuar',
                                  -buttons        => ['Continuar'],
                                  -bitmap         => 'question')->Show();

       }
      else
      {
          if(-e $nombre_arch)
          {
            open(FL, "$nombre_arch");

            while(<FL>){
                $t->insert("end", $_);
            }
            close(FL);

          }
          else
          {
            my  $pregunta2=$editorarch_v->Dialog(-title => 'No existe',
                                        -text => 'No existe el archivo',
                                        -default_button => 'Continuar',
                                        -buttons        => ['Continuar'],
                                        -bitmap        => 'question')-
>Show();
          }
       }
     }
```

```
    sub Cargar_arch{

      $info="Cargando el archivo ...'$nombre_arch'..";
      $t->delete("1.0", "end");
      if(!open(FH, "$nombre_arch"))
      {
          $t->insert("end", "ERRROR");
          return;

          {
            while(<FH>){
                $t->insert("end", $_);
            }
            close(FH);
            $info="Archivo cargado";
          }

      }


    }

    sub Salvar_arch{

      $info="Salvando el archivo";
      open(FH,">$nombre_arch");

      print FH $t->get("1.0", "end");
      $info="Saved";
    }
}


sub guardar_arch {

    my($m)=TEXT;
    print $m "\n";
}

sub bitacora_m2{

    my $bitacora_vm2 = MainWindow->new();
    $bitacora_vm2->configure(-title => 'BITACORA');

    $I_text = $bitacora_vm2->Scrolled( 'Text' );
    open(SOURCE, "bitacoraUID") or die "no se puede abrir";
    tie(*TEXT, 'Tk::Text', $I_text);
    print TEXT <SOURCE>;
    close(SOURCE);
    $text->pack();
```

```
    my $Bsalirb_m2 = $bitacora_vm2->Button(-text    => CERRAR,
                                           -command =>
                                          sub { $bitacora_vm2->destroy;});

    $Bsalirb_m2->pack (-side => 'top' );
}


sub Ayuda {

my $ayuda_v = MainWindow->new;
$ayuda_v->title('Ayuda del Sistema');
my $Ccanvas = $ayuda_v->Scrolled('Canvas',-background=>'white', -width =>
400, -height => 400);
my $I_informa  = $Ccanvas->createText(100, 50, -text => '
                              INTRODUCCION

                              El MDS es una aplicacion que permite a los
                              administradores, monitorear su sistema.

                             Esta aplicacion cuenta con 5 modulos

                               - Servicos Abiertos
                               - Usuarios con UID 0
                               - Archivos SUID y/o SGID
                               - Sistema de archivos
                               - Respaldos



                              Cada uno de ellos es encargado de monitorear
                              una parte del sistema.
                              Y en  cada uno de ellos existe  la opcion de
                              comenzar el monitoreo y detenerlo.

                              Asi como  de revisar  bitacoras, para saber
                              que a  pasado durante el  monitoreo de algun
                              modulo.

                              A continuacion se muestra la ventana
principal
                              del sistema asi como los de los modulos.

                              ');


my $dino = $ayuda_v->Photo(-file => 'wstux.gif');
my $img_ven = $Ccanvas->createImage(150, 300, -image => $dino);
```

```
my $Bexit = $ayuda_v->Button(-text    => 'Cerrar',
                             -command => [$ayuda_v => 'destroy']);
$Ccanvas->pack;
$Bexit->pack;

}
```

#Función que muestra la versión del sistema

```
sub Acerca_de {

    $acerca_de_v=MainWindow->new();
    $acerca_de_v->configure(-width  =>'150',
                            -height =>'150');

    $acerca_de_v->configure(-title => 'Acerca de MDS');

    $Bsistema=$acerca_de_v->Button();
    $img_sist =$Bsistema->Photo(-file => 'mds.gif');
    $Bsistema->configure(image => $img_sist);
    $Bsistema->pack(-side => 'top');

    $Eicono=$acerca_de_v->Label(-text =>'MDS-1.2');
    $Eicono->pack(-side   =>'top',
                  -expand => 1,
                  -padx   =>76,
                  -pady   => 8);

    my $Bcreditos=$acerca_de_v->Button(-text    => "Creditos",
                                       -command => \&Creditos);

    $Bcreditos->pack(-side =>'left');
    my $Bcerrar=$acerca_de_v->Button(-text    =>"Cerrar",
                                     -command => sub {$acerca_de_v-
>destroy;});
    $Bcerrar->pack(-side =>'right');
}


#Funcion que muestra la información acerca del programador

sub Creditos{

    my $creditosv=MainWindow->new;
    $creditosv->configure( -title  => "CREDITOS");
    my $Ccreditos = $creditosv->Scrolled('Canvas',
                              -background => 'white',
                              -width      =>  410,
                              -height     => 160);
```

```
    my $text  = $Ccreditos->createText(189, 60,  -text =>'

                                UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

                                                FACULTAD DE CIENCIAS


                                                  Veronica Vanegas Lomas

                                                                        ');

    my $escudo = $creditosv->Photo (-file => 'unam-2.gif');
    my $img_escudo = $Ccreditos->createImage(50, 60, -image => $escudo);

    $Ccreditos->pack();
}
```

GLOSARIO

passwd El contenido del archivo /etc/passwd determina quién puede acceder al sistema de manera legítima y qué se puede hacer una vez dentro del sistema.

suid Este bit describe permisos al identificador de usuario del archivo. Cuando el modo de acceso de ID de usuario está activo en los permisos del propietario, y ese archivo es ejecutable, los procesos que lo ejecutan obtienen acceso a los recursos del sistema basados en el usuario que crea el proceso (no el usuario que lo lanza).

sgid Si está activo en los permisos de grupo, este bit controla el estado de "poner id de grupo" de un archivo. Actúa de la misma forma que SUID, salvo que afecta al grupo. El archivo tiene que ser ejecutable para que esto tenga algún efecto.

scp El comando scp permite copiar archivos entre dos máquinas. Utiliza ssh para la transmisión de la información, por lo que ofrece la misma seguridad que el ssh.

i-nodo Un i-nodo contiene el tamaño de un archivo, las tres fechas relacionadas con el archivo (modificación del contenido, acceso al contenido y modificación del i-nodo), los permisos, el dueño, el grupo, el número de enlaces e información sobre dónde están físicamente los datos (los bytes) del archivo en el disco.

Sistema de archivos El sistema de archivos de UNIX se denomina sistema jerárquico de archivos.

Firma digital La firma digital permite que una parte pueda enviar un mensaje "firmado" a otra parte, con las propiedades de autentificación.

BIBLIOGRAFÍA

- Eric J. Braude
  Ingeniería de software
  Una perspectiva orientada a objetos
  Alfaomega, 2003
  ISBN 970-15-0851-3

- Michael Mcmillan
  Perl from the ground
  McBraw-Hill, 1998
  ISBN 0-07-882404-4

- Mike Glover, Aidan Humphreys
  Perl 5
  Waite Group, 1996
  ISBN 1571690581

- Simson Garfinkel y Gene Spafford
  Seguridad práctica en Unix e Internet
  McGRAW-Hill, 1999
  ISBN 970-10+2071-5

http://www.cs.cinvestav.mx/~EVOCINV/tutorials/latex.htm	(mayo, 2004)
http://www.iespana.es/perl-es/TK.htm	(marzo, 2004)
http://kal-el.ugr.es/~jmerelo/recursos-PERL.html	(junio, 2004)
http://www.perl.com/	(febrero, 2004)
http://PERL-TK/181977.html	(enero, 2004)
http://www.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html	(enero, 2004)
http://www.pconline.com/~erc/perltk.htm	(febrero 2004)