TESIS DEFENDIDA POR:

Yuridia Azucena Hernández Ontiveros

Y APROBADA POR EL SIGUIENTE COMITE

M.C. Josefina Rodríguez Jacobo

Director del Comité

Dra. Ana Isabel Martínez García

Miembro del Comité

M. C. César Alfonso Reyes Zamora

Miembro del Comité

M. C. Lidia Elena Gómez Velazco

Miembro del Comité

M.C. José Luis Briseño Cervantes

Jefe del Departamento de Ciencias de la computación

Dr. Luis Alberto Delgado Argote

Director de Estudios de Posgrado

5 de julio del 2002.

CENTRO DE INVESTIGACION CIENTIFICA Y DE EDUCACION SUPERIOR DE ENSENADA

División de Física Aplicada
Departamento de Ciencias de la Computación

Pruebas: Una guía de valoración de sistemas de software durante el ciclo de vida

Tésis

Que para cubrir parcialmente los requisitos necesarios para obtener el grado de MAESTRO EN CIENCIAS presenta:

YURIDIA AZUCENA HERNANDEZ ONTIVEROS

Ensenada Baja California, 5 de julio de 2002.

RESUMEN de la tésis de **YURIDIA AZUCENA HERNANDEZ ONTIVEROS** presentada como requisito parcial, para la obtención del grado de **MAESTRO EN CIENCIAS** en **CIENCIAS DE LA COMPUTACION**, Ensenada Baja California, México, julio de 2002.

PRUEBAS: UNA GUIA DE VALORACION DE SISTEMAS DE SOFTWARE DURANTE EL CICLO DE VIDA

Resumen aprobado por:

_____

M.C. Josefina Rodríguez Jacobo
Director de Tésis

Las pruebas de sistemas de software constituyen una de las etapas más importantes del proceso de desarrollo de software. Estas pruebas sirven como una medida de valoración de la calidad de los productos de software previos a su liberación.

Esta tésis presenta una revisión del "estado del arte de las pruebas de software", se analiza información sobre los diferentes tipos de pruebas mecanismos y métricas que existen y la forma en que se aplican. A partir de este análisis se desprende la necesidad de contar con técnicas y mecanismos que faciliten la realización de esta tarea. Para dar respuesta a esta necesidad se desarrolla una guía que facilite la aplicación de las pruebas a sistemas de software. La guía se divide en cinco etapas: planificación, aplicación, realización de reportes, seguimiento y aplicación de métricas de la prueba.

La guía de pruebas muestra una manera sistemática para realizar el proceso de pruebas de software. Su objetivo es detectar el mayor número posible de defectos en cada fase del desarrollo del producto. En complemento a esta guía se propone el uso de listas de revisión en cada fase del proceso de desarrollo del software. Estas listas son útiles para verificar los requerimientos, especificaciones, arquitectura del sistema y su implementación.

La guía propuesta ofrece a los ingenieros de software una herramienta para mejorar la calidad del software. El uso de la misma se ilustra con el "Sistema Colaborativo de Apoyo a Revisiones Técnicas" (SCART).

Un grupo de seis personas, conocedoras del área, relizaron una evaluación de las cinco etapas que comprende la guía. Aunque esta evaluación es preliminar, los comentarios obtenidos muestran una tendencia de que la guía es útil para apoyar el proceso de pruebas a sistemas de software.

Palabras clave: pruebas de software, listas de revisión, verificación.

Abstract of the thesis of **Yuridia Azucena Hernández Ontiveros**, submitted in partial fulfillment of the requirements to obtain the degree **of MASTER IN COMPUTER SCIENCE**. Ensenada, Baja California, Mexico july 2002.

Testing: A guide for Software System Valuation During Its Life-Cycle

Abstract approved by:

_____

M. C. Josefina Rodríguez Jacobo
Thesis Director

Software system testing constitute one of the most important stages of the software development process. These tests serve as a valuation metric for the quality of software products before they have been released.

This thesis presents a review of the "state-of-the-art on software testing" . It analyses information on different types of test, existing metrics and techniques, and the way these are applied. From this analysis arises the need to use techniques and mechanisms to facilitate the accomplishment of this task. In order to give answer to this need, we development a guide that facilitates the application of software systems testing. The guide consists of five stages: planning, applications, report elaboration, follow through and metric application.

The testing guide shows a systematic procedure to implement the software testing process. Its goal is to detect the largest number of defects in each phase during a product development. As complement of this guide, we propose the use of revision list in each phase of the software development process. These list are useful to verify the requirements, specifications, system architecture and its implementation.

The proposed guide offers a tool to improve the quality of software. We illustrate its use on the Collaborative System for the Support of Technical Reviews (SCART). A group of six people, experts in the area, evaluated the five stages of the guide. Although this evaluation is preliminary, their comments show a tendency that indicates this guide is useful to support the software system testing process.

Keywords: software testing, revision list, verification.

Dedicatorias

A Dios
Porque me ha dado una familia muy bonita.

A mi familia

Porque juntos hemos aprendido a ser cada día mejores y ha tener siempre en claro que es lo que queremos y el camino que hay que recorrer para alcanzarlo.
Gracias papá por trabajar tanto para sacarnos adelante
Gracias mamá por darnos lo mejor de ti y por encaminarnos a mi y a mis hermanos por el buen camino.
Gracias Alfredo por ponerme el ejemplo para lograr tener éxito en la vida
Gracias Juan por prestarme tus cuadernos y libros de ingeniería y por ser tan agradable

Agradecimientos

A la M. en C. Josefina Rodríguez Jacobo, mi directora de tésis, por su atención y tiempo que me dedicó desde mi entrada a CICESE. Josefina gracias por ser paciente y comprenderme y por haber hecho posible la realización de este trabajo.

A los miembros del Comité de Tésis, la Dra. Ana Isabel Martínez García, al M.C. César Alfonso Reyes Zamora y la M. C. Lidia Elena Gómez Velazco que con sus correcciones enriquecieron mi trabajo de tésis.

A mi mejor amiga Ilse por estar conmigo en los momentos buenos y malos.

A mis compañeros de clase Adrian, César, Enrique, Juan, José, José Adrian, Sonia, Aglay, Jorge, Rafael, Gabriel, Elitania, Brenda, Francisco y Yolanda, de quienes aprendí tanto. En especial a Enrique, Rafa y a José por tener siempre una sonrisa y palabras de aliento para salir adelante.

A mis amigos Ricardo, Everardo, Silver, Carlos y Marco por escucharme y por alivianarme cuando me sentía triste.

Al Consejo Nacional de Ciencia y Tecnología por otorgarme la beca crédito que me permitió realizar estos estudios.

TABLA DE CONTENIDO

CAPITULO I. INTRODUCCION.....1

I.1. Planteamiento del problema.....4

I.2. Objetivo de la tésis.....6

I.3. Alcances.....6

I.4 Organización de la tésis.....7

CAPITULO II. CARACTERIZACION DE LAS PRUEBAS.....8

II.1. Introducción.....8

II.2. Propiedades de las pruebas.....8

II.2.1. Prueba estructural vs funcional 8

II.2.2. Prueba dinámica vs estática 9

II.2.3. Manual vs automatizada 10

II.3. Clasificación de las pruebas 10

II.3.1. Pruebas de bajo nivel 11

II.3.1.1. Prueba de unidad 11

II.3.1.2. Prueba de integración.....11

II.3.2. Pruebas de alto nivel 12

II.3.2.1.Prueba de usabilidad.....12

II.3.2.2. Prueba de función.....13

II.3.2.3. Pruebas de sistema..... 14

II.3.2.4. Pruebas de aceptación.....17

II.4. Mecanismos de prueba.....20

II.5. Casos de prueba.....27

II.5.1. Caja blanca.....28

II.5.2. Caja negra.....28
CAPITULO III. GUIA DE PRUEBAS.....30
III.1. Introducción.....30
III.2. Propósito de la guía de pruebas.....33
III.3. Alcances de la guía de pruebas.....33
III.4. Establecimiento del plan de prueba.....34
III.4.1. Introducción.....34
III.4.2. Objetivos.....35
III.4.3. Perspectiva del plan de pruebas.....35
III.4.4. Características del plan de pruebas.....36
III.4.5. Restricciones.....36
III.4.6. Contenido del plan de pruebas.....37

III.5. Aplicación de la prueba.....42

III.5.1. Perspectivas de la aplicación de la prueba.....42

III.5.2. Características de la aplicación de la prueba.....43

III..5.3. Restricciones de la aplicación de la prueba.....43

III.5.4. Diseño de las pruebas 43

III.5.5. Caso de prueba.....45

III.5.6. Procedimiento de prueba.....46

III.6. Realizar reportes de pruebas.....47

III.6.1 Perspectivas de los reportes de pruebas.....48

III.6.2. Características de los reportes de pruebas.....48

III.6.3. Restricciones de los reportes de pruebas.....49

III.6.4. Bitácora de prueba.....49

III.6.5. Reporte de incidentes de prueba.....50

III.6.6. Reporte de transmisión.....51

III.6.7. Resumen de reportes de las pruebas.....52

III.7. Seguimiento de la prueba.....54

III.7.1. Perspectivas del seguimiento de la prueba.....55

III.7.2. Características del seguimiento de la prueba.....55

III.7.3. Restricciones del seguimiento de la prueba.....55

III.7.4. Gráfica de elementos probados a la fecha vs estimados.....56

III.7.5 Gráfica de estado de los elementos.....56

III.7.6. Gráfica de la distribución de defectos en los elementos.....56

III.7.7. Gráfica de defectos corregidos por tipo.....57

III.7.8. Reporte de pruebas.....57

III.8. Aplicación de métricas.....57

III.8.1. Perspectivas de la aplicación de métricas.....58

III.8.2. Características de las métricas.....58

III.8.3. Restricciones de las métricas.....59

III.8.4. Análisis de los defectos.....59

III.8.4.1. Distribución de defectos por módulo.....59

III.8.4.2. Distribución relativa de defectos.....59

III.8.4.3. Distribución de defectos por el ambiente (plataforma).....60

III.8.4.4. Distribución de defectos por tipo.....60

III.8.4.5. Distribución de defectos por quién lo encontró.....60 III.8.4.6. Distribución de defectos por fase en el ciclo de vida.....61

III.8.4.7. Distribución de causas de defectos.....61 III.8.5. Magnitud de la prueba.....61

III.8.5.1. Caminos ejecutados.....61

III.8.5.2. Instrucciones ejecutadas.....62

III.8.6. Efectividad de la prueba.....63

III.8.6.1. Defectos descubiertos debido a las pruebas.....63

III.8.6.2. Eficacia de la selección de pruebas.....64

III.8.6.3. Defectos encontrados en relación al código.....64

III.8.6.4. Tiempo por corregir defectos.....65

III.8.7. Costo de la prueba.....65

III.8.7.1. Costo de la etapa de prueba.....66

III.8.7.2. Presupuesto alcanzado.....66

III.8.7.3. Prueba de automatización.....67

CAPITULO IV. UNA PROPUESTA DE PRUEBAS EN EL PROCESO DE DESARROLLO DE SOFTWARE.....
.....69

IV.1. Fases de pruebas durante el ciclo de desarrollo del software.....72

IV.2. Análisis del sistema.....72

IV.2.1. Prueba o verificación al documento de análisis de requerimientos.....
..73

IV.2.2 Prueba o verificación al documento de especificación de requerimientos.....
..77

IV. 3. Diseño del sistema.....79

IV.3.1 Prueba o verificación al diseño de alto nivel 79

IV.3.2. Prueba o verificación al documento de diseño de bajo nivel .....
.....80

IV.3.3. Prueba o verificación al diseño del sistema utilizando el lenguaje de modelado UML
.....81

IV.3.4. Prueba o verificación de entrada/salida.....81

IV.3.5. Prueba o verificación de las condiciones estéticas.....82

IV. 3.6. Prueba de validación.....83

IV.3.7. Prueba o verificación de las búsquedas.....84

IV.3.8. Prueba o verificación de procedimientos.....85

IV.4. Implementación.....86

IV.4.1. Prueba o verificación de la codificación.....86

IV.4.2. Prueba o verificación de archivos.....87

IV.4.3. Prueba o verificación de interfaz gráfica.....88

IV.4.4. Prueba o verificación de navegación.....89

IV.4.5. Prueba o verificación de integridad.....90

IV.4.6. Prueba o verificación de transacciones.....91

IV.5. Pruebas de liberación.....92

IV.5.1. Prueba o verificación instalación.....92

IV.5.2. Prueba o verificación de la ayuda.....93

IV.5.3. Prueba o verificación de requerimientos no funcionales.....94

IV.5.4. Prueba o verificación de desempeño.....95

CAPITULO V. EVALUACION PRELIMINAR DE LA GUIA DE PRUEBAS 97

V.1. Evaluación preliminar de la guía de pruebas.....98

V.1.1. Selección de usuarios.....98

V.1.2. Entrevista.....99

V.1.3. Resultados de la entrevista.....99

V.1.4. Comentarios y sugerencias.....101

V.1.5. Modificaciones a la guía de pruebas.....103

V.2. Utilización de la guía de pruebas.....104

V.3. Una propuesta para probar la guía de pruebas.....106

V.3.1. Aplicación de la guía al sistema “HEPRODES”.....112

V.3.2. Aplicación de la guía al sistema “SCART”.....113

CAPITULO VI. CONCLUSIONES, APORTACIONES Y TRABAJO FUTURO 116

VI.1. Conclusiones.....116

VI.2. Aportaciones.....118

VI.3. Trabajo futuro.....119

BIBLIOGRAFIA.....
120

APENDICE A. DEFINICIONES, ACRONISMOS Y ABREVIATURAS.....125

APENDICE B. MATRICES.....127

B.1. Matriz de clases / funciones.....128

B.2. Matriz de seguimiento de requerimientos.....129

B.3. Matriz de riesgos.....130

APENDICE C. LISTAS DE REVISION DE DEFECTOS.....131

C.1. Lista de revisión de defectos R1.....131

C.2. Lista de revisión de defectos U1.....133

C.3. Lista de revisión de defectos DA1.....135

C.4. Lista de revisión de defectos DB1.....137

C.5. Lista de revisión de defectos ES1.....140

C.6. Lista de revisión de defectos CE1.....141

C.7. Lista de revisión de defectos P1 143

C.8. Lista de revisión de defectos V1.....144

C.9. Lista de revisión de defectos E1 145

C.10. Lista de revisión de defectos B1.....147

C.11. Lista de revisión de defectos PR1.....148

C.12. Lista de revisión de defectos C1.....149

C.13. Lista de revisión de defectos AR1.....153

C.14. Lista de revisión de defectos I1.....154

C.15. Lista de revisión de defectos CG1.....157

C.16. Lista de revisión de defectos N1.....159

C.17. Lista de revisión de defectos ID1.....160

C.18. Lista de revisión de defectos T1.....161

C.19. Lista de revisión de defectos IN1.....162

C.20. Lista de revisión de defectos AY1.....164

C.21. Lista de revisión de defectos RNF1.....166

C.22. Lista de revisión de defectos D1.....168
APENDICE D. FORMATO DE REGISTRO DE DEFECTOS.....169
APENDICE E. APLICACIÓN DE LA GUIA A SCART.....170
E.1. Plan de pruebas.....170
E.2. Aplicación de las pruebas.....175
E.2.1. Diseño de pruebas.....175
` E.2.2. Procedimiento de pruebas.....178
E.2.3. Casos de prueba.....179
E.3. Realizar reportes de las pruebas.....188
E.3.1. Bitácora de pruebas.....188
E.3.2. Reporte de incidentes de pruebas.....190
E.3.3. Reporte de transmisión.....193

E.3.4. Resumen de reportes de pruebas 193

E.4. Seguimiento de las pruebas.....196

E.5. Métricas.....209

APENDICE F. CUESTIONARIO DE EVALUACION.....213

LISTA DE FIGURAS

Figura 1. Diagrama con las etapas del proceso de la guía de pruebas.....32

Figura 2. Forma para la verificación de la prueba al documento de requerimientos.....75

Figura 3. Evaluación de las etapas de la guía de pruebas.....100

Figura 4. Gráfica del estado de los elementos.....198

Figura 5. Gráfica de distribución de defectos en el análisis del sistema.....199

Figura 6. Gráfica de distribución de defectos en el diseño del sistema.....201

Figura 7. Gráfica de distribución de defectos en el sistema.....203

Figura 8. Gráfica de distribución de defectos críticos.....205

Figura 9. Gráfica de distribución de defectos corregidos mayores.....206

Figura 10. Gráfica de distribución de defectos corregidos menores.....207

Figura 11. Distribución de defectos por clase.....209

Figura 12. Distribución de defectos por fase en el ciclo de vida.....210

LISTA DE TABLAS

Tabla I. Características generales de los diferentes tipos de pruebas.....19

Tabla II. Relación de los mecanismos de pruebas con las fases del ciclo de vida del software 26

Tabla III. Calendarización de actividades.....109

Tabla IV. Tabla de riesgos y contingencias.....110

Tabla V. Calendario de pruebas.....174

Tabla VI. Tabla de ejecución de actividades.....189

Tabla VII. Tabla de elementos probados a la fecha vs estimados.....197

Tabla VIII. Tabla de datos del estado de los elementos.....199

Tabla IX. Tabla de defectos del análisis del sistema.....200

Tabla X. Tabla de defectos del diseño del sistema.....202

Tabla XI. Tabla de defectos del sistema.....204
Tabla XII. Tabla de distribución de defectos corregidos por tipo.....208

Introducción

Para que una empresa de desarrollo de software tenga éxito y permanezca competitiva en el mercado es necesario que realice sistemas de calidad. Una empresa que produce software consistente utilizando eficientemente sus recursos tiene grandes probabilidades de crecer y permanecer en el gusto de los clientes.

Las pruebas de software son una medida para evaluar la calidad del software que se construye, son un filtro para eliminar los defectos del sistema y son una parte del proceso de desarrollo de software (Pressman, 1999).

Las pruebas de software suelen realizarse durante el desarrollo del sistema y después de que éste ha sido terminado. Estas consisten en ejercitar el sistema utilizando datos similares a los datos reales, observar los resultados y deducir la existencia de errores o insuficiencias del programa a partir de las anomalías de los resultados (Sommerville, 1988).

Unas pruebas efectivas requieren la aplicación de métodos, técnicas y herramientas. El ingeniero de pruebas utiliza sus habilidades y experiencia para verificar que el sistema haga lo que debe hacer y obtenga resultados confiables en un tiempo razonable.

Todos los sistemas de software tienen su grado de complejidad, sin embargo resulta más fácil probar sistemas bien diseñados que los que usaron un diseño mediocre. Esto depende mucho del grado de madurez de la empresa desarrolladora. Sin embargo, un mal diseño no justifica que las pruebas fracasen, al contrario, en estos casos se debe poner mayor atención en las fases de desarrollo y ser más estrictos con el propósito de

eliminar los defectos que se presenten durante el proceso. Como las pruebas empiezan en la definición de los requerimientos, es posible detectar errores en fases tempranas y obtener resultados más seguros desde el inicio y en cada una de las fases (Perry, 1995).

A pesar de lo necesarias e imprescindibles que son las pruebas, no todos los países les dan la importancia que merecen. En México son limitados los esfuerzos que se han hecho para mejorar el proceso de pruebas, lo que genera software poco competitivo y de baja calidad. En contraste con otros países como Estados Unidos, India, Brasil donde las pruebas tienen el mismo nivel que las demás fases del ciclo de desarrollo del software.

En los últimos años, el avance en la investigación sobre pruebas de software ha disminuido y los investigadores no han encontrado nuevos métodos o técnicas que revolucionen la manera en que éstas se realizan. Las mismas pruebas que se utilizaban para probar sistemas estructurados se han adaptado para probar las nuevas aplicaciones que han surgido en el mercado como sistemas cliente/servidor, sistemas sobre web, entre otros.

Investigaciones más recientes se orientan hacia el mejoramiento del proceso de pruebas cubriendo diversos aspectos como uso de métricas para proyectos pequeños (Dutoit, 1998), desarrollo rápido de casos de prueba (Yamaura, 1998), evaluación de sistemas utilizando métricas de comunicación (Dutoit, 1998), evaluación de la legalidad y certificación de sistemas (Wakid, 1999), entre otros.

Por otra parte, la revisión bibliográfica no muestra evidencia del uso de guías o modelos de uso común que faciliten a los desarrolladores de software el proceso de pruebas de sistemas. En su lugar, la mayoría de los autores mencionan el desarrollo de un plan de pruebas. Un plan de pruebas es un conjunto de requerimientos para realizar las pruebas, el

cual contiene información sobre el sistema, el ambiente de pruebas, calendario de actividades, riesgos y contingencias, entre otros; por lo tanto no se considera guía o modelo.

Adicionalmente, el instituto de ingenieros en electrónica y eléctricos de sus siglas en inglés IEEE (Institute of Electrical Electronics Engineers), ha proporcionado estándares para apoyar el proceso de pruebas entre los cuales están los estándares para realizar pruebas de unidad y para la documentación de la prueba (estándares: 829-1998, 1008-1987(R1993) y 1012-1998). Estos estándares tienen la limitante de que no proporcionan mucha de la información que se necesita para realizar las pruebas y son muy generales debido a que son de dominio público.

Debido a la falta de lineamientos y metodologías que ayuden a los desarrolladores de software a probar estos, se presenta este trabajo de tésis, el cual contiene una guía de pruebas que apoya el desarrollo de sistemas de alta calidad. Esta guía permite ampliar la visión del proceso de pruebas, mostrando una panorámica del contexto de las pruebas de software con información sobre tipos de prueba, mecanismos, métricas y técnicas. Algunas de sus características se describen a continuación:

- Familiariza al usuario con la terminología más usada en el contexto de pruebas de sistemas.
- Mantiene un plan de pruebas de acuerdo a los estándares marcados por el IEEE.
- Organiza la aplicación de la prueba desglozando el plan; en casos, procedimientos y diseños de prueba.

- Registra la ejecución de la prueba en bitácoras de prueba, reportes de incidentes, reportes de transmisión y resumen de reportes de pruebas.
- Supervisa la ejecución del proceso de pruebas mediente gráficas que miden el avance de la prueba.
- Evalúa los resultados de las pruebas aplicando un conjunto de métricas.

Planteamiento del problema

Una de las problemáticas que en la actualidad enfrentan los desarrolladores de sistemas de software, es que la mayoría de las veces los sistemas que se entregan resultan con baja calidad y deficientes. Ocasionando inconformidad de los clientes y pérdida de recursos como tiempo y dinero.

El software de alta calidad es aquel que satisface las necesidades del cliente. En ingeniería de software la calidad se logra aplicando un proceso de pruebas eficiente que permita detectar los defectos del sistema para corregirlos antes de su presentación al cliente.

El problema con las pruebas de sistemas es que no se les dá la importancia que requieren. Por ejemplo, en empresas desarrolladoras de software es muy común que los administradores de los proyectos contraten personal para otras áreas como diseño ó programación y que las pruebas sean realizadas por los mismos programadores. Como los

programadores dividen su tiempo entre detectar los defectos y corregirlos, entonces se ven obligados a entregar el sistema sin terminar las pruebas.

Otro problema es que algunos ingenieros perciben a las pruebas como un "proceso destructivo". Esto hace que se eviten o se desplacen hasta el final del proceso de desarrollo de software, sin considerar que los defectos se presentan en todas las fases y que los más costosos son aquellos que se encuentran en las fases más tempranas.

Otro factor que dificulta el proceso de pruebas, es que los ingenieros de prueba requieren conocimientos y experiencia en las fases del proceso de desarrollo de software y estar familiarizados con los métodos y técnicas que existen en el mercado para realizar pruebas efectivas. Encontrar este tipo de gente no es una tarea fácil en nuestro país.

En la revisión bibliográfica no se encontró una metodología clara para realizar el proceso de pruebas de sistemas de software que facilite a los ingenieros de prueba su aprendizaje.

Tomando en cuenta la problemática presentada, es clara la necesidad de crear una guía que brinde apoyo a los ingenieros de prueba para realizar el proceso de las mismas. La guía permitirá a los ingenieros probar sistemáticamente cada fase del proceso de desarrollo de software de manera que este se dé a la par con las pruebas, eliminando los errores conforme éstos se vayan presentando.

Objetivo de la tésis

Debido a la necesidad de contar con una guía que facilite y apoye el proceso de pruebas de sistemas durante el ciclo de vida del mismo, se plantea como objetivo principal de este trabajo:

Crear una guía de valoración de sistemas de software que permita probar de manera efectiva a los mismos en cada una de las fases del ciclo de su desarrollo del software.

Alcances

En esta tésis se describe una guía para probar sistemas de software durante su proceso de desarrollo. La guía establece las bases necesarias para realizar las pruebas; sin embargo, no sugiere la aplicación de pruebas particulares. Esta deja a criterio del ingeniero de pruebas la selección del material que considere más importante según las necesidades del software que esté bajo prueba.

Organización de la tésis

En el capítulo II se muestra una caracterización de las pruebas. Se describen los conceptos más utilizados en las pruebas de software, se explican sus propiedades, su clasificación y los mecanismos más empleados utilizados por los desarrolladores de software.

En el capítulo III se presenta la guía de pruebas. Se describen sus etapas: desarrollo del plan, aplicación, reportes, seguimiento y métricas.

En el capítulo IV se muestra una propuesta de nuevas pruebas a aplicar en el proceso de desarrollo de software. Se describen las pruebas o la verificación a realizar en cada fase del mismo, utilizando listas de revisión entre las cuales se encuentran: requerimientos, diseño, validación, entrada/salida, condiciones estéticas, búsquedas, entre otros.

En el capítulo V se describen los resultados de la evaluación de la guía, se presentan los comentarios y sugerencias de los usuarios y los cambios realizados.

En el capítulo VI se presentan las conclusiones, aportaciones y trabajo futuro con respecto al estado del arte de las pruebas y las recomendaciones para trabajo futuro.

Caracterización de las pruebas

Introducción

El propósito de este capítulo es apoyar a los usuarios a que adquieran conocimiento sobre la terminología convencional utilizada en el área de pruebas para una mejor comprensión del proceso de aplicación de las mismas. Se describen aspectos generales al contexto de las pruebas: propiedades, su clasificación y algunos mecanismos que se utilizan para realizarlas. El manejo de estos conceptos facilita la comprensión de los capítulos posteriores.

Propiedades de las pruebas

Las propiedades de las pruebas se definen de acuerdo a la forma de aplicación y se dividen en tres tipos: estructural vs funcional, dinámica vs estática y manual vs automatizada.

Prueba estructural vs funcional

Las pruebas se pueden clasificar en dos tipos aquéllas que se aplican sobre la funcionalidad de un sistema o la estructura del mismo. Se realiza el análisis estructural y funcional para asegurar que las pruebas se aplican adecuadamente. En el análisis estructural de las pruebas la tendencia es encontrar errores que ocurren durante la codificación del sistema. En el análisis funcional de las pruebas la tendencia es encontrar errores que ocurren en la implementación de requerimientos o especificaciones del plan.

Las pruebas funcionales aseguran que el sistema cubre los requerimientos y las funciones para el cual fue diseñado. Estas únicamente proporcionan información sobre el resultado del proceso (Perry,1995).

Las pruebas estructurales aseguran la comprobación de la implementación de una función. Este tipo de pruebas se utiliza generalmente durante la fase de codificación. Se recomienda utilizar el análisis estructural en todas las fases del ciclo de vida en donde el sistema se representa formalmente como un algorítmo, diseño o lenguaje (Perry, 1995).

Prueba dinámica vs estática

Las pruebas tambien se clasifican en dinámicas y estáticas. El análisis dinámico requiere que el sistema se ejecute. Es decir, el programa se corre en algún caso de prueba y se examinan los resultados del desempeño del mismo para verificar si el programa opera como se espera. La prueba dinámica se utiliza generalmente en la fase de prueba del sistema.

El análisis estático normalmente no involucra la ejecución real del programa. Las técnicas comunes del análisis estático incluyen tareas como verificación de la sintaxis. La prueba estática se realiza sin ejecutar los programas operacionales, esto requiere hacer un seguimiento en papel para determinar que los sistemas funcionan apropiadamente cuando se ejecutan. La prueba estática se utiliza generalmente en la fase de requerimientos y de diseño.

Prueba manual vs automatizada

Las pruebas pueden ser ejecutadas de forma manual y/o automáticas. Las pruebas manuales son ejecutadas por las personas, y las pruebas automatizadas por una computadora, mediante herramientas que realizan esta tarea. En los procesos de desarrollo automatizados se recomienda utilizar procesos de pruebas automatizados por su simplicidad. Cuando se cuenta con personal capacitado para analizar, documentar y desarrollar sistemas a mano, se recomienda utilizar la prueba manual (Perry, 1995).

Clasificación de las pruebas

Las pruebas se clasifican en pruebas de bajo nivel (pruebas de unidad, de integración) y de pruebas de alto nivel (pruebas de usabilidad, de sistema, de funcionalidad y de aceptación).

Pruebas de bajo nivel

Las pruebas de bajo nivel involucran las pruebas de módulos individuales del programa, uno por uno o combinados. Requieren un conocimiento de la estructura interior del programa. Los tipos de prueba de bajo nivel son: de unidad y de integración.

Prueba de unidad

La prueba de unidad se define como el proceso de probar los módulos individuales de un programa. Su propósito es descubrir las diferencias entre la especificación de la interface del módulo y su conducta real (Kit, 1996). La prueba de unidad está formada de

combinaciones de varias pruebas. Esto facilita la detección de errores y permite probar varios módulos simultáneamente y en forma independiente.

Prueba de integración

La prueba de integración es el proceso de combinar y probar varios módulos unidos. El objetivo principal de la integración es descubrir los errores en las interfaces entre los módulos. Existen varias alternativas en las pruebas de integración, una de ellas es la integración incremental. En la integración incremental se prueba cada módulo agregándose al conjunto de componentes previamente probados. Hay dos enfoques de la integración incremental: abajo-arriba ("bottom-up") y arriba-abajo ("top–down").

Prueba de alto nivel

La prueba de alto nivel se refiere al conjunto de pruebas que se aplican a un producto en la fase de liberación. Los tipos de prueba de alto nivel son: usabilidad, función, sistema, aceptación. A continuación se describen estas pruebas, sus características, las actividades que se desarrollan en cada una y en algunos casos las categorías en que se dividen.

Prueba de usabilidad

La prueba de usabilidad es el proceso de identificar las diferencias entre las interfaces del usuario de un producto y los requerimientos del cliente (Kit, 1996), la prueba de usabilidad implica tener a los usuarios trabajando con el sistema y observando su reacción ante él,

recopilando información de los problemas específicos que reportan los usuarios, a menudo involucra la evaluación de la presentación de un producto en lugar de su funcionalidad. Los elementos a considerar en esta prueba son:

- **Accesibilidad.** Casos en que los usuarios pueden introducirse, navegar y salirse del sistema.
- **Sensibilidad.** Los usuarios pueden hacer lo que ellos quieren fácilmente y cuando ellos quieren.
- **Eficiencia.** Los usuarios pueden hacer lo que ellos quieren en una mínima cantidad de pasos y tiempo.
- **Comprensibilidad.** Los usuarios entienden la estructura del producto, la ayuda del sistema y la documentación.

Actividades o tareas a seguir durante la ejecución de la prueba:

Definir los objetivos de la prueba y las partes a probarse, planear las pruebas y desarrollar el material, realizar la prueba, anotar los resultados, analizar los resultados y hacer recomendaciones de los cambios

Prueba de función

La prueba de función implica descubrir las diferencias entre la especificación funcional de un sistema y su conducta real, (Kit, 1996). En un sistema cada función tiene una sola especificación definida con claridad. La prueba de función se realiza antes de que el producto esté disponible al cliente y puede empezar a aplicarse siempre que el producto

tenga la funcionalidad suficiente para ejecutar algunas de las pruebas, o después de que las pruebas de unidad e integración han terminado.

Actividades o tareas a seguir durante la prueba de función:

Analizar la especificación del diseño funcional, particionar la funcionalidad en módulos lógicos, hacer una lista de las funciones detalladas, desarrollar casos de prueba que permitan detectar defectos de acuerdo a un conjunto de entradas de datos (ver sección II.5), desarrollar una matriz de clases/funciones (ver apéndice B.1). Esta es una matriz que lista funciones/clases específicas a ser probadas, la prioridad por probar cada función/clase, localización de los casos de prueba que contienen las pruebas para cada función y ejecutar casos de prueba.

Pruebas de sistema

> En las pruebas de sistema se intenta demostrar que el sistema cumple sus requerimientos y objetivos originales como se declaró en el documento de requerimientos. No exite metodologías de diseño para casos de prueba de este tipo, ya que los requerimientos y objetivos no describen las funciones del programa o sistema en términos precisos. Los requerimientos deben ser lo más específicos posible para probarse pero conservando la generalidad para permitir la libertad en el diseño funcional (Kit, 1996).

Como no hay metodología a seguir, las pruebas de sistema requieren creatividad. Se piensa en la perspectiva del usuario y el problema que el usuario está tratando de resolver. Para

realizar las pruebas de sistemas se pueden utilizar varias categorías de casos de prueba de sistema como son:

- **Pruebas de volumen**. Determinan si el programa puede ocuparse de los volúmenes requeridos de datos según la demanda (comparación de datos de transacciones sensibles a la fecha, alimentación del compilador con programas fuentes, unión de programas que contienen miles de módulos, saturación de la cola de un sistema en operación, inundación del correo con mensajes grandes, etc.).

- **Pruebas de cargas de tensión**. Identifican las condiciones de carga máxima en que el programa falla, manejando las cargas del proceso requerido dentro de los tiempos requeridos (introducción de datos en un período de tiempo corto, cargas variantes para tiempo real interactivo y control de procesos, introducción simultánea de un gran número de transacciones, usuarios múltiples accesando al sistema en el mismo minuto, entre otros).

- **Seguridad.** Muestran que los requerimientos de seguridad del programa se pueden infringir (acceso a archivos, cambios en las tablas de seguridad, rechazo de contraseñas inválidas, autorización de contraseñas válidas, función de autentificación, características de seguridad de forma remota, contraseñas múltiples y contraseñas incompletas).

- **Pruebas de desempeño.** Determinan si el programa reúne los requerimientos de desempeño, para los que fué diseñado. Algunos de los atributos que se verifican son: utilización del CPU, número de entradas y salidas por instrucción, utilización del almacenamiento en memoria principal y secundaria, porcentaje del tiempo de ejecución por

módulo, porcentaje de tiempo en almacenamiento, porcentaje de tiempo en espera y número de veces que el control es pasado de un módulo a otro, etc.

- **Pruebas de configuración.** Determinan si el programa opera adecuadamente cuando el software o hardware esta configurado de manera requerida.

- **Pruebas de conversión.** Determinan si se han reunido los objetivos de conversión del programa. Esto es realizar análisis y comparación de datos convertidos, revisar el soporte y la administración de la base de datos, examinación de los datos viejos de inexactitudes o discrepancias y realizar las conversiones durante horas de ausencia de operación, entre otros.

- **Pruebas de compatibilidad**. Determinan si se han reunido los objetivos de compatibilidad. Esto se refiere a la capacidad de operación del sistema con los demás sistemas ya instalados en la computadora.

- **Pruebas de instalabilidad.** Identifican si los procedimientos de instalación llevan a resultados incorrectos. Esto es capacidades técnicas del instalador, documentación de la instalación con pasos específicos y concisos, modificación automática de los archivos de inicio como config.sys etc., establecimiento del ambiente adecuado para la instalación, confirmación de la instalación apropiadamente del sistema.

- **Pruebas de recuperación**. Determinan si el sistema o el programa reúne los requisitos para la recuperación después de una falla. Esto es, restauración de archivos que eran

respaldados durante el mantenimiento, verificación de procedimientos de recuperación manuales y de la seguridad durante la recuperación.

- **Pruebas de confiabilidad/disponibilidad.** Determinan si la operación del sistema está libre de fallas en un lugar determinado y durante un tiempo específico.

Las pruebas del sistema pueden empezar cuando el sistema tiene suficiente funcionalidad o después de que las pruebas de unidad e integración se han completado. Pueden conducirse en paralelo con las pruebas de función.

Actividades o tareas a seguir durante la ejecución de la prueba de sistema:
Descomponer y analizar los requerimientos, particionarlos en categorías lógicas y para cada módulo hacer una lista detallada de las mismas, desarrollar casos de prueba y desarrollar una matriz de seguimiento de requerimientos (ver apendice B.2) y por último ejecutar casos de prueba.

Pruebas de aceptación.

Las pruebas de aceptación implican comparar el producto final a las necesidades actuales del cliente. Normalmente se realizan una vez que se han completado las pruebas de usabilidad, función, y de sistema satisfactoriamente. Estás involucran el funcionamiento y operación del software en un período específico de tiempo que puede durar hasta un año y se efectuan con datos reales, (Sommerville, 1988).

Si el software es desarrollado bajo contrato, las pruebas de aceptación son realizadas por el cliente. El criterio de aceptación se define en el contrato. Si el producto no se desarrolla bajo contrato, la organización puede proponer formas alternativas de prueba de aceptación como son: *alfa* y *beta,* (Kit, 1996). Las pruebas alfa y beta son formas de pruebas de aceptación. Primero es la alfa y luego la beta. Las dos involucran el funcionamiento y operación del sistema en un período preespecificado.

Las pruebas alfa normalmente son realizadas dentro de la compañía desarrolladora, se utiliza el sistema en forma natural con el desarrollador como observador del usuario y registrando los errores y problemas de uso. Las pruebas beta se llevan fuera de la compañía que desarrollo el software. A diferencia de la prueba alfa el desarrollador no está presente. Así la prueba beta es una aplicación en "vivo" en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa al desarrollador. La prueba se dá por terminada al finalizar el período preespecificado por el cliente.

Pruebas alfa y beta. El primer paso en la implementación alfa y beta es definir objetivos primarios de la prueba: pruebas progresivas y/o regresivas. Las pruebas progresivas son el proceso de probar código nuevo para determinar si contiene errores. La prueba regresiva es el proceso de probar el sistema para determinar si un cambio ha introducido errores en el código que se ha modificado.

A continuación se muestra la tabla I, la cual concentra de manera general las características de la clasificación de las pruebas que hemos descrito.

Tabla I. Características generales de los diferentes tipos de pruebas.

Tipos de prueba	**Características generales**				
Unidad	Garantizan que se ejerciten todos los caminos independientes de cada módulo	Prueban todos los bucles en sus límites operacionales	Prueban las estructuras internas de datos para asegurar su validez	Verifican todas las decisiones lógicas en sus vertientes verdadero y falso	Describen discrepancias entre los módulos de la especificación de la interface y su conducta real
Integración	Combinan y prueban componentes múltiples juntos	Descubren errores en las interfaces entre los componentes			
Funcional	Intentan descubrir diferencias entre la especificación funcional del programa y su conducta	Se aplican todos los métodos de caja negra	Se ejecutan antes de que el producto está disponible a los clientes		
Usabilidad	Verifican que se adapte el software al estilo real del usuario en lugar de forzar a los usuarios a adaptar su trabajo al software	Involucran tener a los usuarios trabajando con el sistema y observando sus impresiones al contacto con el mismo	Involucran la evaluación de la presentación de un producto en lugar de funcionalidad	Verifican las diferencias entre las interfaces del usuario y los requisitos del usuario	
Sistema	Intentan demostrar que un programa no encuentra sus requisitos originales y objetivos como se declaró en la especificación de requisitos	Requieren un alto grado de creatividad			
Aceptación	Comparan el producto final con las necesidades actuales de usuarios finales	Se realizan por el cliente	Tienen dos formas alternativas de prueba alfa y beta	Involucran el funcionamiento y operación del software en un período preespecificado	Informan problemas, progreso y estado

Una vez definidas las pruebas se describen en la siguiente sección los mecanismos que se utilizan en el proceso.

Mecanismos de prueba

Los mecanismos de prueba son herramientas que apoyan el proceso de pruebas en diferentes formas ya sea para generar, revisar, analizar, presentar e interpretar los resultados y pueden ser utilizados durante el ciclo de vida de sistemas (Perry, 1995). A continuación se muestran brevemente algunos de los muchos que existen. El objetivo de dar una idea de la aplicación que tienen en las pruebas de sistemas.

- **Criterio de aceptación.** Se elabora un estándar de aceptación antes de que el sistema sea aceptado por el cliente, este puede estar establecido por el contrato ó documento de requerimientos.

- **Partición de equivalencia.** Se seleccionan datos de entrada representativos de un dominio, se ejecutan en la computadora y el resultado sirve para predecir el comportamiento de otros datos similares. Si una condición de entrada acepta números enteros positivos, se debe diseñar un caso de prueba con números negativos como entrada donde el valor representativo sea un entero negativo y un caso de prueba para fracciones donde el valor representativo sea una fracción positiva y/ó negativa (Lewys, 2000).

- **Análisis de valores límite.** Los errores frecuentemente se encuentran más en los límites del campo de entrada que en el centro, por lo que se deben elaborar casos de prueba en los límites del campo de entrada. Si una condición de entrada especifica un intervalo de

valores, se deben diseñar casos de prueba por debajo y por encima de los límites respectivos (Pressman, 1999).

- **Listas de defectos**. Son una serie de preguntas diseñadas para utilizarse en revisiones a una función o a una área predeterminada del sistema (ver apéndice C).

- **Comparación de código.** Se utilizan las mismas especificaciones para desarrollar versiones independientes de una aplicación, se prueba cada versión con los mismos datos, se ejecutan todas las versiones en paralelo y se hace una comparación en tiempo real de los resultados, para garantizar la consistencia (Perry, 1995).

- **Comparación de archivos.** Es un instrumento de software de propósito general que informa de las diferencias entre archivos. Un archivo contiene el resultado esperado de un programa. Se ejecuta la prueba y la salida real se envía a otro archivo. Posteriormente se comparan los archivos y se destacan las diferencias entre ellos, (Sommerville,1988).

- **Complejidad ciclomática.** Este mecanismo está basado en la teoría de grafos y nos proporciona una medición cuantitativa de la complejidad lógica de un programa, que se puede calcular de las siguientes formas:

a) El total de regiones del grafo de flujo donde la región es un área delimitada por aristas y nodos.
b) El número de aristas menos el número de grafos más dos.
c) El número de nodos predicado más uno. Un nodo predicado es aquel que contiene una condición y está caracterizado porque dos o más aristas emergen de él.

Este valor, obtenido de la medición, define el número de caminos independientes de un programa y nos dá un límite superior para la cantidad de pruebas que se deben realizar para asegurar que se ejecuta cada línea de código.

- **Análisis del Flujo de Datos.** Método para asegurar que los datos utilizados por el programa se han definido y se utilizan adecuadamente. Este mecanismo sirve para detectar inconsistencias en los sistemas tal como variables no definidas o no referenciadas. El análisis del flujo de datos se realiza utilizando el compilador (Perry, 1995).

- **Análisis de flujo de control.** Requiere de una representación gráfica del programa, para analizar el flujo y la lógica dentro del mismo, para identificar los problemas lógicos (Perry, 1995). Se pueden utilizar diagramas de flujo, diagramas de Estados, diagramas OMT, diagramas de eventos entre otros.

- **Diccionario de Datos.** Es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, componentes de los almacenes y si existen, de los cálculos intermedios. El diccionario de datos se utiliza como una base de datos, el analista puede hacer preguntas como ¿dónde se usa?, ¿cómo se usa? y obtener respuestas (Pressman, 1999). Además, sirve para probar la validación de elementos de datos definidos con los datos de prueba a través de los atributos definidos para cada uno.

- **Generador de datos para prueba.** Programas que generan automáticamente una gran cantidad de entradas de prueba para un sistema, de esta forma se podrá probar el rendimiento de este último en un ambiente realista (Sommerville,1988).

- **Inspección.** Tipo de revisión formal a los documentos que se producen en cada fase del ciclo de vida del sistema, que sirven para identificar defectos potenciales. Las inspecciones requieren la realización de una lista de criterios predeterminada que sirva de base para la misma (Lewys,2000).

- **Revisión de escritorio.** Revisiones informales realizadas por el desarrollador de un documento (requerimientos, diseño, módulo de código etc.) durante la fase de desarrollo de software con el objetivo de detectar defectos en el mismo.

- **Revisión por pares.** Proceso de revisión de pares de usuarios que revisan el producto de un desarrollador y discuten puntos relativos al mismo con el objetivo de obtener productos de mejor calidad y de acuerdo a las necesidades del cliente.

- **Revisiones del diseño.** El objetivo primario de la revisión del diseño es asegurar que este satisface la metodología establecida. Aquí se revisan los documentos de diseño de alto y bajo nivel.

- **Funcionamiento paralelo.** Es la ejecución de la versión anterior y actual del sistema al mismo tiempo para identificar las diferencias entre las dos versiones. El método es muy eficaz cuando hay un número mínimo de cambios entre las versiones del proceso.

- **Instrumentación.** Uso de monitores y/o contadores para determinar la frecuencia en la cual ocurren eventos predeterminados (Perry, 1995).

- **Facilidad de la prueba integrada.** Este mecanismo permite la introducción de datos de prueba en un ambiente para que puedan probarse otras aplicaciones al mismo tiempo que

se esta ejecutando y la acumulación de resultados encima de muchas iteraciones (Perry,1995).

- **Organigrama.** Es la representación gráfica del flujo del sistema que muestra los requerimientos y diseño del mismo. Los organigramas permiten entender y valorar la estructura y lógica de un sistema desde otro enfoque diferente a la expresión verbal (Pressman, 1999).

- **Transformación.** Proceso en el que se analizan las partes de un sistema que son ejecutadas durante la prueba y la frecuencia de ejecución de cada declaración, clase, rutina o función. Puede usarse para detectar defectos en el sistema, para determinar que porcentaje de un programa se ejecuta durante las pruebas, y para identificar áreas dónde el código es más eficaz y poder reducir el tiempo de la ejecución (Perry, 1995).

- **Trazado.** Representación de los caminos seguidos en una base de datos para localizar una o más piezas de información utilizadas para producir un registro lógico de procesamiento (Lewys, 2000).

- **Matriz de riesgos.** Establece el uso de controles a través de la identificación de riesgos y controles llevados a cabo en cada parte del sistema reduciendo los riesgos a un nivel aceptable al usuario (ver apéndice B.3).

- **Foto instantánea "snapshot".** Método para imprimir el estado de la memoria de la computadora en puntos predeterminados durante el proceso de pruebas. El cual puede imprimirse cuando se ejecutan instrucciones ó se procesan datos con atributos específicos.

Al conocer estos datos el ingeniero de pruebas tiene elementos suficientes para evaluar y tomar deciciones que le permitan monitorear el proceso (Perry, 1995).

- **Simuladores.** Son programas que imitan las acciones de otro programa, dispositivo de hardware o clase de dispositivos. Los simuladores se usan a menudo en pruebas para reemplazar un hardware que se espera, pero que no está disponible de inmediato (Sommerville, 1988).

- **Prueba exhaustiva.** Consiste en realizar todas las pruebas necesarias para evaluar cada camino y condición posible en el sistema .

- **Prueba del desastre.** Procedimiento en el que se simula un desastre con el objetivo de poner en acción los procedimientos y el entrenamiento de la recuperación de la falla.

La tabla II muestra los mecanismos de prueba que se describieron y que se proponen utilizar para probar el software según la fase del ciclo de desarrollo en que se encuentre el sistema; esta tabla permitirá seleccionar aquellos mecanismos que el ingeniero de pruebas considere más conveniente utilizar de acuerdo al tipo de software que esté desarrollando, extensión del mismo, esto decir su tamaño y el tiempo estimado para realizar las pruebas .

Tabla II. Relación de los mecanismos de pruebas con las fases del ciclo de vida del software.

Mecanismos de prueba	**Fases del ciclo de vida**			
	Análisis Reque / Espec	**Diseño Alto / Bajo**	**Implementación**	**Prueba**
Criterio de aceptación			x	
Partición de equivalencia		x	x	
Análisis de valores límite		x	x	
Listas de defectos	x	x	x	x

Comparación de código				x
Comparación de archivos				x
Complejidad ciclomática			x	
Análisis del flujo de datos			x	
Análisis del flujo de control			x	
Diccionario de datos	x			x
Generador de datos de prueba				x
Inspección	x	x	x	x
Revisión de escritorio	x	x	x	
Revisión por pares	x	x	x	x
Revisiones de diseño		x		
Funcionamiento paralelo				x
Instrumentación				x
Facilidad de prueba integrada				x
Organigrama	x	x	x	
Transformación			x	
Trazado			x	
Matriz de riesgos	x	x		
Foto instantánea "snapshot"				x
Simuladores				x
Prueba exhaustiva			x	
Prueba desastre				x

Además de los mecanismos mencionados anteriormente se resalta la existencia de los casos de prueba ya que son primordiales y no se deben excluir al realizar las pruebas de sistemas. En la siguiente sección se explica a detalle en que consiste este mecanismo tan importante.

Casos de prueba

Un caso de prueba es el conjunto de condiciones de prueba, incluyendo objetivos, ambiente, entradas de datos, resultados actuales y esperados, entre otros, (Lewis,2000).

Algunas de las ventajas de utilizar casos de prueba son:

- **Acercamiento al sistema en ejecución**. En las especificaciones o en el diseño funcional no se tratan las particularidades del software a desarrollar, los casos de prueba consideran datos de prueba ó corridas que no se habian considerado en ningún documento.

- **Casos de prueba reusables.** El escenario de prueba se repite para reiterar que no se encuentren los mismos errores cada vez que se introduzca un nuevo cambio.
- **Ejecución indistinta**. Un caso de prueba bien documentado puede ser ejecutado por cualquier persona. Los casos de prueba permiten estimar la calidad del software diseñado, los errores encontrados en un porcentaje de casos de prueba ejecutados permite hacer una estimación del total de errores en el sistema.

Los casos de pruebas se dividen en dos tipos de caja blanca y caja negra. A continuación se explica en detalle cada uno de ellos.

Caja blanca

La prueba de caja blanca se basa en el minusioso examen de los detalles procedurales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado, (Pressman, 1999).

Mediante pruebas de caja blanca, el ingeniero de pruebas de sistema puede obtener casos en donde se:

- Garantize que se ejercita por lo menos una vez todos los caminos independientes de cada módulo
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa
- Ejecuten todos los lazos en sus límites y con sus límites operacionales
- Ejerciten las estructuras internas de datos para asegurar su validez.

Caja Negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del sistema y se centran en los requerimientos funcionales del mismo. Es decir, permite al ingeniero de sistemas obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales de un programa (Pressman, 1999).

La prueba de caja negra pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, y se mantiene la integridad de la información externa. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software

La prueba de caja negra es un enfoque diferente al de caja blanca que intenta descubrir otros tipos de errores.

Según Pressman (1999) los tipos de errores que intenta descubrir esta prueba son: funciones incorrectas o ausentes, errores de interfaz, errores en las estructuras de datos o en accesos a bases de datos externas, errores de rendimiento, errores de inicialización y de terminación.

En este capítulo se habló sobre las pruebas que existen actualmente para probar sistemas de software, sus principales características y las actividades que se realizan. Además, se mencionó la existencia de mecanismos que se requieren durante la aplicación de las mismas y las fases del ciclo de desarrollo en donde se pueden utilizar. Por último se resalta la importancia que tienen los casos de prueba como elemento primordial del proceso de prueba.

En base a la información se propone una guía de pruebas práctica cuyo objetivo es evaluar sistemas de software durante las fases de desarrollo. Esta guía constituye la principal aportación de este trabajo de tesis, esta permitirá llevar paso a paso a los ingenieros de pruebas en la realización de las pruebas a los diferentes sistemas de software que implementen. En el siguiente capítulo se describe la guía propuesta.

Guía de pruebas

Introducción

En este capítulo se presenta la propuesta de una guía que describe el proceso de pruebas de sistemas de software para ser aplicada durante las fases de su ciclo de vida.

Una guía es un procedimiento empleado para controlar o dirigir la realización de alguna actividad ó proceso. Estas sirven para plasmar claramente ¿qué se necesita?, ¿qué se va ha hacer?, ¿cómo se va ha hacer?, ¿para qué?, entre otras cosas. En este trabajo se establece una guía que define el proceso de pruebas para la valoración de sistemas de software. Esta se divide en cinco etapas: establecimiento del plan de pruebas, aplicación de las pruebas, reportes, seguimiento de la prueba y aplicación de métricas. Enseguida se describen éstas en más detalle.

1) Establecimiento del plan de pruebas; consiste en describir el sistema bajo prueba, las necesidades de personal y los recursos de hardware y software que se requieren para llevar a cabo el proceso de pruebas. El objetivo es establecer los requisitos necesarios para empezar y culminar las pruebas eficientemente.

2) Aplicación de las pruebas; esta etapa consiste en describir las pruebas que se van a aplicar al sistema para facilitar la implementación de las mismas, enumerando los casos de prueba de cada una y el procedimiento a seguir para ejecutarlas.

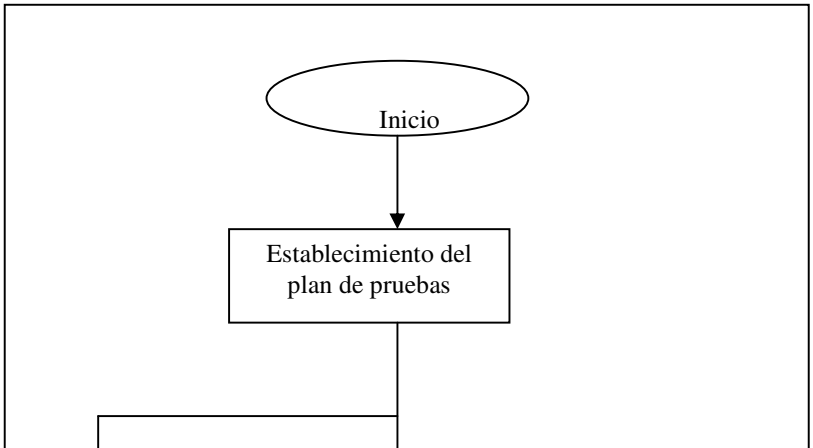
3) Reportes; etapa que consiste en registrar los resultados de las pruebas en cuatro reportes diferentes: bitácora, reporte de incidentes, reporte de transmisión y resumen de reportes

de las pruebas. Estos documentos recaban información del ambiente en que se desarrollan las pruebas. Los dos primeros registran los sucesos relevantes durante la ejecución de las pruebas, el tercero describe los elementos que se probaron y el último reporta el concentrado de los resultados de las actividades de pruebas y la evaluación de los resultados.

4) Seguimiento de la prueba; consiste en revisar y analizar los reportes generados en la etapa anterior para medir el avance en la ejecución de las pruebas y hacer estimaciones del trabajo pendiente permitiendo tomar acciones necesarias para terminar en las fechas previstas.

5) Aplicación de métricas; se utilizan para realizar el análisis de los resultados de las pruebas cuyos objetivos principales son evaluar la prueba y dar un diagnóstico del sistema que se está evaluando.

La figura 1 muestra gráficamente las etapas del proceso de pruebas de la guía propuesta.



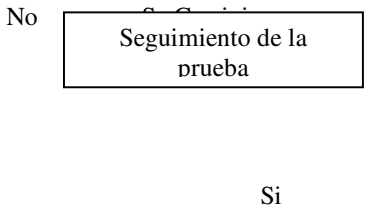


Figura 1. Diagrama con las etapas del proceso de pruebas de la guía propuesta

Como se observa el proceso planteado en la guía propuesta inicia con el establecimiento del plan en donde se describen los requerimientos necesarios para empezar las pruebas, a continuación se realiza la aplicación en donde se obtienen diseños, casos y procedimientos

de prueba, posteriormente se ejecuta la prueba registrando los defectos en reportes, conforme se van desarrollando se mide el avance de la ejecución de las pruebas revisando y analizando los mismos, se hacen estimaciones del trabajo pendiente, como pruebas por realizar y defectos por corregir. Una vez terminadas las correcciones se analizan los resultados del proceso mediante la apliación de métricas y se hace el diagnóstico del sistema que se está evaluando.

Propósito de la guía de pruebas

El propósito de la guía es apoyar la realización de pruebas a sistemas de software durante el ciclo de vida de su desarrollo. Para ello se debe familiarizar al usuario con el proceso de pruebas describiendo los elementos del mismo y los documentos involucrados, de manera que le facilite la creación, ejecución, seguimiento y evaluación de las pruebas que realice.

Alcances de la guía de pruebas

Este documento está dirigido a usuarios con conocimientos en el área de Computación ó áreas afines al desarrollo de sistemas, que deseen aprender a probar sistemas de software de manera rápida y eficiente de acuerdo a sus necesidades.

La guía de pruebas presenta una descripción en forma individual de las etapas que se compone especificando sus perspectivas, características y restricciones de cada una, posteriormente muestra el contenido de los documentos que contiene cada una. El propósito de cubrir estos puntos es explicar claramente ¿qué se hace?, ¿que características cubre cada etapa? y ¿qué se va a obtener? para facilitar la comprensibilidad de las mismas.

Establecimiento del plan de prueba

Introducción

El plan de prueba es un documento que permite conducir el proceso de pruebas de sistemas cubriendo todas las fases del ciclo de desarrollo de software. En el plan de prueba se muestra información sobre ¿Qué se va hacer?, ¿Quién lo va ha hacer?, ¿Cómo se va hacer (orden cronológico)?, ¿Donde ? y ¿Para qué?.

Un plan de prueba permite que se mantengan ideas, pensamientos y experiencia de los integrantes del grupo sobre los problemas que se presentan durante su implementación y plasmar ese conocimiento de manera formal para que les sirva de antecedente para solucionar problemas futuros (Black, 1999).

Objetivos

El objetivo general del uso de un plan de pruebas es:

Iniciar y concluir satisfactoriamente las pruebas para cada una de las fases del proceso de desarrollo de software.

Los objetivos particulares del plan de pruebas son:

- *Preparar pruebas para cada fase del proceso de desarrollo de software.*
- *Especificar los mecanismos de pruebas que se aplican en cada fase del desarrollo del software (Black, 1999).*

Perspectiva del plan de pruebas

El usuario debe realizar un plan de pruebas que abarque todas las fases del ciclo de vida del desarrollo de software, desde la fase de requerimientos hasta la entrega al usuario final. En general, se espera que los usuarios manejen en forma eficiente el proceso de pruebas independientemente de la forma en la cual lo desarrollen. En la práctica, la administración del proceso de pruebas puede variar dependiendo del sistema a probar ya sea administrativo, científico, general, etc. Cuando se desarrolla un plan de pruebas no se debe descuidar el propósito del sistema para el cual se está realizando, por lo que el esfuerzo de pruebas se debe enfocar en probar, en primera instancia, aquellos módulos que le den la funcionalidad requerida al software.

Características del plan de pruebas

El plan de pruebas debe tener las siguientes características:

- **Consistencia**. Conserva un diseño uniforme de los formatos.
- **Expansibilidad**. Incorpora documentos, casos de prueba o anexos tantos como se requieran.
- **Estandarización**. Cumple con los formatos establecidos por la oficina nacional de estandares y la IEEE (estándares: 829-1998, 1008-1987(R1993) y 1012-1998).

- **Flexibilidad**. Se adapta a las necesidades del sistema a probar sin afectar el proceso de desarrollo de software.
- **Generalidad**. Aplicable a cualquier tipo de sistema.
- **Simplicidad**. Facilita el proceso de pruebas.
- **Trazabilidad**. Facilita el seguimiento del sistema desde la representación de los requerimientos hasta el diseño o un componente real del sistema.

Restricciones

El plan de pruebas recaba información general sobre el software, las pruebas a realizarse, el ambiente requerido y el calendario de actividades a seguir. La claridad de la información es fundamental para continuar con el proceso de pruebas y llegar a resultados satisfactorios.

Contenido del plan de pruebas

El contenido del plan de pruebas planteado recaba información necesaria para llevar a cabo el proceso de pruebas. Este documento deberá contener los siguiente puntos:

- **Identificador del plan de prueba.** El identificador es un número alfanumérico que se obtiene de acuerdo a lo especificado en la administración de configuración.

- **Introducción.** En la introducción se deberá dar un resumen de los antecedentes del sistema bajo prueba, se debe describir el enfoque de la prueba y hacer referencia a documentos relacionados con la misma, entre los cuales se encuentran:

a) Autorización del proyecto

b) Plan del proyecto

c) Plan de aseguramiento de calidad

d) Plan de administración de configuración

e) Políticas relevantes

f) Estándares relevantes

- **Objetivos.** Los objetivos son las metas que se pretenden alcanzar al realizar las pruebas. En este punto se describen brevemente y se debe dar una explicación del mismo.

- **Descripción del Sistema.** En la descripción se deben describir la funcionalidad del sistema valiéndose de figuras, gráficas, entre otros. Se debe dar un panorama de lo que hace y no hace el sistema en ejecución.

- **Elementos de la prueba.** Elementos de la prueba son los elementos que están bajo prueba. Los elementos se listan y se hace referencia a los documentos del proyecto, si se tienen. Estos documentos pueden ser: requerimientos, especificación de requerimientos, diseño (alto y bajo nivel) , guía de usuario, guía de operación y guía de instalación. Los elementos son diferentes en cada etapa. Los elementos del análisis son la descripción de requerimiento y su especificación y en el diseño son la arquitectura, subsistemas, base de datos y el modelado.

- **Descripción de la Prueba.** En la descripción se listan las pruebas a realizarce y se asocian cada una de ellas a una prueba. Otra alternativa es realizar un plan de pruebas a cada prueba considerada. Una descripción de las pruebas se lista en el capitulo II sección II.3.

- **Documentación del sistema.** Son los documentos utilizados para el desarrollo del plan de pruebas, entre los cuales tenemos el análisis y diseño del sistema, guía de usuario, de operación y de instalación

- **Documentos de la prueba.** Son los documentos que resultan de aplicar el plan de pruebas. Estos documentos son los siguientes:

a) Plan de pruebas
b) Diseño de pruebas
c) Casos de prueba
d) Procedimientos de pruebas
e) Bitácoras de pruebas
f) Reportes de incidentes de pruebas

- **Mecanismos de prueba.** Son las herramientas que asegurarán que las pruebas se realicen adecuadamente. Una descripción detallada se lista en el capitulo II sección II.4.

- **Presupuesto.** Son los recursos económicos que se requieren para realizar el plan de pruebas.

- **Necesidades de personal y entrenamiento.** Se refiere al grupo necesario para realizar las pruebas y las habilidades que deben tener. Se lista el personal y se identifican las necesidades de entrenamiento.

- **Responsabilidades**. Responsabilidades son las actividades de administrar, diseñar, preparar, ejecutar y evaluar las pruebas. Se definen los grupos responsables de proporcionar

la documentación, el ambiente de prueba y de realizar las actividades mencionadas previamente.

- **Requisitos de software y hardware.** Es la infraestructura que se utiliza para realizar la prueba. Se lista el hardware, software, redes, espacio de laboratorio y cualquier otro software o suministros necesarios para apoyar la prueba.

- **Calendario.** En el Calendario se listan las actividades y el tiempo requerido para realizar las pruebas. Se especifica el periodo que utilizará cada recurso de prueba (infraestructura, mecanismos de prueba y personal).

- **Aislamiento del error y clasificación.** Grado en que se aíslan los errores y la clasificación de los mismos. Aislar el error significa experimentar con el sistema bajo prueba en un esfuerzo por encontrar la raíz del error. Se realiza una tabla en donde se especifican los tipos de error que pueden ser de conectividad, consistencia, integridad, documentación, interfaz, instalación, seguridad, entre otros y su tratamiento.

- **Administración de configuración.** Son los criterios que se establecen para definir los identificadores de los documentos de la guía de pruebas, los cuales sirven para administrar el proceso de pruebas. Se propone utilizar un identificador alfanumérico que se componga de cuatro pares de números. El primer par se utiliza para reconocer e identificar el sistema de software a evaluar, el segundo se requiere para identificar los documentos de las tres primeras etapas de la guía de pruebas que son establececimiento del plan de pruebas, aplicación y reportes, el tercero sirve para enumerar los documentos del mismo tipo de las etapas mencionadas en el segundo par de números por ejemplo si son tres documentos de

diseño les corresponderían los números 01,02 y 03 respectivamente. Por ultimo, el par cuatro sirve para especificar la versión de los documentos.

Los identificadores de los documentos (segundo par de números) son los siguientes:

01 Plan de prueba.

02 Diseños de prueba.

03 Casos de prueba.

04 Procedimientos de prueba.

05 Bitácoras de prueba.

06 Reportes de incidentes de prueba.

07 Resumenes de reportes de prueba.

08 Reportes de transmisión.

Por ejemplo, en el número SN020301 los pares de números se descifran de la siguiente manera:

SN. Sistema denominado "Software Nuevo".

02. Tipo de documento diseño de pruebas.

03. Tercero de los documentos de diseños de pruebas.

01. Primera versión del tercero de los diseños de pruebas.

- **Criterios de suspensión y requerimientos de reanudación.** Criterios de suspensión y requerimientos de reanudación son las condiciones que se establecen para suspender y/o

reanudar parcial ó totalmente las actividades asociadas con este plan. Especificar las actividades de pruebas que deben repetirse, cuando las pruebas sean reinicializadas.

- **Riesgos y contingencias.** Los riesgos son los eventos y/o acontecimientos que afectan la ejecución del plan de prueba. Se realiza una tabla que contenga los riesgos y planes de contingencia a dicho riesgo.

Con este punto finalizamos la primera etapa de la guía de pruebas que corresponde al establecimiento del plan de pruebas, en la siguiente etapa se explica la manera en que se definen los documentos que se utilizarán para diseñar y ejecutar las pruebas que se necesitan para probar el sistema.

Aplicación de la prueba

Los documentos muy grandes son complejos, poco manejables e incomprensibles, por esta razón no es conveniente que el plan de pruebas concentre toda la información, por lo que se requiere el apoyo de otros documentos. A partir del enfoque del plan de pruebas, se realizan diseños de pruebas. A su vez, por cada diseño se realizan un conjunto de casos de prueba. Por último, se diseñan procedimientos de prueba para plasmar el proceso a seguir al llevar a cabo la ejecución de los casos de prueba. Los documentos que apoyan la aplicación de la prueba son: diseños, casos y procedimientos de prueba.

Perspectivas de la aplicación de la prueba

El usuario podrá desglosar el plan de pruebas realizando diseños de pruebas, casos de prueba y procedimientos de prueba. El diseño de estos documentos elimina la especulación del usuario sobre el contenido de los mismos, de manera que traspase su información fácilmente.

Características de la aplicación de la prueba

La aplicación de las pruebas debe ser clara, consistente y comprensible, definiéndose como:

Claridad. La eliminación de la ambiguedad en los formatos.

Consistencia. Que emplea de un diseño uniforme en los formatos.

Comprensibilidad. Que facilita el entendimiento del usuario con formatos sencillos y fáciles de llenar.

Restricciones de la aplicación de la prueba

La aplicación de la prueba proporciona información sobre las características que se van a probar del software, los casos de prueba y los procedimientos necesarios para ejecutarlos.

En las siguientes secciones se describirán los diferentes documentos de la aplicación de la prueba como son diseño de las pruebas, casos de pruebas y procedimientos de prueba.

Diseño de las pruebas

El diseño de las pruebas es el documento que desglosa en forma detallada la funcionalidad de una parte específica del sistema bajo prueba, implica identificar un conjunto de casos de

prueba relacionados con el objetivo de observar el desempeño del mismo. Los puntos que se consideran en el diseño de las pruebas son los siguientes:

- **Identificación del diseño de las pruebas.** Aqui se especifica el identificador alfanumérico que corresponde a este diseño de las pruebas obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas.

- **Requerimientos funcionales.** Se mencionan los requerimientos funcionales listados en el análisis del sistema.

- **Funciones estructurales.** Se describe con detalle las funciones estructurales a ejercitarse durante la ejecución de las pruebas.

- **Características a probar.** Se describen las características a probar que corresponden al diseño de las pruebas. Se incluye la referencia a los requerimientos asociados en la especificación de requerimientos o en el diseño.

- **Enfoque.** En el enfoque se describe la metodología que asegurará que las características se prueban adecuadamente. Por cada característica listada se debe especificar el método para analizar los resultados de la prueba.

- **Identificación de la prueba.** Identificación de la prueba es el conjunto de casos de prueba que pertenecen al diseño de la misma. Aquí se listan los identificadores y una breve descripción de cada caso de prueba y procedimiento asociado con el diseño.

- **Criterio de éxito/fracaso.** Criterios de éxito o fracaso son las condiciones que se establecen para que las pruebas sean aceptadas o rechazadas al realizar la misma.

Caso de prueba

Un caso de prueba es un documento que permite establecer condiciones de prueba incluyendo entradas de datos, resultados esperados y actuales, entre otros, para garantizar que el sistema responde como se espera a entradas válidas e inválidas y que no corrompe otras partes del sistema. Los puntos a considerar en el caso de prueba se mencionan a continuación:

- **Identificación del caso de prueba.** Aqui se especifica el identificador alfanumérico que corresponde a este caso de prueba obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas.

- **Descripción del caso de Prueba.** Se describe brevemente el caso de prueba y las características que se van a probar en el mismo. Se debe hacer referencia a la siguiente documentación: especificación de requerimientos, diseño, guía de usuarios, guía de operación y guía de instalación.

- **Especificaciones de entrada.** Especificación de la entrada son los datos que se requieren para ejecutar el caso de la prueba. Estas se especifican por valor y por nombre cuando se trate de tablas constantes o transacciones de archivos. Adicionalmente, se identifican bases de datos, archivos, mensajes y valores requeridos.

- **Especificaciones de salida.** Especificación de la salida son los datos que se esperan como resultado de ejecutar el caso de prueba. Se debe especificar el valor exacto de cada salida esperada.

- **Requerimientos de procedimientos especiales.** Aquí se describe cualquier restricción sobre los procedimientos de prueba que ejecutan este caso de prueba. Estas restricciones pueden involucrar instalación especial, intervención del operador y procedimientos de determinación de salidas.

- **Dependencias entre casos.** Es la lista de los identificadores de casos de prueba que deben ejecutarse según las prioridades a este caso.

Procedimiento de prueba

Un procedimiento de pruebas es un documento que permite especificar la secuencia de pasos a seguir para iniciar, ejecutar, suspender, terminar, restaurar y parar la ejecución de los casos de prueba. Los puntos que contiene el procedimiento de pruebas son los siguientes:

- **Identificación del procedimiento de prueba.** Aqui se especifica el identificador alfanumérico que corresponde a este procedimiento de prueba obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas. Proporciona la referencia al plan de prueba asociado.

- **Propósito.** Se describe el propósito del procedimiento y se listan los casos de prueba relacionados al procedimiento.

- **Requerimientos especiales.** Requerimientos especiales son cualquier requisito que sea necesario para la ejecución del procedimiento entre los cuales tenemos requerimientos previos, de habilidades especiales y de ambientales especiales.

- **Pasos del procedimiento de pruebas**

Los pasos del procedimiento de pruebas se describen enseguida:

a) Bitácora. Se registra el identificador de la bitácora en donde se regitran los resultados de ejecución de la prueba.

b) Inicio. Preparar el ambiente y empezar la ejecución de la prueba.

c) Ejecución. Llevar a cabo los casos de prueba.

d) Apagado. Suspender las pruebas, cuando se presenten eventos no previstos.

e) Parar. Dar por terminada la ejecución de la prueba.

f) Reinicialización. Restaurar el ambiente de prueba.

Al terminar la definición de estos documentos continuamos con la siguiente etapa que consiste en realizar reportes de pruebas, la cual registra la información concerniente a la aplicación de la prueba.

Realizar reportes de pruebas

Los reportes de pruebas son el conjunto de documentos que se utilizan para registrar los sucesos relevantes, percances y resultados de la prueba. Estos son: bitácora de prueba, reporte de incidentes, reporte de transmisión y resumen de los reportes de pruebas. Primeramente se generan bitácoras de prueba que describan paso a paso la ejecución de las actividades. A su vez se realizan reportes de incidentes conforme se van presentando

acontecimientos o eventos inesperados. Enseguida se realiza el resumen de reportes de la prueba en el cual se concentran los resultados obtenidos tales como: resultados, varianzas, evaluación y resumen de actividades de pruebas. Por último se realiza el reporte de transmisión para informar sobre el estado de los elementos. En las siguientes secciones se describen a detalle estos documentos.

Perspectivas de los reportes de pruebas

El usuario deberá registrar los resultados de la aplicación del plan de pruebas en cuatro tipos de reportes: bitácora de prueba, reporte de incidentes, reporte de trasmisión y resumen de los reportes de las pruebas.

Características de los reportes de pruebas

Las características que tienen estos reportes son claridad, consistencia, flexibilidad y generalidad. Entendiéndose como:

Claridad. La eliminación de la ambigüedad de los formatos.

Consistencia. El empleo de un diseño uniforme en los formatos.

Flexibilidad. Que se adecue a las necesidades del usuario.

Generalidad. Que se pueda utilizar en las pruebas de diferentes sistemas.

Restricciones de los reportes de pruebas

Los reportes de prueba dependen directamente de la disponibilidad de los documentos de diseño de las pruebas, casos de pruebas y procedimientos de prueba. Sólo a partir de la

ejecución del proceso de prueba, se proporciona información sobre el desarrollo de las actividades, incidentes presentados durante la prueba, errores encontrados y los resultados de la prueba.

En las siguientes secciones se describirán los diferentes tipos de reportes como son bitácora de pruebas, reporte de incidentes, reporte de trasmisión y resumen de los reportes de las pruebas.

Bitácora de prueba

La bitácora de prueba es el documento que recaba información acerca del ambiente en que ocurren las pruebas, personal encargado y el calendario de actividades del día. Incluye la fecha y hora en que inicia y termina cada actividad y los incidentes que se presenten durante la misma. Los puntos que contiene este documento son los siguientes:

- **Identificador de la bitácora de prueba.** Aqui se especifica el identificador alfanumérico que corresponde a esta bitácora de prueba obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas

- **Descripción.** La descripción abarca el contexto en el que se realizan las pruebas. Identifica los elementos a probar y una referencia a su reporte de transmisión, si existe. Identifica los atributos del ambiente en que se realiza la prueba incluyendo la infraestructura, hardware (cantidad de memoria, número del modelo del cpu, número y modelo de los manejadores, y/o dispositivos de almacenamiento en masa), el sistema de software utilizado, y recursos disponibles.

- **Actividades**. Son las tareas que involucran la ejecución de la prueba. Se realiza una tabla que contenga la planeación del calendario de actividades de prueba que incluye el día y hora en que ocurre, el tiempo requerido a lo largo de todo el proceso y los datos del autor.

- **Descripción de la ejecución.** Especifica el identificador del procedimiento de prueba a ejecutar y el personal involucrado durante la ejecución de la misma incluso los probadores, operadores y observadores.

- **Resultados del procedimiento.** Los resultados del procedimiento son las observaciones obtenidas al ejecutar el procedimiento de prueba. Se toman en cuenta mensajes de error generados, abortos, y respuestas del software.
- **Identificadores de reportes de incidentes.** Incidentes son los eventos no calendarizados que afectaron la ejecución de la prueba. Aquí se listan los identificadores de los reportes de incidentes de prueba.

Reporte de incidentes de prueba

El reporte de incidentes es un documento que describe a detalle los incidentes que se mencionan en la bitácora de pruebas y el impacto que tuvieron es decir, a que grado afectó la ejecución de la pruebas. Los puntos que contiene el reporte de incidentes son los siguientes:

- **Identificación del reporte de incidente de prueba.** Aqui se especifica el identificador alfanumérico que corresponde a este incidente de prueba obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas

- **Resumen.** Resumir brevemente el incidente. Identificar los elementos de la prueba involucrados. Hacer referencia a procedimientos de prueba, casos de prueba y bitácora de prueba.

- **Descripción del incidente**. Se proporciona una descripción del incidente. La descripción debe incluir los siguientes puntos: resultados esperados, actuales, anomalías, día y hora, paso en el procedimiento, ambiente, intentos repetidos, probadores, observadores.

- **Impacto.** Impacto es el grado en que afecta la continuidad de la ejecución de la prueba. Describe el impacto del incidente en los planes de prueba y procedimientos de prueba o de casos de prueba.

Reporte de transmisión

El reporte de transmisión es el documento que describe los elementos y documentos que se afectaron durante la ejecución de las pruebas y que van a ser transmitidos al equipo de desarrollo. A continuación se muestran los puntos que contiene el reporte de transmisión:

- **Identificación del reporte de transmisión.** Aqui se especifica el identificador alfanumérico que corresponde a este reporte de transmisión obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas

- **Elementos transmitidos.** Elementos transmitidos son los elementos y documentos afectados al realizar las pruebas. Indica las personas responsables de los elementos

transmitidos. Proporciona la referencia a la documentación del proyecto y al plan de pruebas.

- **Localización.** Se designa el lugar en donde se almacenan los elementos trasmitidos, y se listan los dispositivos que los contienen.

- **Estado.** Describe el estado de los elementos que se transmiten. Se deben incluir las desviaciones de la documentación, transmisiones previas de estos elementos y del plan de prueba, listar los reportes de incidentes que se espera que sean resueltos por los elementos transmitidos e indicar si hay modificaciones pendientes de la documentación que afecta a los elementos listados en el reporte de transmisión.

Resumen de reportes de las pruebas

El resumen de reportes de las pruebas concentra la información de la ejecución de las pruebas a una fecha determinada. En este se resumen las pruebas que se realizaron, el avance en el calendario global, los incidentes que se presentaron y se evaluan los resultados de las pruebas. Los puntos que contiene el resumen de reportes de las pruebas son los siguientes:

- **Identificación del reporte del resumen de las pruebas.** Aqui se especifica el identificador alfanumérico que corresponde a este resumen de reporte de las pruebas obtenido de acuerdo a lo estipulado en la administración de configuración del plan de pruebas.

- **Resumen.** Se resumen las pruebas que se evaluan. Se identifican las pruebas y el ambiente en el cual las actividades de pruebas toman lugar. Se deben referenciar los siguientes documentos: plan de prueba, especificaciones del diseño de prueba, especificaciones de procedimientos de prueba, reportes de transmisión, bitácoras de prueba, y reportes de incidentes de prueba.

- **Varianzas.** Se reportan las varianzas de la especificación de diseño (varianzas del plan de prueba, diseños de prueba o procedimientos de prueba) contra los resultados y se realiza una justificación de las mismas.

- **Evaluación de la Completitud.** Se evalua la completitud del proceso de pruebas contra el criterio de completitud especificado en el enfoque del plan de prueba, se compara lo esperado contra lo real y se justifican las características que no fueron los suficientemente probadas.

- **Resumen de resultados.** Esta actividad implica identificar los incidentes resueltos y no resueltos y sus respectivas soluciones.

- **Evaluación.** La evaluación se hace tomando en cuenta los resultados de las pruebas, los criterios de éxito/fracaso y la estimación del riesgo definida en el plan de pruebas. Se debe proveer una evaluación de cada elemento de prueba.

- **Resumen de actividades.** Se resumen las principales actividades de pruebas indicando los recursos de datos consumidos, el personal, tiempo total de cpu consumido y tiempo utilizado para cada una de las actividades de pruebas.

Una vez que se realizan los reportes de la pruebas es necesario conocer que representan estos reportes a nivel global, para esto desarrollamos el seguimiento de las pruebas, en donde se generan una serie de gráficas cuya interpretación permite ubicar el avance de las pruebas y hacer estimaciones del trabajo pendiente por realizar.

Seguimiento de la prueba

El seguimiento de la prueba es la administración de las mismas mediante una supervisión cuidadosa basada en gráficas como son: elementos probados a la fecha vs estimados, estado de los elementos, distribución de defectos en los elementos, defectos corregidos por tipo y reporte de pruebas.

Estas gráficas se revisan y analizan periódicamente de manera que muestren una visión clara de cómo va la ejecución de las pruebas y se realizan estimaciones del trabajo pendiente por realizar. Estas deben actualizarse regularmente de manera que los retrasos o acontecimientos no previstos se detecten y se tomen acciones correctivas buscando la terminación del sistema en la fecha prevista.

Perspectivas del seguimiento de la prueba

Se espera que el usuario realice el seguimiento de las pruebas de manera que le permita tomar decisiones acertadas en cuanto al manejo de los recursos y las estrategias necesarias para concluir el proyecto satisfactoriamente.

Características del seguimiento de la prueba

La fiabilidad, estimación y predicción son las características más importantes que debe considerarse al seguir las pruebas de sistemas, entendiéndose por:

Fiabilidad. Grado en que se puede esperar que una persona lleve a cabo sus funciones con la precisión requerida.

Estimación. Aproximación a futuro que se hace basándose en datos reales.

Predicción. Adelantar un acontecimiento previo a que ocurra.

Restricciones del seguimiento de la prueba

Las gráficas y reportes muestran algunos aspectos relacionados a la ejecución de la prueba entre los cuales tenemos los siguientes: cuantificación de los elementos probados y los faltantes por probar, clasificación de los defectos, visualización de los resultados reales contra los esperados y el grado de avance de la ejecución de las pruebas.

En las siguientes secciones se describirán los diferentes tipos de gráficas como son elementos probados a la fecha vs estimados, estado de los elementos, distribución de defectos en los elementos, defectos corregidos por tipo y reporte de pruebas.

Gráfica de elementos probados a la fecha vs estimados

La gráfica de elementos probados a la fecha muestra el avance de la prueba respecto al tiempo contra lo estimado. Se espera que la ejecución se realize lo más cercano a lo estimado, esto es, comenzar a probar desde el inicio del ciclo de desarrollo del sistema. Se utiliza como apoyo para terminar a tiempo la ejecución de pruebas al sistema.

Gráfica de estado de los elementos

La gráfica muestra el estado de la prueba, se dividen en elementos libres de errores, y con errores, sin probar y estimados. Durante la prueba se presta atención a los elementos estimados que no han sido evaluados.

Gráfica de la distribución de defectos en los elementos

La gráfica muestra cómo se distribuyen los defectos entre los elementos que se han probado identificandolos en críticos, mayores y menores. La gráfica se realiza de acuerdo a la fase del ciclo de desarrollo del sistema. Durante la ejecución de la prueba se presta atención a los defectos críticos ya que provocan desviaciones en las demás fases.

Gráfica de defectos corregidos por tipo

La gráfica muestra los defectos corregidos contra los no corregidos por tipo a una fecha determinada. Los tipos de defectos son críticos, mayores y menores. Es recomendable hacer una gráfica para cada fase del proceso de desarrollo.

Reporte de pruebas

El reporte resume el resultado de la ejecución de las pruebas ejecutadas, defectos encontrados, defectos corregidos, pruebas pendientes, entre otros. Es decir, se analizan los resultados y se determina el grado de avance de la prueba.

Una vez concluido el seguimiento de la prueba se deben aplicar métricas para realizar la evaluación completa del proceso de pruebas que proporciona indicativos que le permitirán al ingeniero de pruebas evaluar y diagnosticar el sistema.

Aplicación de métricas

Una métrica es una indicación medible de algunos aspectos cuantitativos de un sistema como son análisis de los defectos, magnitud, efectividad y costo de la prueba. El objetivo de las métricas es realizar una valoración del software mediante el análisis y evaluación de los resultados en la última etapa de la ejecución del plan de pruebas. En primer lugar se realiza el análisis de defectos presentando información sobre estos organizada en diferentes formas. Enseguida se calcula la magnitud de la ejecución de la prueba para conocer cuanto del código se ha probado. Posteriormente se evalúa la efectividad de la prueba es decir, la eficiencia del proceso al seleccionar las mismas y detectar defectos en el sistema. Por último, se cálcula el costo de la misma. En las siguientes secciones se mencionan las métricas que corresponden a cada una de estas métricas.

Perspectivas de la aplicación de métricas

Se espera el poder valorar la calidad de los sistemas a través del análisis de los resultados de las pruebas y evaluar la efectividad de las mismas.

Características de las métricas

Las métricas deben tener las características de medible, independiente, contable y precisa, entendido por:

Medible. Una métrica debe ser medible. Si el fenómeno no puede ser medido, no hay un modo de aplicar métodos de administración que lo controlen.

Independiente. Las métricas necesitan ser independientes de la influencia humana. La interpretación y la forma que se aplique es la misma independientemente de en dónde y quien la aplique.

Contable. Cualquier interpretación analítica de los datos métricos en bruto es producto de los propios datos, no se requieren de ajustes adicionales.

Preciso. Es una función de exactitud. Por consiguiente, la precisión es una métrica que está explícitamente documentada como parte del proceso de colección de datos.

Restricciones de las métricas

Los resultados de las métricas dependen de la información que se analiza, unos resultados pobres resultan en un diagnóstico pobre.

Análisis de los defectos

El análisis de defectos consiste en clasificar los defectos que se presentan durante las pruebas considerando ambiente, probadores, tipos entre otros. Cabe mencionar que cada una tiene diferente utilidad según su enfoque. A continuación se describen a detalle.

Distribución de defectos por módulo.

La métrica mide la cantidad de defectos de los módulos que se presentaron al realizar las pruebas, esta puede representarse gráficamente acumulando los defectos encontrados en cada módulo del sistema. La atención se enfoca en los módulos que fueron menos estables durante las pruebas (es deseable que los módulos de mayor jerarquía sean los más estables).

Distribución relativa de defectos

Esta métrica normaliza la distribución del defecto presentada en la anterior. La normalización puede ser por puntos de función o líneas de código (Pressman, 1999), en la anterior no se había considerado el tamaño de los módulos. Las proporciones del defecto podrían normalizarse a defectos por 100 puntos por función o defectos por 1000 líneas de código para que se puedan hacer comparaciones entre los módulos. Este gráfica permite identificar los módulos que tienen mayor proporción de defectos.

Distribución de defectos por el ambiente (plataforma).

La métrica mide el grado de compatibilidad con las plataformas. Esto se refiere al comportamiento de los sistemas de software, el cual varía entre computadoras debido a su configuración y sus capacidades La métrica puede representarse gráficamente recolectando los defectos que surgen de las pruebas del sistema en diferentes ambientes.

Distribución de defectos por tipo

La métrica muestra la cantidad de defectos del sistema clasificados por tipo. Esta puede representarse en forma gráfica clasificándolos de acuerdo a su origen que pueden ser de arquitectura, de conectividad, de consistencia, de integridad de la base de datos, de documentación, de interfaz, de instalación, de memoria, de seguridad, estándares y convenciones.

Distribución de defectos por quién lo encontró.

La métrica muestra la cantidad de defectos detectados por los usuarios del sistema que pueden ser el equipo de pruebas, los clientes, los desarrolladores, entre otros. Se espera que la mayoría de los defectos sean encontrados por el equipo de pruebas. Esta métrica puede ser representada graficamente definiendo grupos de usuarios y acumulando los defectos detectados por cada uno.

Distribución de defectos por fase en el ciclo de vida.

La métrica muestra los defectos que se descubrieron en cada fase del ciclo de vida del desarrollo del software. Esta puede ser representada por medio de una gráfica, se espera que vaya decreciendo en cada fase de desarrollo.

Distribución de causas de defectos

La métrica muestra las causas que ocasionaron la mayor parte de los defectos en el sistema. Dada una causa se acumulan los defectos ocasionados por la misma y se realiza una gráfica.

Magnitud de la prueba

La magnitud de la prueba mide la proporción del código que ha sido probado. En este caso se consideran los caminos ejecutados y las instrucciones ejercitadas. A continuación se describen a detalle.

Caminos ejecutados

Un camino es la ruta que se sigue al cumplir una condición. Por ejemplo la proposición *if* contempla dos caminos, el *case* contempla varios caminos, entre otros. Los caminos ejecutados son aquellos para los cuales se realizaron casos de prueba. Estos se calculan dividiendo el número de caminos probados entre el número total de caminos, como se muestra en la ecuación 1.

$$C_E = \frac{N_{CP}}{N_{TC}} \qquad (1)$$

Donde $C_{E=}$ *caminos ejecutados,* N_{CP} *= número de caminos probados y* N_{TC} *= número total de caminos (*N_{TC} *).*

Esta métrica calcula la magnitud de los caminos de la lógica que se ha probado, se utiliza durante las pruebas al código para asegurarse que cada camino se ejecutó por lo menos una

vez (Perry, 1995). El resultado nos muestra el porcentaje de ejecución, es decir el 0.5 indica que se probó la mitad de los caminos, el 0.7 indica que se probó el 70% y asi sucesivamente.

Instrucciones ejecutadas

Las instrucciones ejecutadas se calculan dividiendo el número de instrucciones ejecutadas entre el número total de instrucciones, como se muestra en la ecuación 2.

$$I_E = \frac{N_{IE}}{N_{TI}} \qquad (2)$$

Donde: I_E = *Instrucciones ejecutadas,* N_{IE} = *número de instrucciones ejercitadas y* N_{TI} = *número total de instrucciones.*

Esta métrica mide cuántas instrucciones se ejecutan durante el proceso de la prueba, como una base para evaluar cuanto código ha probado. Una instrucción se refiere a una línea de código. Esta métrica se utiliza durante las pruebas de código para asegurar que cada instrucción se ejecutó por lo menos una vez (Perry, 1995). El resultado nos muestra el porcentaje de código probado, es decir el 0.6 indica que se 60% del código, el 0.7 indica que se probó el 70% y asi sucesivamente

Efectividad de la prueba

La efectividad de la prueba es el grado en que las pruebas han permitido la detección de defectos en el sistema, es decir la eficiencia del grupo de pruebas al aplicar las mismas. En este punto se consideran cuatro métricas, las cuales se describen a continuación.

Defectos descubiertos debido a las pruebas

Los defectos descubiertos debido a las pruebas se calculan dividiendo el número de defectos detectados con las pruebas entre los defectos del sistema total, como se muestra en la ecuación 3.

$$D_{DP} = \frac{D_P}{D_T} \tag{3}$$

Donde: D_{DP} *= defectos descubiertos debido a las pruebas,,* D_P *= número de defectos detectados con las pruebas y* D_T *= defectos del sistema total.*

Esta métrica mide la efectividad del proceso de pruebas. La cantidad de defectos encontrados durante el proceso debe ser mayor comparada contra aquellos que se detectaron por otros medios (Perry, 1995). El resultado nos muestra el porcentaje de defectos descubiertos por el proceso. Un 0.9 indica que las pruebas fueron efectivas en un 90%.

Eficacia de la selección de las pruebas

La eficacia de la selección de las pruebas se calcula dividiendo el número de pruebas entre los defectos del sistema total, como se muestra en la ecuación 4.

$$E_P = \frac{N_P}{D_T} \tag{4}$$

Donde: E_P = Eficacia de la selección de pruebas, N_P = número de pruebas y D_T = defectos del sistema total.

Esta métrica muestra la habilidad al seleccionar y aplicar el plan de pruebas. El plan debe considerar pruebas que detecten la mayor cantidad y variedad de defectos (Perry, 1995). El resultado nos muestra el porcentaje que cada defecto representa respecto a una prueba. Entre menor sea el porcentaje más defectos fueron detectados por la prueba y viveversa. Un 0.2 indica que cada error es un 20% de los errores detectados por la prueba.

Defectos encontrados en relación al código

Los defectos encontrados se calculan dividiendo el número de los defectos encontrados entre el tamaño del sistema, como se muestra en la ecuación 5.

$$D_E = \frac{N_D}{T_S} \tag{5}$$

Donde: D_E =Defectos encontrados, N_D =Número de defectos encontrados y T_S = el tamaño de sistema.

Esta métrica mide la cantidad de defectos del sistema en relación al código (Perry, 1995). La métrica asume que hay una cantidad de defectos ocultos en un sistema de software que deben ser detectados.

Tiempo para corregir defectos

El tiempo para corregir defectos se calcula dividiendo el tiempo total de corrección entre el número de defectos corregidos, como se muestra en la ecuación 6.

$$T_{CD} = \frac{T_{TC}}{N_{DC}} \quad (6)$$

Donde: T_{CD} =Tiempo para corregir defectos, T_{TC} = tiempo total de corrección y N_{DC}= número defectos corregidos.

Esta métrica mide el tiempo que lleva la corrección de los defectos (Perry, 1995). El tiempo de correción debe ser directamente proporcional al tamaño del defecto.

Costo de la prueba

El costo de la prueba es la cuantificación de los recursos utilizados durante el proceso de pruebas y la comparación de los mismos contra el presupuesto planeado. Enseguida se describen a detalle las métricas de que se compone este punto.

Costo de la etapa de prueba

El costo de la etapa de prueba se calcula dividiendo el total de los costos de la prueba entre el costo del sistema total, como se muestra en la ecuación 7.

$$C_P = \frac{T_{CP}}{C_{ST}} \quad (7)$$

Donde : C_P = Costo de la etapa de prueba ,T_{CP} = total de los costos de la prueba y C_{ST} = el costo del sistema total.

Este métrica determina cuanto del costo total del desarrollo del sistema se asigna a probar (Perry, 1995).

Presupuesto alcanzado.

La métrica de presupuesto alcanzado se obtiene del costo esperado al realizar la prueba dividido entre el costo real de la prueba, como se muestra en la ecuación 8.

$$P_A = \frac{C_E}{C_R} \tag{8}$$

Donde P_A = Presupuesto alcanzado, C_E = costo esperado al realizar la prueba y C_R = costo real de probar.

Esta métrica indica la efectividad del equipo dirigiendo la prueba dentro del presupuesto (Perry, 1995).

Prueba de automatización

La prueba de automatización se calcula dividiendo el costo de la prueba manual entre el costo de la prueba total, como se muestra en la ecuación 9.

$$P_A = \frac{C_{PM}}{C_{TP}} \tag{9}$$

Donde P_A = Prueba automatizada, C_{PM} = Costo del esfuerzo de la prueba manual y C_{TP} = costo de prueba total.

Esta métrica muestra cuantos de los recursos de la prueba se han consumido manualmente y cuanto representa la automatización del proceso de pruebas (Perry, 1995).

En este capítulo se describió una guía propuesta para probar sistemas de software, la cual se dividió en cinco etapas. La primera etapa establece el plan de prueba con el propósito de presentar una visión global del entorno en el que se desarrolla, la segunda desglosa las pruebas que se van a aplicar identificando casos y procedimientos, la tercera etapa registra los resultados de la aplicación de las mismas en reportes con el objetivo de dejar plasmados los eventos y acontecimientos que se van presentando durante el proceso, la cuarta etapa supervisa el desarrollo de ellas para terminar en el tiempo previsto y la quinta etapa permite la aplicación de métricas para evaluar los resultados obtenidos y diagnosticar el sistema.

En el capitulo IV se describe una propuesta de pruebas nuevas a aplicar en el proceso de desarrollo de sistemas utilizando la guía presentada en este capítulo. Para la realización de las pruebas se utilizan listas de revisión para cada fase del ciclo de desarrollo de sistemas. Las cuales se utilizan para garantizar que cada una cumpla su objetivo y sirven como filtro para detectar errores y mejorar la calidad de los sistemas.

Una propuesta de pruebas en el proceso de desarrollo de software

El proceso de pruebas de la guía planteada en el capítulo anterior tiene la finalidad de cambiar significativamente la forma en que éstas se realizan, esto es, pasar del enfoque tradicional en donde las pruebas se hacen al final del ciclo de vida del software a un nuevo enfoque donde se empiece a probar desde el principio y en cada una de las fases del mismo.

Se busca ir eliminando paulatinamente aquella información redundante, ambigua, inconsistente y contradictoria que confunda o desvíe el desarrollo del proyecto utilizando pruebas independientes que descubran errores en cada fase del ciclo de vida.

La planeación y ejecución de las pruebas es lógica y deducible. Se trata de ir probando los documentos que van surgiendo durante el ciclo de desarrollo de software para que no se arrastren errores a las otras fases.

El proceso de pruebas inicia con la revisión del documento de requerimientos, el cual debe ser analizado a conciencia para detectar la mayor cantidad de defectos posible con el objetivo de medir el impacto que puedan tener a nivel global y para estimar el tiempo y recursos que se necesitan para corregirlos a esto se le llama verificación.

La verificación es una forma de pruebas que se realiza en cada fase del proceso de desarrollo de software para mejorar la calidad del mismo (Kit ,1996). El propósito de la verificación es asegurar que el software reúne las necesidades del cliente como se especificó en el documento de requerimientos.

El concepto de verificación incluye dos criterios fundamentales. Primero, el software debe realizar correcta y adecuadamente todas las funciones definidas. Segundo, el software no debe realizar ninguna función que por sí misma ni en la combinación con otras funciones pueda degradar el desempeño del sistema entero (Lewys, 2000).

La verificación es importante porque permite establecer una trazabilidad entre las fases del desarrollo del software asegurando que los requerimientos se prueben adecuadamente por medio de un rastreo controlado. Por otro lado asegura que las funciones del software cumplan su propósito y los atributos requeridos como portabilidad, acoplamiento, entre otros.

Una premisa es que la verificación no sólo se enfoca en la detección de defectos sino que también permite evaluar si el sistema es utilizable o no. Es decir, un sistema puede funcionar adecuadamente y no le servirle al usuario.

La verificación de sistemas se hace mediante listas de revisión. Una lista de revisión es una serie preguntas acerca de aspectos específicos de un sistema (Perry, 1995). El ingeniero de pruebas debe tener claro el propósito de cada una de las preguntas y saber cómo aplicarlas a una aplicación específica .

La interpretación de las listas de revisión se realiza en forma individual y dependiendo del aspecto que se esté evaluando. Primeramente se deberán contestar las preguntas que contiene la misma. Cada pregunta tiene cinco rangos de evaluación, los cuales están en la escala del 1 al 5 en donde los valores que se consideran son ninguno, varios, la mitad,

mayoría y todos. La respuesta depende del criterio del ingeniero de pruebas, la idea es que determine cuántos de los elementos satisfacen la pregunta positivamente y en base a esto seleccionar el rango. Una vez que termina de llenar la lista verifica cuál es el rango que más se repite, este corresponderá al resultado global de la lista de revisión, por ejemplo si el rango es 4 entonces significa que la mayoría de las preguntas tuvieron una respuesta favorable y se interpreta "la lista de revisión en forma global tuvo una tendencia de 4" . Las empresas desarrolladoras necesitan mantener rangos altos para mantener su calidad y su prestigio.

Las ventaja de las listas de revisión es que limitan el alcance de la prueba al número de preguntas y se pueden usar para documentar el resultado del proceso de la prueba.

Una desventaja del uso de listas de revisión es que frecuentemente muchas de las preguntas no son apropiadas ó no se entiende la intención de las preguntas. Otra desventaja es que aumentan el costo de una etapa del desarrollo de software considerablemente. Sin embargo, reducen verdaderamente el costo completo del software porque los defectos se corrigen desde el principio y no se acarrean a las demás etapas (Perry, 1995).

Fases de prueba durante el ciclo de desarrollo del software

Las fases a probar o verificar son análisis de sistemas, diseño, implementación y liberación. En la fase de análisis lo que se prueba o verifica es consistencia, claridad, redundancia, ambigüedad de los documentos de requerimientos y especificación.

En la fase de diseño se verifica que los diagramas de la especificación del análisis pasen correctamente por ejemplo que los requerimientos no sean ambiguos y que no se omitan. Esto se realiza en el diseño de alto y bajo nivel. En la fase de implementación se verifica la modularidad, la interacción, la funcionalidad y usabilidad .En la fase de liberación verificamos la modularidad , la interacción, la funcionalidad y usabilidad. En la fase de liberación se aplican pruebas alfa y beta realizada por el desarrollador y por gente externa, para su validación.

A continuación se muestran una propuesta de pruebas para cada fase del proceso de desarrollo del software.

Análisis del sistema

El análisis del sistema es la fase del desarrollo del software en donde se elaboran documentos que presentan los requerimientos y la arquitectura del sistema de software. Estos pueden ser: análisis de requerimientos y especificación de requerimientos.

El análisis del sistema es la base para el diseño del sistema, es la fase en donde se toman las decisiones más importantes que darán la pauta a seguir para cumplir con las necesidades dispuestas por el cliente.

Para asegurar que se realizó un buen análisis se requiere de la aplicación de pruebas que permitan detectar errores e inconsistencias en el mismo. No probarlo aumenta las probabilidades de que permanezcan errores sin detectar y se acarreen a las demás fases.

En las siguientes secciones se muestra un propuesta de las pruebas que se aplican a los requerimientos y especificaciones.

Prueba o verificación al documento de análisis de requerimientos

El análisis de requerimientos es la descripción de las características que debe cubrir el sistema (Pressman, 1999), las cuales fueron tomadas de común acuerdo entre los desarrolladores del sistema y el cliente.

Para verificar el análisis de requerimientos proponemos la utilización de la lista de revisión de defectos R1 (ver figura 2 ó apéndice C.1) que permite identificar requerimientos redundantes, contradictorios, ambiguos, incompletos y olvidados. Cada uno debe ser analizado individualmente y si se presenta algún error debe reportarse bajo el formato de registro de defectos (ver apéndice D) en este se clasifican los defectos encontrados, se describe la severidad, el tipo de error, su ubicación y sus características.

Una vez corregidos los defectos se utiliza la matriz de seguimiento de requerimientos (ver apendice B.2). Se espera que sean lo suficientemente claros para que puedan ser modelados en la especificación de requerimientos.

A continuación se propone la verificación de la especificación de requerimientos, la cual puede realizarse de acuerdo a diferentes metodologías. En este trabajo se toma como referencia el modelado UML (Unified Modeling Languaje) ya que es de los métodos más completos (Larman, 1999) y está basado en la unificación de varios criterios, además tiene una probabilidad de que se convierta en un estándar para desarrolladores de software. En la siguiente sección se describe una propuesta para realizarla utilizando el lenguaje UML.

Lista de Revisión de defectos al Documento de Requerimientos (R1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Los requerimientos muestran una diferencia clara entre las funciones y los datos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Los requerimientos explican toda la información que será desplegada al usuario?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Los requerimientos sugieren las respuestas del sistema a las condiciones error?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Son claros los requerimientos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Son concisos los requerimientos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se puede probar cada uno de los requerimientos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Hay requerimientos insiduados?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Hay requerimientos contradictorios?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Hay puntos que no están mencionados en la especificación de requerimientos del sistema ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Se declaran requerimientos de desempeño?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿ Los requerimientos incluyen frases complejas difíciles de entender?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Figura 2. Forma para la verificación de la prueba al documento de requerimientos

12. ¿Los requerimientos están formulados de manera que pueden actualizarse?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Hay requerimientos que contienen un diseño detallado innecesario?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Se especifican los requerimientos en tiempo real ?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿Se especifica la precisión y exactitud de cálculos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿Es posible desarrollar un conjunto de pruebas basado en la información de la especificación de requerimientos del sistema?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

¿Qué información está faltando? _____

17. ¿Están declaradas las asunciones y/o dependencias claramente?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿El documento contiene toda la información que necesite para apoyar la comprensión de los requerimientos del sistema?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Figura 2. Continuación

Prueba o verificación al documento de especificación de requerimientos

La especificación de requerimientos es la representación de los requerimientos en modelos diagramáticos de manera que conduzcan a una correcta implementación del software (Pressman, 1999).

El enfoque que prueba a la especificación de requerimientos se puede realizar utilizando modelos diagramados con el Lenguaje de Modelado Unificado (UML de sus siglas en ingles Unified Languaje Modeling) por considerarse este un estándar.

Los diagramas básicos que se consideran más relevantes y proporcionan más información son casos de uso, diagramas de clases, diagramas de secuencia y diagramas de estados. A continuación se presenta una breve descripción de estos diagramas:

Casos de uso: Los casos de uso son interacciones típicas entre un usuario y un sistema de cómputo, se obtienen analizando los distintos componentes que el usuario desee agregar al sistema (Fowler, 1999).

Diagrama de clases: Los diagramas de clase muestran una panorámica de todo el sistema mostrando las clases principales y derivadas, sus dependencias y asociaciones.

Diagramas de secuencia: El diagrama de secuencia es una representación de los eventos generados por actores externos, su orden y los eventos internos del sistema mostrados en un determinado caso de uso (Larman, 1999).

Diagramas de transición de estados: Los diagramas de transición de estados presentan el ciclo de vida de un objeto, esto es, proporcionan información sobre los cambios o transacciones que ocurren entre objetos y sobre el comportamiento dirigido por eventos de un objeto.

Una descripción más completa del tema se proporciona en el libro "UML y patrones" (Larman, 1999).

Como verificación a los modelos generados en la especificación de requerimientos se propone la utilización de la lista de revisión de defectos en diagramas UML (ver apéndice C.2) que permite identificar diferentes tipos de errores en los modelos. Cada modelo debe ser analizado individualmente y si se presenta algún error debe reportarse bajo el formato de registro de defectos (ver apéndice D) en este se clasifican los defectos encontrados, se describe la severidad, el tipo de error, la ubicación y las características.

Con esta prueba se concluye la verificación al análisis del sistema. En la siguiente sección se presenta la propuesta de pruebas al diseño del sistema.

Diseño del sistema

El diseño de sistemas es la fase de desarrollo en donde se traduce el análisis del sistema en una representación del software. Este se divide en diseño de alto y bajo nivel. El diseño de alto nivel se centra en la transformación de los requisitos de datos y en la arquitectura del

software y el diseño de bajo nivel se ocupa del refinamiento de la arquitectura que lleva a una estructura de datos detallada y a las representaciones algorítmicas del software (Pressman, 1999).

Verificar el diseño es muy importante porque permite guardar y mantener una estructura estable del sistema. Como el diseño está muy cercano a la implementación, corregir errores en esta fase aumenta las probabilidades de terminar el sistema a tiempo.

En las siguientes secciones se muestra un propuesta de las pruebas que se aplican en esta fase, para verificar el diseño de alto y bajo nivel, la entrada/salida, condiciones estéticas, validación, búsquedas y procedimientos.

Prueba o verificación al diseño de alto nivel

El diseño de alto nivel define la arquitectura del sistema de acuerdo a sus requerimientos considerando el tipo de usuarios y la plataforma en donde será implementado, describe la división del sistema en subsistemas y las interfases necesarias para llevar a cabo la funcionalidad definida. También, incluye el diseño de archivos o base de datos de acuerdo a las necesidades de información (Pressman, 1999).

Para verificar el diseño de alto nivel se propone utilizar la lista de revisión de defectos DA1 (ver apéndice C.3) que permite identificar requerimientos y funciones redundantes, contradictorios, ambiguos, incompletos y olvidados, además permite detectar errores en la definición de la interfaz. Los errores se reportan bajo el formato de registro de defectos (ver apéndice D) en este se clasifican los defectos encontrados, se describe la severidad, el tipo de error, la ubicación y las características.

Prueba o verificación al documento de diseño de bajo nivel

En el diseño de bajo nivel se detalla la funcionalidad de las interfases especificando las clases/funciones de que se compone cada módulo. Describe las prioridades de implementación de las interfases. Por último se muestra el pseudocódigo que dará la funcionalidad a las interfases del sistema.

Para la verificación del diseño de bajo nivel se propone utilizar la lista de revisión de defectos DB1 (ver apéndice C.4) que permite identificar problemas en la arquitectura del sistema que dificulte u obscurezcan la comprensibilidad de los módulos. Los errores se reportan bajo el formato de registro de defectos (ver apendice D) en este se clasifican los defectos encontrados, se describe la severidad, el tipo de error, la ubicación y las características.

En la siguiente sección se describe la verificación a los diagramas del diseño, los cuales al igual que el análisis están basados en los diagramas UML.

Prueba o verificación al diseño del sistema utilizando el lenguaje de modelado UML

El modelo de análisis nos muestra una comprensión detallada de los requerimientos, mientras que el modelo del diseño modela el sistema dándole forma para que soporte todos los requerimientos. Los modelos del diseño se obtienen del modelo del análisis.

Para la verificación del modelo del diseño se propone utilizar el procedimiento mencionado anteriormente en la especificación de requerimientos. Por ejemplo cada caso de uso del modelo del análisis tenga su correspondiente en el modelo del diseño.

Prueba o verificación de entrada/salida

Entrada son los datos proporcionados por el usuario al sistema y salida es la información que se recibe del sistema. El diseño de la entrada de un sistema determina la calidad de la salida del mismo. Sin embargo, antes de convertirse en una salida adecuada, algunos datos sufren transformaciones y otros sólo se almacenan.

Las salidas son muy importantes para el usuario porque de éstas depende la realización de sus funciones. Toda salida tiene un propósito y si la salida no cumple con una función, ésta no debe crearse, no es suficiente que se presente al usuario sólo porque es posible hacerlo (Kendall, 1991). En consecuencia, el contenido y la estructura de la información del despliegue de las pantallas de captura y reportes generados por el sistema debe ser cuidadosamente analizado y evaluado con el propósito de facilitar a los usuarios la comprensibilidad del sistema.

Se propone realizar la verificación de entrada/salida utilizando la lista de revisión de defectos ES1(ver apéndice C.5), los errores se reportan bajo el formato de registro de defectos (ver apéndice D).

Con esta prueba se verifica que el despliegue de las pantallas y los reportes se realicen adecuadamente. En la siguiente sección se describe la prueba que permite evaluar la apariencia del diseño de las pantallas.

Prueba o verificación de las condiciones estéticas

Las condiciones estéticas son la distribución y presentación de los componentes de la interfaz. El diseño de las pantallas deben tener un flujo lógico y una apariencia ordenada, mantener la consistencia y ser fáciles de recordar. Si el usuario se siente atraído por la interfaz es muy probable que sea más productivo y cometa menos errores.

El propósito es que el usuario se sienta cómodo al utilizar el sistema ya que dura mucho tiempo sentado enfrente de su computadora y si la interfaz no es la apropiada el cansancio se puede presentar tempranamente. En el diseño de la interfaz se consideran alineación, colores usados, tipos y tamaño de letra, ortografía, entre otros.

Las condiciones estéticas se pueden verificar utilizando la lista de revisión CE1 (ver apéndice C.6) y la lista de revisión P1 (ver apéndice C.7). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Después de detectar los defectos en la apariencia de la interfaz, se verifica la validación del sistema. En la siguiente sección se describe esta prueba.

Prueba de validación

La validación consiste en atrapar los errores durante la entrada de datos por medio de un dispositivo, antes de su procesamiento y almacenamiento. Los tipos de error pueden ser: datos faltantes, incorrectos, inválidos, obsoletos, fuera de rango, entre otros.

Como los errores no pueden evitarse es sumamente importante que se tomen en cuenta como parte de la programación, esto es, el código del sistema debe tener rutinas específicas que permitan cubrir los diferentes tipos de error en que el usuario puede incurrir. De esta manera se previene al usuario y le dá la oportunidad de repetir la entrada hasta que sea aceptada como válida por el sistema. El propósito de la validación es asegurar la calidad de los datos previniendo problemas futuros que se ocasionan por la entrada de datos erróneos.

Para la validación se propone la utilización de las listas de revisión V1 (ver apéndice C.8) y la lista de revisión E1 (ver apéndice C.9). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Una vez que se detectan los diferentes tipos de error en la interfaz, se verifica que se cubran las necesidades de información de forma completa y correctamente. En la siguiente sección se muestra en que consiste esta prueba.

Prueba o verificación de las búsquedas

Las búsquedas son las cuestiones que se plantean en relación con los datos almacenados en una base de datos. Una de las necesidades del usuario al utilizar un sistema es obtener

información de la base de datos que cumpla determinadas características en un momento determinado. Por esta razón, es necesario que se haga una investigación profunda acerca de las solicitudes de información más comunes requeridas por el usuario.

Las búsquedas involucran tres elementos: entidad, atributo y valor. Entidad es cualquier objeto acerca del cual alguien escoge recolectar datos como una persona, una ciudad. Atributo es alguna característica de una entidad como nombre, edad, sexo, entre otros. Valor es el contenido del atributo. Cada búsqueda tiene por lo ménos dos de éstos elementos y su objetivo es obtener el tercero.

Para la verificación de las búsquedas se propone utilizar la lista de revisión B1 (ver apéndice C.10). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Al realizar esta prueba se verifica que las necesidades de información del usuario se hayan cubierto correcta y completamente. En la siguiente sección se describe como se realiza la verificación del manual de procedimientos.

Prueba o verificación de procedimientos

Un procedimiento es una serie de pasos para realizar una actividad específica como instalación del sistema, recuperación de fallas, respaldos entre otros. Nada es más frustante que seguir fielmente un procedimiento y obtener resultados que no coinciden con los anticipados por el mismo, estas deficiencias ocasionan retardos en el desempeño del trabajo porque los usuarios pierden tiempo tratando de adivinar cómo solucionar el problema.

La prueba de los procedimientos tiene dos propósitos: comprobar la claridad de la redacción y comprobar que se obtienen resultados satisfactorios al llevarlos a cabo.
Los procedimientos que se utilizan para el manejo y administración de un sistema de software se encuentran descritos en el manual de procedimientos.

El contenido del manual de procedimientos se puede verificar utilizando la lista de revisión PR1 (ver apéndice C.11). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Con esta prueba se concluyen la verificación del diseño del sistema. En la siguiente sección se presenta la propuesta de pruebas a la implementación.

Implementación

Implementación es la fase en la cual se traduce el diseño de bajo nivel en líneas de código utilizando un lenguaje de programación. En las fases de análisis y diseño se realizó la parte creativa, la fase de implementación depende del diseño, los sistemas con un diseño pobre son más dificiles de entender, programar y ampliar. Uno bien definido simplifica enormemente la tarea que programar, sin embargo se necesita creatividad para enfrentar los problemas del lenguaje de programación.

En las siguientes secciones se muestra una propuesta de las pruebas que se aplican en esta fase, se presentan pruebas para codificación, archivos, interfaz, navegación, integridad y transacciones. Para esto, se hizo una revisión bibliográfica de analisis y diseño de sistemas.

Prueba o verificación de la codificación

Código es el conjunto de instrucciones que forman un sistema. Un sistema se implementa utilizando las estructuras y operaciones suministradas por el lenguaje de programación. A través del entendimiento de cómo se implementan las características del lenguaje, se mejora la habilidad para escribir sistemas. Es decir, cuando se entiende cómo se crean y manipulan los datos como arreglos, cadenas, listas o registros, se pueden escribir programas más eficientes que sean fáciles de entender.

Después de implementarlo, es necesario depurarlo ó poner a prueba cada una de sus partes para garantizar su confiabilidad y legibilidad, eliminando cualquier problema que pueda causar confusión.

Para la verificación del código se propone utilizar la lista de revisión C1 (ver apéndice C.12) para probar cada módulo. Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Esta prueba pretende complementar las pruebas que se realizan al código que se encontraron en la literatura revisada las cuales se describen previamente en el capítulo II sección 5, en donde se realiza una descripción de los casos de prueba de caja blanca para probar la lógica buscando que todas las líneas de código sean ejecutadas de manera que se prueben todas las líneas de código por lo menos una vez y de caja negra para probar el comportamiento a las entradas de datos del usuario. En la siguiente sección se describe la prueba de archivos.

Prueba o verificación de archivos

Un archivo es un conjunto de registros. Cada registro está formado por campos a su vez cada campo está formado por caracteres o bytes. Cuando los sistemas requieren del almacenamiento de información, ya sea para el uso del sistema o para generar resultados se valen de los archivos, los cuales permiten guardar la información durante un período temporal o indefinido de tiempo.

Los archivos pueden organizarse en dos formas: secuencial y directa. En los archivos secuenciales los registros están almacenados en una secuencia que depende de algún criterio definido, estos tienen la desventaja de que el acceso es lento cuando se manejan grandes volumenes de información. Por otra parte, los archivos directos permiten accesar directamente un registro de información sin tener que buscar uno por uno todos los registros del archivo, utilizando una llave de acceso dentro del archivo.

La verificación de archivos es importante porque garantiza que la información puede almacenarse y accesarse adecuadamente.

Para la verificación de archivos se propone utilizar la lista de revisión AR1 (ver apéndice C.13). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Al realizar esta prueba se verifica que los archivos tengan un funcionamiento coherente. En la siguiente sección se prueba el diseño de los componentes de la interfaz.

Prueba o verificación de interfaz gráfica

La prueba de interfaz gráfica se refiere a la verificación de los sistemas generados a partir de ambientes visuales. La interfaz gráfica debe quedar muy bien desarrollada para que el

usuario interactue con el sistema fácilmente. En ambientes visuales la funcionalidad inicial de los elementos del sistema se logra sin realizar mucho esfuerzo, únicamente se incorporan al sistema y se compilan. Posteriormente el programador la puede incrementar escribiendo código manualmente.

La verificación de la interfaz gráfica es importante porque garantiza la consistencia del sistema con los ambientes visuales, de manera que los usuarios se familiarizan fácilmente con la funcionalidad de sus componentes.

Para la verificación de la interfaz gráfica se propone utilizar la lista de revisión I1 (ver apéndice C.14) y la lista de revisión CG1 (ver apéndice C.15). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Una vez que los componentes han sido probados y tienen la funcionalidad precisa se realiza la prueba de navegación, la cual se describe en la siguiente sección.

Prueba o verificación de la navegación

La navegación es la interacción o mobilidad que presenta el usuario al utilizar el sistema. Los usuarios pueden seleccionar caminos distintos mediante el teclado o ratón seleccionando opciones del menú y de la barra de herramientas.

Los usuarios tienen libertad de movimiento y seleccionar caminos que nunca fueron considerados en las pruebas. Es decir, pueden navegar libremente por el sistema, seleccionar opciones y verificar su funcionalidad.

Si una opción está desactivada o no es visible nunca será usada, independientemente de que haya funcionado apropiadamente en otro lugar dentro de la aplicación. Suele suceder que la

opción funcione desde el menú y no lo haga a través del ícono que le corresponde, otro problema puede ser que la funcionalidad no corresponda con la opción seleccionada.

Esta prueba se realiza con el objetivo de verificar que la interfaz funciona correcta y completamente.

Para la verificación de la navegación se propone utilizar la lista de revisión N1 (ver apéndice C.16). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Después de realizar la navegación del usuario a través de la interfaz, se necesita verificar la información que se presenta al usuario. En la siguiente sección se describe en que consiste la prueba de integridad.

Prueba o verificación de la integridad

La integridad es el proceso que garantiza que los datos se almacenan correcta y completamente. Las bases de datos almacenan información y no las operaciones que permiten el acceso a ellos. Lo ideal sería que las operaciones se coloquen dentro de la base de datos, de modo que siempre que una aplicación necesite alterar la base de datos se llamara a las operaciones adecuadas. Sin embargo, como esto no es posible la integridad de los datos se debe garantizar dentro del código.

El propósito de la integridad es verificar el contenido de la información para prevenir que se presenten errores durante las transacciones.

Para la verificación de la integridad se propone utilizar la lista de revisión ID1 (ver apéndice C.17). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Después de realizar la prueba de integridad se tiene la seguridad de que la información está libre de errores. En la siguiente seccion se describe la prueba de transacciones, la cual se enfoca directamente en la validación de la base de datos.

Prueba o verificación de las transacciones

Una transacción es el proceso o serie de pasos que permiten alterar el contenido del medio de almacenamiento de los datos (base de datos, archivo, entre otros). Una transacción es una llamada que dura segundos, minutos o días para terminar su procesamiento.

Los fundamentos de una transacción se enumeran a continuación:

- Atomicidad. Se asume que una transacción está completa o incompleta.
- Consistencia. Cuando empieza una transacción, el sistema se encuentra en un estado conocido, cuando termina, debe estar de nuevo en un estado conocido y consistente.
- Aislamiento. Las transacciones deben ser individuales y no afectar o depender de otras transacciones.
- Durabilidad. En cuanto finaliza la transacción no hay ninguna razón para que se deshaga la operación.

El propósito de verificar las transacciones es comprobar que se realicen adecuadamente. En esta prueba se propone utilizar la lista de revisión T1 (ver apéndice C.18). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Con esta prueba se concluyen la verificación de la implementación del sistema. En la siguiente sección se presenta la propuesta de pruebas en la liberación del sistema.

Pruebas de Liberación

Las pruebas de liberación se realizan para asegurar que un sistema de información está terminado y es operativo, se trata de poner el sistema a disposición de usuarios bien capacitados para que lo utilizen. El papel principal de la fase de liberación es supervisar la instalación y el desempeño del sistema interna y externamente al grupo de desarrollo. Esto es, realizar pruebas ALFA y BETA (ver capítulo II sección II.3.2.4).

La interaccción de los usuarios con el sistema es una parte importante del proceso, debido a que ellos deben ser capaces de ejecutar el sistema con la responsabilidad de los recursos que están usando, querer aprender y ser capaces de formular sus propios casos de prueba con base en su propio conocimiento. Es decir, ellos deben hacer una investigación exploratoria de la funcionalidad del sistema.

En las siguientes secciones se muestra una propuesta de las pruebas que se pueden aplicar en esta fase, se presentan pruebas para instalación, ayuda, requerimientos no funcionales y desempeño.

Prueba o verificación de la instalación

La instalación es el proceso que consiste en crear las condiciones ambientales que el sistema requiere para su operación, incluyendo el proceso de instalación.

Algunas empresas han experimentado las pérdidas e inconvenientes debido a un proceso impropiamente planeado y ejecutado de la instalación (Perry, 1995). Mientras el proceso de

instalación es relativamente corto en la duración, puede ser un proceso complejo de completar. El proceso de instalación se debe verificar para asegurar que está completo y tener la certeza de que se realizaron los cambios a los datos y a los archivos durante la instalación

Para la verificación de la instalación se propone la utilización de la lista de revisión IN1 (ver apéndice C.19). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

En la siguiente sección se describe la prueba de ayuda al usuario, cuyo objetivo es apoyar al usuario en la utilización del sistema.

Prueba o verificación de la ayuda

La ayuda es el proceso que consiste en obtener información relacionada particularmente con cualquier función que se lleve a cabo por el sistema. La ayuda puede tener varias presentaciones como la clásica utilizando la tecla F1 que sirve para obtener ayuda rápida sobre la marcha o por contenido en donde se presenta la información completa por temas o en orden alfabético.

El propósito de la verificación de la ayuda es que el usuario sienta confianza de que el sistema le proporcionará las respuestas a las dudas que se le presenten en cualquier momento.

Para la verificación de la ayuda se propone la utilización de la lista de revisión AY1 (ver apéndice C.20). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Después de verificar la ayuda que ofrece el sistema al usuario, es necesario verificar los requerimientos no funcionales del mismo. En la siguiente sección se describen esta prueba.

Prueba o verificación de requerimientos no funcionales

Los requerimientos no funcionales son aquellos que no se consideran en el modelado, sino que son verificados una vez que el sistema está terminado. Sin embargo, deben ser tomados en cuenta como parte del análisis del sistema. Algunos ejemplos son autorización, fiabilidad, portabilidad, acoplamiento, mantenibilidad entre otros.

Para verificar los requerimientos no funcionales se propone la utilización de la lista de revisión RNF1 (ver apéndice C.21) la cual permite identificar requerimientos olvidados. Los errores se reportan bajo el formato de registro de defectos (ver apéndice D) en este se clasifican los defectos encontrados, se describe la severidad, el tipo de error, la ubicación y las características.

Uno de los requerimientos más importantes que se consideran dentro del sistema es la prueba de desempeño. En la siguiente sección se muestra en que consiste esta prueba.

Prueba o verificación del desempeño

El desempeño es el tiempo que dura la ejecución de ciertas actividades dentro de un sistema. Entre las cuales se encuentran el desempeño de la entrada de datos, generación de reportes, cálculos internos, respaldos, comunicaciones en línea, entre otros.

El desempeño de un sistema es muy importante para medir su calidad. Los sistemas con bajo rendimiento no son tomados seriamente, en su lugar los usuarios intencionalmente exageran el problema y suelen desesperarse al utilizarlo por esta razón es necesario establecer tiempos máximos de respuesta.

En sistemas cliente/servidor se mejora el desempeño del sistema cuando la mayor parte de los cálculos y manipulación de datos se cargan al servidor y el cliente se concentra en presentar los datos y almacenarlos.

El propósito de verificar el desempeño del sistema es cubrir las necesidades de los usuarios al hacer sus peticiones en un tiempo razonable.

Para la verificación del desempeño se propone la utilización de la lista de revisión D1(ver apéndice C.22) y el formato de registro de defectos (ver apéndice D).

En este capítulo se describieron las pruebas que se proponen aplicar en cada fase del proceso de desarrollo del software. La cual se realiza utilizando listas de revisión. Primeramente, en el análisis se describe la propuesta para probar requerimientos y especificaciones respectivamente. Posteriormente se presenta la propuesta para verificar la arquitectura del sistema de acuerdo a sus requerimientos, la entrada y salida de datos, condiciones estéticas, búsquedas y procedimientos. Enseguida proponemos la verificación de la navegación, transacciones, interfaz, integridad y archivos. Por último, se describe la propuesta de la ayuda, instalación, requerimientos no funcionales y desempeño.

Con esto, se pretende complementar las actividades de pruebas que se encontraron en la literatura revisada, las cuales te dan una guía paso a paso para que el ingeniero de pruebas

emplee menos tiempo en la revisión así como evite la omisión de alguna revisión que pudiera ser importante.

En el siguiente capítulo se describe la evaluación preliminar de la guía propuesta utilizando algunas de las listas de revisión propuestas en este capítulo.

Evaluación preliminar de la guía de pruebas

La evaluación de la guía de pruebas propuesta es presentada en este capítulo. El proceso que se llevo a cabo para esto es el siguiente, primeramente se puso a disposición de un grupo de usuarios y se recabó información acerca de la misma. En este caso, se seleccionó a diversos especialistas de la computación con el objetivo de conocer las opiniones que pudieran tener sobre la guía y verificar si se cumplía con las necesidades individuales al utilizarla al probar diferentes sistemas de software.

La evaluación preliminar no contempla el desarrollo de las etapas de la guía de pruebas, debido a su tamaño y al tiempo disponible para probar, solamente se enfoca en medir la factibilidad de su aplicación y eliminar cualquier deficiencia que haya sido detectada por los usuarios durante su revisión.

También, se presenta la aplicación de la guía de pruebas basada en tres enfoques de completitud del sistema: pruebas a sistemas en todo su ciclo de desarrollo, a sistemas terminados o parcialmente probados y pruebas tradicionales.

Por último, se describe una propuesta para probar la guía de pruebas y los resultados de la aplicación de la misma a dos sistemas: SCART (Gómez ,1998) y HEPRODES (Vázquez, 2002).

Evaluación preliminar de la guía de pruebas

La evaluación preliminar de la guía de pruebas consiste en realizar una revisión del documento para medir la factibilidad de aplicarse a diferentes sistemas de software. Para

esto se seleccionó un grupo de usuarios y se realizó una entrevista en donde se les encuesto sobre la utilidad de la guía y sus características como: complejidad, utilidad, tamaño y aplicación. En las siguientes secciones se describen los detalles de la selección de usuarios y de la entrevista. Se presenta un resumen de los comentarios y sugerencias de los usuarios entrevistados y se describen las modificaciones realizadas a la guía en base a los resultados obtenidos.

Selección de usuarios

Para la evaluación de la guía se seleccionó un grupo de estudiantes de Maestría y Doctorado en Ciencias de la computación del Centro de Investigación Científica y Educación Superior de Ensenada (CICESE), cuyo principal requisito fué que mostraran interés sobre las pruebas de sistemas de software.

El grupo de usuarios se formó de seis personas: dos estudiantes de doctorado y cuatro estudiantes de maestría. Los cuales emitieron su opinión sobre la factibilidad de aplicar la guía a sus sistemas de software, esto se llevó a cabo mediante una entrevista.

En la siguiente sección se describe como se compone la entrevista que se le realizó a los usuarios para evaluar el documento de la guía de pruebas.

Entrevista

Una vez seleccionados los usuarios, se les proporcionó en forma impresa el documento de la guía de pruebas por un período de una semana. Después se realizó la entrevista que se dividió en dos partes. En la primera parte se les pidió a los usuarios que calificaran la

facilidad o dificultad de realizar cada una de las cinco etapas de la guía en la escala del 1 al 5 en donde los valores que se consideran son muy fácil, fácil, regular, difícil y muy difícil. En la segunda parte, se les pidió que opinaran libremente sobre algunos aspectos de la misma como contenido, complejidad, utilidad y tamaño.

Al final, se anotaron sus propuestas sobre cambios, agregaciones y modificaciones a la guía, con el objetivo de tomarlos en cuenta y realizar una versión mejorada de la misma.
La entrevista se plasmó en un cuestionario, el cual se presenta en el apéndice F.
En la siguiente sección se muestran los resultados de la entrevista realizada a los usuarios de la guía de pruebas.

Resultados de la entrevista

Los resultados de la entrevista en donde se califica la facilidad o dificultad de realizar cada una de las cinco etapas de la guía en la escala del 1 al 5 en donde los valores que se consideraron fueron muy fácil, fácil, regular, difícil y muy difícil se muestra a continuación.

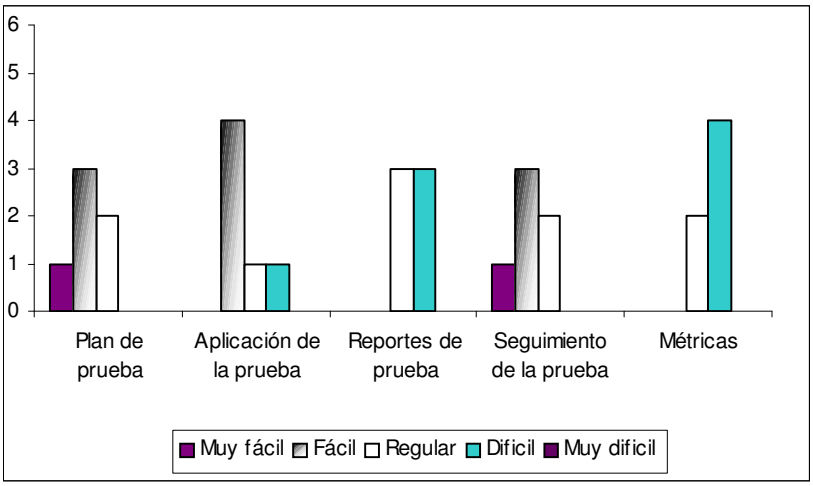


Figura 3. Evaluación de las etapas de la guía de pruebas

- Plan de pruebas. **La opinión que los usuarios tuvieron sobre el desarrollo del plan de pruebas, es decir el esfuerzo para entender y describir cada sección, fue que se consideró muy fácil para un usuario, fácil para 3 usuarios y regular para 2 usuarios, con esto resumimos que el desarrollo de la guía es fácil (ver figura 3).**

- **Aplicación de la prueba.** La aplicación del plan de pruebas consiste en desglosarlo en diseños, casos y procedimientos de prueba. De la totalidad de usuarios entrevistados cuatro opinan que la etapa es fácil, uno piensa que es regular y uno que es difícil (ver figura 3).

- **Reportes de pruebas.** La evaluación consistió en recabar la opinión de los usuarios sobre el registro de los resultados de las pruebas en cuatro tipo de reportes como son: bitácora de prueba, reporte de incidentes, reporte de transmisión y resumen de reportes de prueba. En donde tres de los usuarios opinaron que la etapa es regular, mientras que los demás piensan que es difícil (ver figura 3).

- **Seguimiento de la prueba.** La etapa de seguimiento consiste en realizar una serie de gráficas que permitan ver el avance de la prueba es decir, como se está llevando a cabo el proceso de pruebas. En los resultados, un usuario opinó que es muy fácil realizar el seguimiento, tres opinaron que es fácil y dos dicen que es regular (ver figura 3).

- **Métricas**. En esta etapa se evaluó la opinión que los usuarios tienen sobre las métricas y su interpretación. En este caso, dos usuarios piensan que es regular la selección e interpretación de las métricas y cuatro dicen que es difícil (ver figura 3).

En la siguiente sección se muestran los resultados obtenidos de la segunda parte de la entrevista que consistió en recabar su opinión sobre las características de la guía de pruebas.

Comentarios y sugerencias

Los aspectos considerados en la segunda parte de la entrevista realizada son contenido, complejidad, utilidad, tamaño y aplicación de la guía de pruebas. A continuación se describen en detalle los resultados obtenidos:

- **Contenido.** Los usuarios piensan que el contenido del material de la guía propuesta concentra la información necesaria para realizar un proceso de pruebas eficiente, los temas y cantidad de información es suficiente y útil. Además, la secuencia de las etapas está relacionada y organizada.

- **Complejidad.** Los usuarios consideran que la guía es comprensible ya que explica claramente ¿cuál sería la utilidad de cada etapa y documento? y ¿qué se podría hacer en cada una de éstas?.

- **Utilidad.** Los usuarios de la guía piensan que el uso de la guía beneficia a desarrolladores de software de todos los niveles ya que es posible acoplarlo a los diferentes tipos de sistemas. Por otra parte, consideran que se puede incorporar fácilmente a cualquier proceso de desarrollo de software y que están dispuestos a utilizarlo.

- **Tamaño del modelo.** El tamaño de la guía se calificó como muy amplio. Los usuarios comentaron que los sistemas pequeños no necesitan tanta burocracia. Generalmente, dedican poco tiempo a analizar y diseñar y empiezan a codificar, de manera que el tiempo que dedican a las pruebas es muy reducido.

En la siguiente sección se describen las modificaciones realizadas a la guía de pruebas de acuerdo a los resultados obtenidos.

Modificaciones a la guía de pruebas

Las modificaciones que se hicieron a la guía como resultado de la evaluación son las siguientes.

- **En el plan de pruebas:** se agregaron dos secciones nuevas *descripción del sistema* y *mecanismos de prueba.* En la primera sección se resume la funcionalidad del sistema y en la segunda se describen los mecanismos que se utilizan para realizar las pruebas.

- La sección *pruebas liberadas* se cambió por *documentos de prueba* y la sección *configuración del ambiente de prueba* se cambió por *requisitos de Software y Hardware*.

- Para evitar la duplicidad de secciones se eliminó la sección *criterio de éxito o fracaso*, esta sección no se considera requerida en el plan de pruebas, es hasta la aplicación de la prueba donde toma efecto.

- **En la etapa de seguimiento:** se trasladaron dos gráficas hacia la etapa de métricas: el *reporte de distribución de defectos* y *reporte relativo de distribución de defectos* debido a que se consideran parte de las métricas de análisis de defectos.

- Se sustituyó la *gráfica de defectos corregidos en un intervalo* por la *gráfica de defectos corregidos por tipo*.

En la siguiente sección se presenta el proceso a seguir para aplicar la guía de pruebas a diferentes sistemas de software.

Utilización de la guía de pruebas

La utilización de la guía se basa en tres enfoques. El primer enfoque consiste en aplicar la guía a sistemas en el inicio de su desarrollo y en cada fase del ciclo de vida del sistema, la idea es detectar la mayor cantidad de errores en las primeras fases de su desarrollo y que vayan disminuyendo conforme avanza el proceso de pruebas.

El segundo enfoque es aplicar la guía a sistemas terminados o parcialmente probados con el objetivo de mejorar su eficiencia y/o obtener un diagnóstico, en este caso se analiza el estado del sistema y se realizan las pruebas que se requieran.

El tercer enfoque consiste en aplicar la guía de forma tradicional, es decir, únicamente en la fase de pruebas del ciclo de vida para evaluarlo. El objetivo de los tres enfoques es obtener resultados que permitan garantizar la calidad de los sistemas.

En la figura 1 mostrada anteriormente en el capítulo III sección I se esquematizan las etapas del proceso de pruebas.

Los pasos para utilizar la guía de pruebas se mencionan a continuación:

- **Planeación de las pruebas.** Los ingenieros de pruebas eligen que pruebas van a aplicar al sistema y el orden en el que se deben realizar, posteriormente realizan el plan de pruebas vaciando los datos correspondientes.
- **Organización de las pruebas.** Los ingenieros de pruebas determinan como probar los requerimientos y el conjunto de especificaciones, con objeto de verificar aquellos que pueden ser probados, para esto seleccionan listas de revisión (ver apéndice E). Adicionalmente, preparan la aplicación de la prueba que consiste del diseño de la prueba, casos y procedimientos. De los casos y procedimientos de prueba utilizados en otros

sistemas escogen aquellos que aún sean pertinentes y los actualizan para utilizarlos. Además, se verifican los elementos que deberán ser probados como se estableció originalmente en el plan de pruebas. El propósito de estas pruebas es verificar las interfaces entre los elementos que están siendo probados y comprobar que pueden trabajar conjuntamente. Los ingenieros de pruebas ejecutan los casos de prueba siguiendo los procedimientos de prueba y anotan los resultados en los reportes de prueba.

- **Dirección.** Los ingenieros de prueba realizan el seguimiento de la prueba y seleccionan aquellas gráficas que se requieren para medir el avance del proyecto analizando cuales elementos han sido probados y cuales tienen errores.
- **Evaluación.** La evaluación consiste en seleccionar y aplicar métricas para realizar una evaluación completa del proceso de pruebas que le permitan al ingeniero de pruebas evaluar y diagnosticar el sistema. Si una prueba no alcanza los objetivos, los casos y procedimientos de prueba deberán ser modificados para lograrlo.

A continuación se mencionan las ventajas y desventajas de utilizar la guía según su enfoque:

- Las ventajas de utilizar la guía en sistemas desde el inicio de su desarrollo es que los errores se detectan y se corrigen desde el principio evitando que surjan mayores problemas en fases posteriores. Por otra parte, la carga de trabajo es balanceada durante el proceso de desarrollo del proyecto. La desventaja es que es costosa y requiere tiempo.

- Las ventajas de utilizar la guía de pruebas en sistemas terminados es que se conoce cuales son las partes del sistema que están fallando, se aplican pruebas muy específicas y

no se tiene que llevar un seguimiento del proceso. La desventaja es que los errores encontrados en el análisis y diseño se van incrementando y muchas veces no se corrigen.

- La ventaja de utilizar la guía en sistemas tradicionales es que es menos costosa y lleva poco tiempo, debido a que se realizan únicamente pruebas de funcionalidad y de usabilidad del sistema. La desventaja es que no se garantiza que se hayan implementado adecuadamente los requerimientos.

En esta sección se presentó en forma sencilla y simplificada como utilizar la guía de pruebas para probar sistemas de software en los tres enfoques mencionados inicialmente, enseguida se muestra una propuesta para probar la guía, cuyo objetivo es obtener un diagnóstico de la misma.

Una propuesta para probar la guía de pruebas

La guía de valoración de sistemas de software durante el ciclo de vida permite realizar pruebas a sistemas en las etapas de análisis, diseño e implementación. La guía se divide en 5 etapas: contenido del plan de pruebas, aplicación de la prueba, reportes seguimiento y métricas, para mayores detalles ver el capítulo III.

La guía se utilizó en dos sistemas SCART (Gómez, 1998) y HEPRODES (Vázquez, 2001), en secciones posteriores se muestran los resultados obtuviéndose unos resultados favorables ya que es fácil de usar y permite la detección de defectos en el sistema. Sin embargo, no se obtuvieron resultados suficientes que nos permitan evaluar la usabilidad de la guía, por esta

razón se requiere que sea utilizada por un grupo mayor de usuarios y obtener un diagnóstico más completo de la misma.

La propuesta de pruebas a la guía contiene los siguientes puntos

- **Objetivos de probar la guía**

Las pruebas a realizarse tienen la finalidad de encontrar la mayor cantidad de defectos en la guía propuesta y obtener retroalimentación a la misma para realizar mejoras.

- **Descripción de las Pruebas**

P1. Plan: Consiste en probar la claridad y comprensibilidad de los puntos del plan de pruebas.

P2. Aplicación: Consiste en probar si los usuarios pueden realizar el desglose del plan de pruebas en diseños de prueba, casos y procedimientos de prueba. Esto es, si los usuarios pueden aplicar el plan de pruebas.

P3. Reportes: Consiste en probar si los usuarios pueden registrar los resultados de la prueba en bitácoras de prueba, reportes de incidentes, reporte de transmisión y resumen de reportes de prueba.

P4. Seguimiento: Consiste en probar la utilidad de los reportes y si con la información recabada es posible la administración del proceso de pruebas.

P5. Métricas: Consiste en probar la utilidad de las métricas y si estas proporcionan una valoración del estado del sistema.

Las impresiones de los usuarios se recaban en un cuestionario como el que se describe en el Apéndice F.

- **Documentos de la prueba.** Como resultado de la prueba se obtienen los siguientes documentos: plan de prueba, aplicación, reportes, seguimiento y métricas de prueba.

- **Necesidades de personal y entrenamiento.** Para la realización de las pruebas a la guía propuesta se requieren personas con un perfil en desarrollo de sistemas y/o conocimientos en pruebas a sistemas. No se requiere entrenamiento en la realización de las pruebas.

- **Requisitos de software y hardware.**

Hardware: Computadoras personales con un sistema operativo como linux o windows.

Software: Se requiere un procesador de palabras como latex ó word.

- **Responsabilidades.** La documentación de la guía se les proporcionará vía internet, en disket ó impreso. La responsabilidad de preparar el ambiente de la prueba y de la ejecución de la prueba recae en el probador.

- **Calendario.** En la tabla III se presenta la calendarización de actividades sugerida para ejecutar el plan de pruebas del modelo. En el se presentan las fechas y las actividades requeridas:

Tabla III. Calendarización de actividades.

Actividades	Tiempo
Elaboración del plan de pruebas	2 semanas
Seleccionar el equipo de pruebas	2 días
Aplicación del plan de pruebas	2 semanas
Ejecución y seguimiento de las pruebas	1 semana
Análisis de resultados	2 semanas

- **Criterios de suspensión y requerimientos de reanudación.** Los usuarios podrán suspender la ejecución cuando se ha realizado parte de la prueba y el resto de la misma resulta confusa ó cuando se presenten aspectos técnicos que hagan imposible continuar la prueba.

- **Riesgos y contingencias.** La tabla IV presenta los riesgos y contingencias que se pueden presentar durante la ejecución de las pruebas.

Tabla IV.Tabla de riesgos y contingencias.

	Riesgos	Contigencias
1	Períodos muy largos de pruebas	Planear las actividades Poner límites de prueba Monitoreo de la prueba
2	Falta de personal de pruebas	Solicitar personal
3	Pérdida de información de la prueba	Hacer revisiones al sistema de respaldo Capacitación para respaldar

- Diseño de pruebas de la guía de pruebas

El diseño de las pruebas desglosa la funcionalidad de una prueba específica del sistema. A continuación se describen cada una de ellas.

a) Funciones estructurales

Realizar el plan de pruebas aplicado a un sistema de software específico (ver capítulo III sección 4).

Desglosar el plan de pruebas en diseños de prueba, casos y procedimientos de prueba de su sistema a aplicar (ver capítulo III sección 5).

Registrar los resultados en cuatro tipos de reportes: bitácoras de prueba, reporte de incidentes, reporte de transmisión y resumen de reportes de pruebas (ver capítulo III sección 6)

Realizar las gráficas de elementos probados a la fecha vs estimados, estado de los elementos, distribución de defectos en los elementos, defectos corregidos por tipo y reporte de pruebas (ver capítulo III sección 7).

Aplicar e interpretar los diferentes tipos de métricas (ver capítulo III sección 8) y realizar un análisis de la información.

- **Características a probar**

Consistencia. La calidad y el tipo de información de la guía.

Complejidad. La facilidad o dificultad de aplicar la guía.

Utilidad. La guía sirve a los usuarios para realizar las pruebas.

Tamaño. La dimensión de la guía.

- **Enfoque.**

El usuario debe realizar el plan de pruebas para probar un sistema de software. Después, desarrollar la aplicación de la prueba basándose en los mecanismos y pruebas mencionadas en el plan y en la documentación del sistema. Además, deberá registrar los resultados de las pruebas en bitácoras, reportes de incidentes, reporte de transmisión y resúmenes de reportes de pruebas.

Para administrar el proceso se realizan gráficas de estado de los elementos, distribución de defectos, defectos corregidos por tipo y diferentes reportes de prueba.

Por último, aplicar las métricas e interpretar los resultados de las mismas y contestar el cuestionario de evaluación (ver apéndice F).

A continuación, en las secciones V.3.1 y V.3.2. se muestran los resultados de la aplicación de las pruebas a los sistemas HEPRODES y SCART.

Aplicación de la guía al sistema "HEPRODES"

La guía de pruebas fué aplicada al sistema HEPRODES ("Herramienta para la Enseñanza de Procesos de Desarrollo de software") por su desarrollador Adrian Vázquez Osorio (Vázquez, 2002). HEPRODES es un sistema que permite la visualización y edición de los procesos de desarrollo y proporciona apoyo para el análisis de los modelos del proceso en forma distribuida.

Un resumen de los resultados de la aplicación de la guía al sistema HEPRODES se menciona a continuación:

El usuario en general considera que la estructura y organización de la guía es comprensible y lógica, sin embargo piensa que existen documentos con estructura parecida (secciones iguales) que dán lugar a indecisiones sobre la información específica de cada documento. Además afirmó que algunos de ellos son muy grandes y tienen secciones de poca importancia o innecesarias. Otro de los comentarios fué que varios nombres de secciones no eran intuitivos y que algunos daban lugar a confusiones. Por otra parte, sugirió que se separaran en varias partes algunas de las secciones puesto que abarcaban más de un tema. Por último, opinó que la parte del modelo en donde se presentaron los problemas fueron en las métricas, específicamente la parte de interpretación, debido a que cada resultado daba lugar a una diversidad de opiniones.

Los resultados de la aplicación de la guía de pruebas al sistema HEPRODES se detallan en el capitulo V de la tésis Herramienta para la enseñanza de procesos de desarrollo de software (Vazquez, 2002).

Aplicación de la guía al sistema "SCART"

La guía de pruebas se aplicó al sistema SCART ("Sistema Colaborativo de Apoyo a Revisiones Técnicas"). SCART es un sistema colaborativo que sirve para realizar revisiones técnicas de los productos que surgen durante el proceso de desarrollo de software y fue desarrollado por Lidia Gómez (Gómez, 1998). En el apéndice E se muestran paso a paso el desarrollo de las etapas de la guía de pruebas. A continuación se muestran los resultados de la aplicación de la guía de pruebas por etapa:

- Establecer el plan de prueba. En el desarrollo del plan de pruebas no se tuvieron problemas en el llenado del mismo ni en la selección de las pruebas a aplicar. Las pruebas que se contemplaron fueron de análisis, diseño, instalación, ayuda, seguridad, funcionalidad e interfaz.
- Aplicación de la prueba. En la aplicación de la prueba se presentó un problema al implementar el ambiente de pruebas descrito en el procedimiento. El problema fué la instalación y configuración del servidor de base de datos de ORACLE debido a que no estaba documentado.
- Reportes de las pruebas. De los reportes de la prueba obtuvimos que la bitácora de prueba es muy útil ya que marca una pauta para realizar las pruebas de una forma ordenada o sistemática. Por otro lado, el resumen de reportes de prueba permite concentrar los resultados de la ejecución de la prueba.

- Seguimiento de la prueba. Las gráficas realizadas durante la administración del proceso no fueron muy útiles, esto se debió a que el período de pruebas fue breve. Además, se necesitan tener datos históricos que sirvieran de base para hacer una mejor planeación.
- Métricas. No se tuvieron problemas para la implementación e interpretación de las métricas.

En este capítulo se presentó la evaluación preliminar del documento guía de pruebas en donde se recabaron las impresiones de los usuarios sobre la facilidad y dificultad de realizar cada etapa y se discutieron aspectos como: contenido, completitud, utilidad y tamaño. Además se describieron la utilización de la guía para probar sistemas de software y las ventajas y desventajas de utilizarlo según su enfoque, el cual puede ser: sistemas probados en todo el ciclo de vida, sistemas terminados probados parcialmente y sistemas probados en forma tradicional. Por último, se describe una propuesta para probar la guía de pruebas y los resultados de la aplicación de la misma en dos sistemas: SCART y HEPRODES.

En el siguiente capítulo se presentan las conclusiones sobre el trabajo realizado, las aportaciones con respecto al estado del arte de las pruebas de sistemas y las recomendaciones para trabajo futuro.

Conclusiones, Aportaciones y Trabajo Futuro

Conclusiones

Actualmente las pruebas de software se realizan siguiendo el camino convencional, una vez terminado el sistema se prueba la funcionalidad básica mediante un plan de pruebas que contiene un conjunto de casos de prueba cuyos resultados son notablemente obvios. Después, ante la respuesta óptima del sistema, se ejecutan las pruebas, se analizan resultados y se libera.

La mayoría de estos sistemas carecen de calidad debido al poco tiempo que se dedicó a probar y detectar errores. Es muy probable que la mayoría de los errores se encuentren una vez liberado el sistema, además algunos de éstos pueden ser críticos y su corrección puede durar meses De este modo, este proceso resulta poco eficiente si lo que deseamos es mantenernos en el mercado creando software de calidad.

La calidad es lo más importante que se debe buscar en cualquier proceso ó actividad. En el desarrollo de sistemas de software lo ideal es que cada fase del ciclo de vida se pruebe antes de continuar con el proceso de desarrollo para obtener mejores resultados al liberar el sistema. Como parte de este trabajo se desarrolló una guía para pruebas de sistemas de software que trata de eliminar estos problemas. Esta guía expone un plan de pruebas por medio del cual se pueden probar todas ó cada una de las etapas del desarrollo del sistema por separado y obtener un diagnóstico del mismo.

Al iniciar esta guía, el enfoque estaba orientado hacia el plan de pruebas y la medición de los resultados utilizando métricas. A través del desarrollo de la misma, se pudo observar que era necesario estructurar mejor el proceso definiendo otras etapas que lo hicieran más claro y completo. Primeramente se observó que el plan de pruebas era bastante amplio lo que impedía el desarrollo del mismo. De aquí que se dividió en tres etapas: contenido del plan de pruebas, la aplicación y los reportes. El contenido explica a grandes rasgos el software, las pruebas a realizar, los recursos requeridos y el calendario a seguir. La aplicación desglosa las pruebas en diseños, casos y procedimientos de prueba. Y los reportes recaban información sobre el ambiente y sucesos que se presentan durante la ejecución de la prueba.

También se agregó una etapa de administración que es el seguimiento. La cual es importante porque apoya a los usuarios en la toma de decisiones ya que muestra el grado de avance de la prueba.

Al final, consideramos que se obtuvo una guía completa que se compone de cinco etapas: contenido del plan de pruebas, aplicación, reportes, seguimiento y métricas. La cual fué puesta a disposición de un grupo de usuarios con el objetivo de obtener sus opiniones sobre su estructura, redacción y aplicabilidad.

Podemos concluir, que la guía desarrollada es útil para usuarios independientes, es de utilidad porque los familiarizará con el concepto global de pruebas. Para una empresa de desarrollo de software es útil porque les permitirá analizar y coordinar eficientemente su proceso de pruebas, reducir tiempos, seleccionar al personal adecuado, reducir costos y errores en los productos.

Por otra parte, de acuerdo a los resultados obtenidos, podemos decir que los usuarios entrevistados consideran que la guía está completa y recomendarían su utilización en pruebas de sistemas de software. Además consideraron que el proceso de pruebas planteado no es complicado, sólo que se requiere tiempo y se pueden realizar sólo las etapas del proceso que les son de utilidad.

Aportaciones

Las aportaciones que presenta este trabajo con respecto al estado del arte de las pruebas de sistemas de software son:

Se proporciona una guía práctica para probar sistemas de software efectivamente. La cual describe los elementos que intervienen en el desarrollo del plan de pruebas, la aplicación y la documentación del mismo utilizando reportes. Además explica cómo administrar la ejecución de las pruebas y estimar el trabajo pendiente por realizar. Por último, permite medir los resultados utilizando métricas.

Se proporciona el proceso a seguir para probar cada fase del ciclo de vida del desarrollo de sistemas: análisis, diseño, codificación y prueba. La descripción del proceso facilita la comprensión de las actividades a seguir en cada etapa especificando elementos a probar, pruebas que se aplican, procedimiento a seguir, resultados y la interpretación de estos.

Se proporciona un conjunto de formatos de listas de revisión, matrices de prueba, listas de errores que se requieren para llevar a cabo el proceso de pruebas de sistemas.

Se proporciona un ejemplo del uso de la guía y el proceso para probar un sistema llamado SCART (Sistema Colaborativo de Apoyo a Revisiones Técnicas) que servirá de base para las pruebas de otros sistemas (ver apéndice E).

Trabajo futuro

Se propone el desarrollo de una herramienta colaborativa cuyo propósito sea apoyar el proceso de las pruebas de software. Esta herramienta cubrirá las etapas documentación, seguimiento, reportes y evaluación. Algunos de los requerimientos que se deberá considerar son que: opere sobre la plataforma web (para trabajo distribuido), que apoye a la elaboración de documentos, de seguimiento a las pruebas, elabore reportes de evaluación, almacenamiento y consulta de información, generación automática de reportes, comunicación en línea entre los integrantes del equipo de prueba.

Se propone aplicar la propuesta de pruebas en diferentes sistemas enfocándose en cada uno por separado con el objetivo de evaluar sus beneficios.

Se propone investigar la aplicación de métricas para analizar aspectos particulares de los sistemas que se evalúan.

Bibliografía

Bourne, Kelly. 1997. Testing Client/Server Systems. McGraw-Hill. New York. 572 pp.

Sommerville, I. 1992. Software Engineering. Addison-Wesley. Menlo Park, Calif. 649 pp.

Wiegers, Karl. 1996. Creating a Software Engineering Culture. Dorset House. 358pp.

Kit, Edward. 1996. Software Testing in the real world. Addison wesley. 241pp.

Rakitin, Steven. 1997. Software Verification and Validation A practitioner's Guide. Artech House. 271 pp.

Black, Red. Managing the testing process. Microsoft press. Washington. 381pp.

Perry, William. 1995. Effective Methods for Software Testing. Wiley-Qed. 539 pp.

McCabe, Thomas J. 1983. Structured Testing. IEEE Computer Society Press. New York. 150pp.

Norris, Mark y Rigby Peter. 1994. Ingeniería de Software Explicada. Noriega Editores. 325 pp.

Martin, James y Odell James. 1992. Análisis y Diseño Orientado a Objetos. Prentice Hall. 546 pp.

Larman, Craig. 1999. UML y patrones. Prentice Hall. 507 pp.

Fowler, Martin. 1997. UML gota a gota. Addison Wesley. 203 pp.

Kendall, Kenneth y Kendall, Julie. 1991. Análisis y diseño de sistemas. Prentice Hall. 881 pp.

Fewster, Mark y Graham Dorothy. 1999. Software Test Automation. Addison Wesley. 574 pp.

Cohen, Daniel y Asín Enrique. 2000. Sistemas de información para los negocios. Mc Graw Hill. 413 pp.

Dustin Elfriede; Rashka Jeff y Paul John. 1999. Automated Software Testing. Addison Wesley. 575 pp.

Schafer Wilhelm; Prieto Rubén y Matsumoto Masao. 1994. Software Reusability. Ellis. 160 pp.

Nielsen Jakob. 2000. Usabilidad Diseños de sitios web. Prentice Hall.415 pp.

Pooley, Rob y Stevens Perdita. 1999. Using UML Addison Wesley. 256 pp.

Rumbaugh, James; Jacobson Ivar y Booch Grady. 1999. El Lenguaje Unificado de Modelado. Addison Wesley. 526 pp.

Gómez, Lidia; 1998. Sistema Colaborativo de Apoyo a Revisiones Técnicas en salas electrónicas de reuniones. 159 pp.

Gómez, Lidia; 1998. Sistema Colaborativo de Apoyo para Revisiones Técnicas “SCART”: Análisis y Diseño. 84 pp.

Vázquez, Adrian. 2002. Herramienta para la Enseñanza de Procesos de Desarrollo de software. 225pp

Dutoit, Allen. 1998. Communication Metrics for software Development. IEEE Trans. Software Eng. Vol 24. Pp 615 - 627.

Schneidewind, Norman. 1999. Measuring and Evaluating Maintenance Process Using Reliability, Risk, and Test Metrics. IEEE Trans. Software Eng. Vol 25. Pp 769 - 781.

Yamaura, Tsuneo. 1998. How To design Practical Test Cases. Hitachi Software Eng. Pp 30 -36.

Frankl, Phyllis; Hamlet Richard y Strigini Lorenzo. 1998. Evaluating Testing Methods by Delivered Reliability. IEEE Trans. Software Eng. Vol 24. Pp 586 - 601.

Wakid, Shukri; Kuhn Richard y Wallace Dolores. 1999. Toward Credible IT Testing and Certification. National Institute of Standards and Technology. Pp 39 - 47.

Bassin Kathryn; Kratschmer Theresa y Santhanam P. 1998. Evaluating Software Development Objetively. IBM T.J. Watson Research Center. Pp 66 - 74.

Kemerer, Chris y Slaughter, Sandra. 1999. An Empirical Approach to studying Software Evolution. IEEE Trans. Software Eng. Vol 25. Pp 493 - 509.

Chidamber, Shyam; Darcy, David y Kemerer, Chris. 1998. Managerial Use of Metric for Object-Oriented Software: An Exploratory Analysis. IEEE Trans. Software Eng. Vol 24. Pp 629 - 639.

Murphy, Gail. 1999. Evaluating Emerging Software Development Technologies: Lessons Learned from Assessing Aspect-Oriented Programming. IEEE Trans. Software Eng. Vol 25. Pp 438 - 455.

Gutjahr, Walter. 1999. Partition Testing vs. Random Testing: The Influence of Uncertainty. IEEE Trans. Software Eng. Vol 25. Pp 661 - 674.

Torii, Koji; Matsumoto, Ken-ichi; Nakakoji, Kumiyo; Takada, Yoshihiro y Shima, Kazuyuki. 1999. Ginger2: An Environment for Computer-Aided Empirical Software Engineering. IEEE Trans. Software Eng. Vol 25. Pp 474 - 491.

Definiciones, acrónimos y abreviaturas

Control de calidad. Trabajo que se hace dentro de un proceso para asegurar que el producto está conforme a los estándares y/o requerimientos.

Aseguramiento de calidad. Actividades sistemáticamente planeadas, necesarias para proveer confianza de que los productos y servicios están libres de errores.

Defectos. Desviaciones de la especificación como pérdidas de información, extravíos de datos ó inconsistencias que causen insatisfacción en el cliente.

Verificación. Todas las actividades de control de calidad a través del ciclo de vida que verifican que la entrega del producto cumple con los requerimientos del cliente.

Validación. Todas las actividades de control de calidad a través del ciclo de vida que validan que el producto final cumple con los requerimientos del cliente.

Pruebas estáticas. Verificación realizada sin ejecutar código.

Pruebas dinámicas. Verificación que se realiza ejecutando código.

Pruebas de unidad. Ejecución de un módulo independiente o unidad de código.

Pruebas de integración. Ejecución de pruebas en grupos de módulos para asegurar que el control y los datos son pasados apropiadamente entre módulos.

Aceptación. Pruebas que se aplican para asegurar que el sistema conoce las necesidades de la organización y el cliente.

Regresión. Pruebas que se realizan después de haber realizado algún cambio para asegurar que solo se realizan cambios deseados.

Pruebas funcionales. Pruebas de requerimientos donde se prueba la funcionalidad para la cual fue hecho el sistema.

Pruebas estructurales. Pruebas que validan la arquitectura del sistema, esto es la implementación del sistema.

Pruebas de caja negra. Pruebas basadas en especificaciones externas sin conocimiento de como el sistema es construido.

Pruebas de caja blanca. Pruebas basadas en el conocimiento de la estructura interna del código y la lógica.

Entradas. Criterios que se utilizan como datos de entrada o entregas necesaria para ejecutar la prueba.

Salidas. Criterios que se utilizan como datos de salida o entregas producidas.

Matrices

El apéndice B contiene formatos de matriz para el manejo o control de la información de las pruebas. Se compone de tres formatos: matriz de clases/funciones, matriz de seguimiento de requerimientos y matriz de riesgos. La matriz de clases/funciones sirve para organizar la ejecución de las pruebas de las clases ó funciones. La matriz de seguimiento de requerimientos permite llevar un control del estado de los requerimientos en las diferentes etapas del desarrollo de sistemas y la matriz de riesgos sirve para controlar los problemas que se presenten durante las pruebas. En general, se utilizan como apoyo a la prueba ya que presentan el detalle de la información facilitando la elaboración de reportes.

Matriz de clases / funciones

El propósito de la matriz de clases / funciones es establecer prioridades sobre las clases o funciones a ser probadas y dar un seguimiento de los mismos. Por cada clase/función se mencionan los casos de prueba que deben ejecutarse durante las pruebas.

Clase/Función	**Localización**	**Prioridad**	**Casos de prueba a ser ejercitados**

Los elementos que componen la matriz de clases/funciones son:

Clase/Función. Nombre de la clase/función.

Localización. Lugar que corresponde a la clase/ función dentro del código

Prioridad. Orden establecido en la ejecución de las pruebas.

Casos de prueba. Serie de casos de prueba a ser ejercitados para probar la clase/función.

Matriz de seguimiento de requerimientos

El propósito de la matriz de seguimiento de requerimientos es conocer el estado en donde se encuentra cada uno de los requerimientos. Sirve al usuario para conocer cuántos de los requerimientos hace falta cumplir y cuantos están terminados.

Etiqueta	**Requeri-miento**	**Categoría**	**E/S**	**Fases**			
				Análisis	**Diseño**	**Implementa-ción**	**Prueba**

Los elementos que componen la matriz de seguimiento de requerimientos son:

Etiqueta. Número de identificador único que corresponde a un requerimiento.

Requerimiento. Descripción de necesidades de un sistema.

Categoría. Tipo de requerimiento, se dividen en funcionales y no funcionales. Además los requerimientos no funcionales pueden ser de desempeño, de volumen, de tensión, entre otros.

E/S. Datos o información utilizada en el requerimiento que sirven de entrada para pasar a las siguientes fases.

Fases. Etapa de desarrollo en la que se encuentra el requerimiento.

Matriz de riesgos

El propósito de la matriz de riesgos es prevenir situaciones problema y en caso de que sucedan saber que hacer al respecto para solucionarlo ó evitar que se presente un problema mayor.

Riesgo	Causa	Contingencia	Responsable

Los elementos que componen la matriz de riesgo son:

Riesgo. Son los eventos que afectan la ejecución del plan de pruebas.

Causa. Son los problemas que hacen que el riesgo se presente.

Contingencia. Son las medidas de prevención para evitar el riesgo.

Responsable. Es la persona encargada de llevar a cabo el plan de contingencia.

Listas de revisión de defectos

El apéndice C contiene listas de revisión para la verificación del proceso de pruebas del sistema. Se compone de 22 listas de revisión entre las cuales podemos mencionar: lista de revisión al documento de requerimientos, lista de revisión de diagramas UML, lista de revisión del diseño de alto y bajo nivel etc. En general se utilizan para verificar diferentes aspectos del sistema.

Lista de revisión de defectos R1

Lista de Revisión de defectos al Documento de Requerimientos (R1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Los requerimientos muestran una diferencia clara entre las funciones y los datos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Los requerimientos explican toda la información que será desplegada al usuario?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Los requerimientos sugieren las respuestas del sistema a las condiciones error?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Son claros los requerimientos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Son concisos los requerimientos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se puede probar cada uno de los requerimientos?

1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Hay requerimientos insinuados?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Hay requerimientos contradictorios?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿Hay puntos que no están mencionados en la especificación de requerimientos del sistema ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

10. ¿Se declaran requerimientos de desempeño?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

11. ¿ Los requerimientos incluyen frases complejas difíciles de entender?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. ¿Los requerimientos están formulados de manera que pueden actualizarse?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Hay requerimientos que contienen un diseño detallado innecesario?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Se especifican los requerimientos en tiempo real ?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿Se especifica la precisión y exactitud de cálculos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿Es posible desarrollar un conjunto de pruebas basado en la información de la especificación de requerimientos del sistema?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

    ¿Qué información está faltando? _____

17. ¿Están declaradas las asunciones y/o dependencias claramente?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿El documento contiene toda la información que necesite para apoyar la comprensión de los requerimientos del sistema?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos U1

Lista de Revisión de Defectos de diagramas UML (U1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

a) Casos de uso

1. ¿Es adecuado el nombre del caso de caso?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se especifica el orden del flujo de acciones?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se especifica cómo y cuando termina el caso de uso?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se identifican los caminos que no están permitidos claramente?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Se identifican los caminos alternativos claramente?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Se especifica la petición de valores?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Diagramas de estado

1. ¿Es adecuado el nombre del diagrama de estado?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se definieron los estados inicial y final?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se definieron todas las condiciones para pasar de un estado a otro?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se estableció un orden lógico en el conjunto de estados?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Todos los estados son alcanzables bajo convinaciones de eventos?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Ningún estado está en un punto que no sea posible salir de él con ninguna convinación de eventos?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Se consideran solo transiciones legales entre estados?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

c) Diagramas de interacción

1. ¿Se identificaron todos los objetos que participan en la interacción?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Los objetos están ordenados jerárquicamente?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se estableció la línea de vida para cada objeto?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Los mensajes explican la semántica de la operación?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Está completa la semántica de la operación?

   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos DA1

Lista de Revisión de Defectos en el Diseño de Alto Nivel (DA1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

a) Requerimientos generales y diseño

1. ¿En el diseño es identificada la falta de requerimientos ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿En el diseño se identifican requerimientos ambiguos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿En el diseño se identifican requerimientos extraños?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿En el diseño se identifican requerimientos imposibles de probar?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿En el diseño se identifican requerimientos insinuados?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Hay funciones/clases que están faltando?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Hay funciones/clases extrañas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Hay funciones/clases imprecisas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿Hay funciones/clases ambiguas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

10. ¿Hay funciones/clases incorrectas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

11. ¿Las desviaciones de los requerimientos están documentadas y aprobadas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12.¿Están documentadas todas las asunciones?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Están documentadas las decisiones del diseño más importantes?

1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿El diseño es consistente con las decisiones de diseño tomadas?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿En el diseño se plantea adecuadamente los requerimientos de tiempo real?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿En el diseño se plantea adecuadamente el desempeño (la memoria y calendarizado de tareas) ?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. ¿En el diseño se plantea adecuadamente la capacidad (CPU y memoria) ?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿En el diseño se plantea adecuadamente los requerimientos de la base de datos ?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

19. ¿En el diseño se plantea adecuadamente la interfaz de usuario?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Especificación de la interfaz y funciones

1. ¿Están identificadas y documentadas las dependencias de las funciones?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Está especificado el proceso claramente para cada función/clase?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Están especificados los requerimientos para verificación de error, manejo de error, y recuperación?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Están identificadas las interfases para cada función/clase?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Las interfaces son consistentes con los módulos manejados ?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Se especifican las interfaces con suficiente detalle para que puedan ser verificadas?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

c) Convenciones

1. ¿El diseño sigue las convenciones establecidas?
1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos DB1

Lista de Revisión de Defectos en el Diseño de Bajo Nivel (DB1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4.La Mayoría 5.Todos

1. ¿En el diseño detallado, los módulos o las interfases satisfacen los requerimientos?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se han identificado módulos o interfases con problemas debido a los requerimientos perdidos?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se han identificado módulos o interfases con problemas debido a los requerimientos contradictorios?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se han identificado módulos o interfases con problemas debido a los requerimientos imposibles de probar?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se han identificado módulos o interfases con problemas debido a los requerimientos insinuados?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se reconocen los requerimientos de alto nivel en el diseño detallado de módulos o interfases?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. En el diseño detallado, ¿se han identificado los problemas con el diseño de alto nivel?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Están descritas todas las funciones completa, precisa y detalladamente?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Todas las interfases se describen completa y precisamente, incluso las palabras clave o parámetros posicionales, campos , atributos, intervalos y límites?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Son consistentes y están completos todos los documentos del diseño detallado?
    1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Estructura e interfaces

1. ¿En el sistema y subsistemas, se han identificado todos los módulos en el modelo de arquitectura de sistema?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿El diseño esta a un nivel de descomposición suficiente para identificar todos los módulos?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se han identificado y definido en términos precisos todas las interfaces entre el sistema y módulos ?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Los niveles subsecuentes de descomposición resultan en niveles subsecuentes de detalle?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Los módulos están realizando más de una función específica?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

c) La lógica

1. ¿Hay errores en la lógica?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se prueban todos los valores únicos?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se prueban todas las posiciones de los valores?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se inicializan los contadores?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Son inicializadas las variables y áreas de datos antes de utilizarlas?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Realizan los módulos la apertura y cierre del procesamiento de tablas correctamente?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Realizan los módulos la toma de decisiones de tablas lógicas correctamente?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Los módulos tienen precisión y exactitud en sus cálculos?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿ La secuencia del procesamiento del mensaje es correcta?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

10. ¿Hay errores en el manejo de los datos?
    1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

11. ¿Hay errores en el manejo de tablas?
    1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. ¿Se actualizan correctamente los datos o las tablas?
    1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Se inicializan correctamente los datos o las tablas?

1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Hay inconsistencia o atributos de datos inválidos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿Están los procedimientos de llamadas y respuestas de la interfase definidas correctamente?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿Están los procedimientos de llamadas y respuestas de parámetros definidos correctamente?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. ¿Se definen las llamadas y respuestas de los parámetros correctamente?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿Es correcta la sintaxis?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

d) Desempeño
1. ¿Son los presupuestos de memoria y tiempo razonables y alcanzables?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

e) Manejo de error y recuperación
1. ¿Se definen condiciones dónde la probabilidad de error es alta?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se definen condiciones donde los resultados de un error serían fatales al sistema?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Son entendibles los mensajes de error?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se permite la recuperación de errores de módulos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿ Se permite la recuperación de errores del sistema operativo?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿ Se permite la recuperación de errores de interrupciones?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Se permite la recuperación de errores del hardware?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

f) Extensibilidad de la prueba
1. ¿Es entendible el diseño (fácil de leer, y seguir la lógica)?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Mantenible ?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se puede probar?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos ES1

Lista de Revisión de Defectos en Entrada/Salida (ES1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿ Se identifican todas las acciones del usuario final?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se identifican con suficiente detalle tal que la contribución de las salidas, puedan ser relacionadas a esas acciones?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Están relacionadas las salidas del sistema a las acciones especificas del usuario?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿El usuario final comprende los reportes de salidas y pantallas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿El usuario final comprende el tipo de lógica y cálculos para producir la salida?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿El usuario identifica las contribuciones de las salidas para las acciones tomadas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿El usuario final identifica si las acciones tomadas son correctas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Se identifican relaciones entre la salida del sistema y acciones definidas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿El usuario final tiene información adecuada para tomar una acción?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿El usuario final hace un número anormal de acciones incorrectas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos CE1

Lista de Revisión de Condiciones Estéticas (CE1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Es correcto el color del fondo de la pantalla?.
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Es correcto el color del apuntador del campo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Es correcto el color del fondo del campo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Es correcto el color del apuntador del campo en modo solo lectura?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Es correcto el fondo del campo en modo solo lectura?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se especifican todos los apuntadores de la pantalla en el tipo de letra de la pantalla?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se especifica el texto en todos los campos en el tipo de letra de la pantalla?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Se alinean perfectamente todos los apuntadores del campo en la pantalla?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Se alinean perfectamente todas las cajas de edición en la pantalla?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Se alinean perfectamente todas las cajas de grupo?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Es modificable el tamaño de la pantalla?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. ¿Es minimizable la pantalla?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
13. ¿Se justifican a la izquierda todos los caracteres o campos alfanuméricos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿ Se definen los valores por omisión para los campos alfanuméricos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. Se justifican a la derecha todos los campos numéricos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. Se definen los valores por omisión para los campos numéricos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. ¿Se verificó la ortografía de la ayuda en texto de las pantallas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿Se revisó la redacción de los mensajes de error en las pantallas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

19. ¿Todas las ventanas tienen una vista consistente?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

20. ¿Todas las cajas del diálogo tienen una vista consistente?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos P1

Lista de Revisión de Presentación (P1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Se requieren entradas de fechas en el formato correcto?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se evitan los campos de solo lectura en la secuencia del Tab?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se evitan los campos deshabilitados en la secuencia de Tab?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se activa la microayuda al poner el cursor en la caja de texto y pulsar el botón derecho con el ratón?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se activa la microayuda al poner el cursor en campos de solo lectura y pulsar el botón derecho con el ratón?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se posiciona el cursor en el primer campo de la entrada o control cuándo la pantalla se abre?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Hay un botón por omisión especificado en cada pantalla?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Trabaja correctamente el botón por omisión?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Cuando ocurre un mensaje de error y es cancelado posteriormente se regresa al campo en donde se presentó el error?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Cuándo el usuario presiona Alt + Tab se mueve la pantalla de aplicación en aplicación?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Todos los campos de las cajas de edición tienen la longitud adecuada?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos V1

Lista de Revisión de Validación (V1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4.La Mayoría 5.Todos

1. ¿Se definió un mensaje de error de usuario por cada falla de validación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se pide al usuario que corrija las entradas que fallaron en la validación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se estableció una validación múltiple para los campos que así lo requiren?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Es consistente la validación en todos los campos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se probaron los valores mínimos, máximos y medios del intervalo de valores permitidos en los campos numéricos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se probó el límite especificado de los caracteres para todos los campos alfanuméricos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se verificó si ese límite es correcto para el tamaño de la base de datos especificada?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Es requerida la captura del usuario en todos los campos obligatorios?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. Se requiere la captura obligatoria de las columnas de valor nulo en las tablas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos E1

Lista de Revisión de Defectos de Errores Funcionales (E1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Se detectaron errores funcionales en la lluvia de ideas con los usuarios finales?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se detectaron condiciones de errores estructurales en la lluvia de ideas con el personal del proyecto?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se detectaron condiciones de errores funcionales para rechazar códigos inválidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se detectaron condiciones de errores funcionales para rechazar los valores fuera del intervalo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se detectaron condiciones de errores funcionales para rechazar fechas inválidas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se detectaron condiciones de errores para rechazar transacciones no autorizadas de valores inválidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se detectaron condiciones de errores para rechazar transacciones no autorizadas de clientes inválidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Se detectaron condiciones de errores para rechazar transacciones no autorizadas de productos inválidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Se detectaron condiciones de errores para rechazar transacciones no autorizadas de precios inválidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Se detectaron condiciones de errores funcionales para datos alfabéticos en campos numéricos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Se detectaron condiciones de errores funcionales para espacios en blanco en campos numéricos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. ¿Se detectaron condiciones de errores funcionales para valores negativos en campos positivos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
13. ¿Se detectaron condiciones de errores funcionales para números campos alfabéticos
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Se detectaron condiciones de errores funcionales para espacios en blanco en campos alfabéticos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿Se detectaron condiciones de errores funcionales para valores más grandes que el campo?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿Se detectaron condiciones de errores funcionales para más grandes de los campos totales?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. ¿Se detectaron condiciones de errores funcionales para acumulación adecuada de totales?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿Se detectaron condiciones de errores funcionales para transacciones incompletas, pérdidas de campos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

19. ¿Se detectaron condiciones de errores funcionales para valores nuevos que por el momento no son aceptables ?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

20. ¿Se detectaron condiciones de errores funcionales para transacciones después de la fecha?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

21. ¿Se detectaron condiciones de errores funcionales para cambio de valores que afecten una relación?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

22. ¿Los mensajes de error son representativos de las condiciones de error?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

23. ¿Se realizaron pruebas para detectar si se ejecutaron todas las instrucciones y hubo condiciones de error?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

24. ¿Se realizaron pruebas para detectar si se ejecutaron todos los caminos y hubo condiciones de error?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

25. ¿Se realizaron pruebas para detectar si se utilizaron las tablas internas y hubo condiciones de error?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

26. ¿Se realizaron pruebas para detectar si se ejecutaron todos los ciclos y hubo condiciones de error?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

27. ¿Se realizaron pruebas para verificar los mensajes de advertencia?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos B1

Lista de Revisión de Defectos en Búsquedas (B1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Se identifican todos los caminos lógicos internos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se identifica toda la lógica de la búsqueda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se identifican todas las rutinas de la autorización?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se identifica toda la lógica del proceso que requiere una búsqueda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se identifican rutinas de búsqueda de bases de datos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se identifican las lógicas de búsquedas complejas ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se realizan las condiciones de prueba para todas las condiciones de la búsqueda precedentes?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿ Se entrevistó al usuario final para determinar el tipo de búsqueda que podrían encontrarse en el futuro?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos PR1

Lista de Revisión de Defectos en Procedimientos (PR1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1- Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Se validaron los procedimientos de instalación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se validaron los procedimientos de consulta?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se validaron los procedimientos de subir archivos ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se validaron los procedimientos de actualización?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se validaron los procedimientos de respaldo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se validaron los procedimientos de almacenamiento fuera del lugar?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se validaron los procedimientos de recuperación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Se validaron los procedimientos operación de terminales?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Se validaron los procedimientos necesarios para operar la terminal cuando la computadora se caiga?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Se validaron los procedimientos para capturar datos cuando las terminales están caidas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos C1

Lista de Revisión de Defectos en Codificación (C1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Es incorrecto el cálculo aritmético?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Son incorrectas las trayectorias?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Hay ciclos indefinidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se violan las reglas del lenguaje de programación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se violan los estándares de programación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿El programador interpreta mal las estructuras del lenguaje?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Existen errores de formato en las entradas/salidas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Terminan los subprogramas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Hay errores al accesar o procesando datos de entrada?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Hay errores en el procesamiento de salidas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Hay errores de procesamiento de los mensajes de error?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. ¿Hay errores de indexación?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Hay errores de procesamiento iteractivo?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Hay errores de manipulación de bits?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿Hay errores de sintaxis?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿Hay errores de inicialización?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. ¿Hay confusión en el uso de parámetros?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿Hay errores en los contadores de ciclos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

19. ¿Son manejados correctamente los resultados de decisión?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

20. ¿Se dan nombres múltiples a las variables o no se definen?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

21. ¿Ocurren errores fuera de los nombres de variables?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

22. ¿Se declaran incorrectamente tipos y dimensiones?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

23. ¿Hay confusión en los nombres de las librerias del programa?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

24. ¿Hay errores de compilado?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

25. ¿Hay errores bajo flujo (sobre flujo negativo) de punto flotante?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

26. ¿Hay errores sobre flujo de punto flotante?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

27. ¿Se permite el punto flotante y división por cero?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

28. ¿Hay errores de secuencia?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

29. ¿Hay errores al guardar y restaurar los registros en los sistemas de tiempo real?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

30. ¿Es correcta la conección de la interfase del software al sistema de hardware ?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

a) Campos

1. ¿Se validaron todos los módulos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se actualizan los campos adecuadamente?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Es suficiente el tamaño del campo para guardar los totales?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Todas las referencias al campo usan el nombre del campo adecuado?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Están validados los contenidos de los campos que son restringidos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Se establecieron reglas para identificar y procesar datos inválidos de los campos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿ Es incluida en las condiciones de la prueba una gama de valores válidos típicos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Se han probado los valores máximos y mínimos para los campos numéricos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿Se han probado los campos numéricos que tienen valor cero?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

10. ¿Se preparó una condición de prueba para números negativos en campos numéricos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

11. ¿Se preparó una condición de prueba vacía para los campos alfabéticos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. ¿Para los campos alfabéticos o alfanuméricos, se preparó una condición de prueba de mayor longitud que el campo para verificar el proceso del truncamiento?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Se probaron todas las condiciones válidas basándose en el diccionario de datos?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Se revisó la especificación del sistema para determinar si se prueban todas las condiciones válidas?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Registros

1. ¿Se prepararon las condiciones para probar el procesamiento del primer registro?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se determinaron las condiciones para validar el procesamiento del último registro?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se procesaron correctamente todos los registros múltiples para la transacción?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Están probadas todas las variaciones en tamaño del registro?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Pueden procesarse dos archivos con el mismo identificador ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Se puede recuperar el primer registro guardado en un archivo del almacenamiento ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Se puede recuperar el último registro guardado en un archivo del almacenamiento?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Se pueden guardar todos los registros de la entrada?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿ La interconección de los módulos tiene el mismo identificador para cada tipo del registro?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

10. ¿La definición del almacenamiento de registro esta conforme a la definición del sistema de registros?
    1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

11. ¿Los formatos de los registros coinciden con los formatos usados en archivos creados por otros sistemas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

c) Ciclos

1. ¿Se han probado todas las trayectorias en ambas direcciones?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se han ejecutado todas las declaraciones?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se han probado todos los ciclos ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se han probado todas las iteraciones de cada vuelta ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Se han probado todos los caminos de la ejecución?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Todos los subprogramas y bibliotecas se han llamado y se han ejecutado durante las pruebas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos AR1

Lista de Revisión de Defectos para Archivos (AR1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Hay una condición para probar cada archivo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Hay una condición para probar la interfase de cada archivo con cada módulo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Hay una condición para validar que se usará la versión correcta de cada archivo?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Hay una condición para probar que los datos almacenados en un archivo regresarán intactos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Hay condiciones para validar que cada archivo se cerró adecuadamente después de que el último registro es procesado?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Hay condiciones para validar que cada tipo del registro puede procesarse intactamente de inicio a fin del sistema?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Hay condiciones para validar que se procesan todos los archivos introducidos a través del sistema?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Hay condiciones para validar que los archivos que están montados se cierran adecuadamente en el final del proceso?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿Hay condiciones de prueba para crear un archivo del cual no existen registros previos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

10. ¿Hay condiciones para validar el cierre de un archivo cuándo se han borrado todos los registros en el archivo?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos I1

Lista de Revisión de Diseño de Interfaz (I1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4.La Mayoría 5.Todos

1. ¿Se inicializa la aplicación con doble pulsación en el icono de inicio?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Al cargar la aplicación se muestra el nombre de la aplicación, número de la versión, y una representación pictórica más grande que el icono de inicio?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Tiene el mismo título la ventana principal de la aplicación y el icono en el administrador de programas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Al cerrar la aplicación, se muestra en la caja de diálogo el mensaje " Esta usted Seguro"?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Al cargarse la aplicación se muestra el progreso del despliegue como alternativa?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Todas las pantallas tienen un botón de Ayuda ó F1?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿La ventana de la aplicación se reduce al pulsar el botón de minimizar y se coloca en la parte inferior de la pantalla?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. En los menús, ¿se selecciona la siguiente opción cuando se presiona la tecla TAB?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. En los menús, ¿ se selecciona la opción anterior cuando se presiona las teclas SHIFT/TAB?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. En las cajas agrupadas, ¿el orden del TAB es de izquierda a derecha, y de arriba a abajo?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Se resalta el texto del campo que consigue el TAB?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. ¿Se cambia la ayuda en línea al moverse sobre los iconos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Se verificó la escritura de la ayuda en línea de los iconos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Se justifica todo el texto hacia la izquierda?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

a) Cajas de texto

1. ¿Todas las cajas de texto pueden tener el cursor?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿En todas las cajas de texto se modifica el cursor de flecha a barra, excepto en las cajas no actualizables donde el texto es gris?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se introduce texto en todas las cajas de texto?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se verificó la anchura física del campo tecleando caracteres hasta inundar la caja de texto?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Se introducen caracteres inválidos en todos los campos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Botones de radio

1. ¿Las opciones se activan pulsando sobre la opción?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se excluyen las demás opciones al seleccionar la ultima opción
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

c) Casillas de verificación

1. ¿Las opciones se activan y desactivan con un clic en la casilla?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

d) Botones comando

1. ¿Los botones que aumentan la pantalla con detalles adicionales, están seguidos de tres puntos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Todos los botones tienen una letra de acceso indicada por una letra subrayada en el texto del botón, excepto los botones Ok y Cancel?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se activan los botones al presionar la tecla Alt + la letra?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Existen duplicaciones en las letras escogidas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Se activa cada botón al presionarse?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Se activa cada botón con la barra espaciadora?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

7. ¿Se activa cada botón con la tecla Enter?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

8. ¿Se definió por cada pantalla un botón por omisión?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

9. ¿Se activa el botón cancel de la pantalla, al presionar la tecla Esc?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

e) Cajas de lista

1. ¿Se despliegan los elementos de la lista al presionar la flecha?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Todas las cajas de lista tienen una barra de desplazamiento cuando lo requieren?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Al presionar cualquier letra se posiciona en el primer elemento de la lista con esa letra?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se despliega la lista de elementos al presionar las teclas Ctrl y F4?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Están ordenados alfabéticamente los elementos de la lista con excepción del espacio en blanco que está en la parte de arriba o en el fondo de la lista?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6. ¿Aparece únicamente un espacio en la lista de elementos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

f) Cajas de diálogo

1. ¿Se abre la carpeta que esté seleccionada presionándola dos veces?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Es idéntico el comportamiento al teclear dos veces sobre la carpeta que seleccionar una carpeta y el botón Open?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿La barra de desplazamiento permite ver todos los valores en la caja?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos CG1

Lista de Revisión de Condiciones Generales de la Interfaz (CG1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4.La Mayoría 5.Todos

1. ¿Existe ayuda en el menú?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se incluyen en cada menú los comandos y las opciones correctas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Son correctos los comandos que corresponden a los botones en las barras de herramientas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se considera para cada comando importante del menú una tecla alternativa?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se accede a las opciones de la lista de desplegable con las teclas seleccionadas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿La tecla ESC deshace cualquier cambio que ha sido hecho ó genera un mensaje de cautela de "se perderán los cambios - sí / no"
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Son iguales las funciones del botón Cancel y la tecla Esc?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Es el ámbito de cada comando exclusivo a una ventana o a una caja de diálogo particular , es decir estos comandos no trabajan detrás de la pantalla activa?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Todos los botones OK y Cancel están agrupados por separado de otros botones comandos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Las etiquetas tienen nombres predefinidos, pero son significativas a los usuarios del sistema?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Todas las ventanas y/o cajas de diálogo tienen un valor por omisión claramente marcado qué se invoca cuando la tecla Enter es presionada?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. ¿Todas las cajas de opción, botones de opción y botones de comandos están agrupados en áreas claramente demarcadas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. ¿Se accesan consistentemente através del teclado y del mouse todas las opciones?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Tienen las pantallas/ ventanas una apariencia ordenada?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. ¿Cuando se presionan las teclas Ctrl + F6 se abre la próxima ventana en la aplicación?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. ¿Cuando se presionan las teclas Shift + Ctrl + F6 se abre la etiqueta anterior dentro de la ventana ?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. ¿Se agrega una barra de desplazamiento en las cajas de lista que tienen más de ocho opciones?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

18. ¿Es el mismo tipo de letra en toda la aplicación?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

19. ¿Cuando se presionan las teclas Alt + F4 se cierra la ventana y se devuelve a la pantalla principal o la pantalla anterior?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos N1

Lista de Revisión de Navegación (N1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5.Todos

1. ¿Son correctos todos los accesos a las pantallas correspondientes a través del menú?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Son correctos todos los accesos a las pantallas a través de la barra de herramientas ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se pueden accesar las pantallas a través de los botones en las pantallas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se abren varias instancias de una pantalla al mismo tiempo ?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos ID1

Lista de Revisión de Integridad de los Datos (ID1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4.La Mayoría 5.Todos

1. ¿Se almacenan los datos cuándo la ventana se cierra por doble clic en la caja cerrar?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se verifican longitudes máximas del campo para asegurar que no hay ningún carácter truncado?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se definen valores por omisión en los campos en donde se requiere un valor?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se verifican los valores máximos y mínimos para los campos numéricos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se almacenan correctamente los valores de los campos numéricos negativos en la base de datos y tiene sentido para el campo aceptar números negativos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se almacenan integramente los datos en la base de datos es decir son libres de truncamientos las cadenas y de redondeo los valores numéricos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos T1

Lista de Revisión de Defectos en Transacciones (T1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Se validaron los archivos de estados vacíos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se validaron los archivos de transacción de estados vacíos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se validaron los registros de estados perdidos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se validaron los registros de estados duplicados?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se validaron las tablas de estados vacios?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se validaron la cantidad de estados insuficientes?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se validaron los balances de estados negativos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Se validó la salida de estados duplicados?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Se validó el estado de introducir transacciones duplicadas?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Se validaron las actualizaciones de estados coexistentes (dos términales que llaman al mismo tiempo al mismo registro maestro)?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Se validaron los estados en que hay más demandas para el servicio o productos que los servicios y productos para soportarlos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos IN1

Lista de Revisión de Instalación (IN1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

a) Instalación

1. ¿Se especificaron las capacidades de la persona que realiza la instalación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Esta documentado el proceso de instalación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Los pasos del procedimiento de instalación son específicos y concisos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Esta completo el procedimiento de instalación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se especifica la instalación para diferentes ambientes: plataformas, software, hardware etc?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se consideró la instalación: típica, mínima y customizada?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Se consideró la instalación desde diferentes medios: disquets, DVD-rom, CD-rom, disco duro , bajo web?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Se verifica sobre los requerimientos de espacio para la instalación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Se crea el directorio destino adecuadamente, si no existe?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Se descompactan/copian los archivos al directorio destino adecuadamente?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿ Existe un indicador del avance de la instalación?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. ¿Se modifican los archivos de configuración?

1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. Al terminar la instalación, ¿existe una indicación de que la instalación ha sido llevada exitosamente?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. ¿Se pide la reinicialización de la computadora para que los cambios en la configuración surjan efecto?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Desinstalar

1. ¿Al desinstalar se remueve el directorio destino?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿Se borran los archivos del sistema?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. ¿Se cuestiona sobre el borrado de los archivos compartidos por otras aplicaciones?
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se permite la desinstalación desde la opción agregar/remover programas del menú?.
1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos AY1

Lista de Revisión de Ayuda (AY1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4.La Mayoría 5.Todos

a) Consistencia

1. ¿Tiene sentido la estructura del sistema de ayuda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Hay flexibilidad en las opciones disponibles al usuario?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Es fácil la navegación en el sistema de ayuda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se organizó jerárquicamente el sistema de ayuda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Es sensible al contexto?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Son consistentes los métodos de búsqueda y navegación?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Es consistente el vocabulario a través del sistema?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Son consistentes los comandos en todo el sistema de ayuda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Son los tipos y tamaños de letra consistentes en todo el sistema de ayuda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Son consistentes los enlaces?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. ¿Son consistentes las pantallas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. ¿Son consistentes los formatos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

b) Contenido

1. ¿Se consideran los distintos niveles de habilidad del usuario en el sistema de ayuda?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2. ¿El sistema de ayuda es tan valioso como su contenido?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Cada instrucción es exacta?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

4. ¿Se ha revisado la gramática y ortografía del contenido?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5. ¿Se ha omitido información importante?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos RNF1

Lista de Revisión de Defectos de Requerimientos No Funcionales (RNF1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5.Todos

1. ¿Se identifican los requerimientos no funcionales del software?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se clasificaron en categorías estos requerimientos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se han desarrollado condiciones de prueba para requerimientos muy importantes?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. Para la exactitud, ¿son exactas y completas las funciones?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. Para la integridad de archivo , ¿está validada la integridad de cada archivo o subesquema?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. Para la autorización, ¿se definieron procedimientos para la autorización de cada transacción?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. Para la auditoria del camino, ¿se verifica por las condiciones de prueba que cada transacción pueda reconstruirse?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. Para la continuidad del proceso, ¿se puede recuperar el sistema y las transacciones capturadas o procesadas durante el periodo de la recuperación dentro de un periodo de tiempo razonable?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. Para servicio, ¿se satisfacen las necesidades del usuario de operación y tiempos de la respuesta?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. Para el control de acceso, ¿esta limitado el acceso a los usuarios autorizados?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. Para la fiabilidad, ¿es incorrecto, incompleto u obsoleto el procesamiento de los datos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

12. Para la facilidad de uso, ¿se puede usar el sistema eficaz y económicamente?

1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

13. Para la mantenibilidad, ¿se puede modificar o ampliar el sistema con el esfuerzo razonable y oportunamente?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

14. Para la portabilidad, ¿se puede cambiar eficazmente a otras plataformas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

15. Para acoplamiento, ¿se puede integrar con otros sistemas?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

16. Para el desempeño, ¿es el desempeño del software aceptable?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

17. Para la facilidad de operación, ¿es capaz el personal de operaciones de operar el software económica y eficazmente?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Lista de revisión de defectos D1

Lista de Revisión de Defectos de Desempeño del sistema (D1)

Proyecto_____**Versión**_____

Etapa del Proyecto _____**Tipo de Prueba** _____

Autor de la Prueba _____

Persona que lo aplica _____ **Puesto que ocupa** _____

1. Ninguno 2. Varios 3. La Mitad 4. La Mayoría 5. Todos

1. ¿Se identifican las capacidades de desempeño de los datos entrada?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. ¿Se identifican las capacidades de desempeño de actualización?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. ¿Se identifican las capacidades de desempeño del tiempo de respuesta?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. ¿Se identifican las capacidades de desempeño en el manejo del error?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. ¿Se identifican las capacidades de desempeño en la generación de reportes?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. ¿Se identifican las capacidades de desempeño de los cálculos internos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. ¿Está identificada la velocidad de la transmisión de las comunicaciones en línea?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. ¿Está identificada la eficiencia del sistema de administración de la base de datos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. ¿Está identificado el respaldo para entrada de datos?
   1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. ¿Están identificadas las capacidades del almacenamiento de archivos?
    1☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Formato de registro de defectos

Lista que se utiliza para el registro de defectos en cada una de las fases del ciclo de desarrollo del sistema.

Proyecto_____**Versión**_____
Etapa del Proyecto _____**Tipo de Prueba** _____
Autor de la Prueba _____
Persona que lo aplica _____ **Puesto que ocupa** _____

Número	Descripción	Ubicación	Tipo	Característica	Severidad

Los elementos que componen la lista de registro de defectos son:

Descripción. Describe los defectos (inconsistencias, errores u omisiones) que se presentan en el proceso de pruebas.

Ubicación. Indica el documento o módulo del sistema en donde se presenta el defecto.

Tipo. Representa el tipo de defecto encontrado. Los cuales se clasifican en errores de arquitectura, conectividad, consistencia, integridad, documentación interfaz gráfica, instalación, seguridad y estándares y convenciones.

Característica. Se asocia al tipo de defecto y se clasifican en ambigüedad, claridad, omisión y organización.

Severidad. Impacto que el defecto provoca en el sistema y puede ser crítico, mayor y menor

Aplicación de la guía a SCART

Para la evaluación de la guía de pruebas se seleccionó el sistema SCART (Sistema Colaborativo de Apoyo a Revisiones Técnicas). SCART es un sistema colaborativo que sirve para realizar revisiones técnicas de los productos que surgen durante el proceso de desarrollo de software. El primer prototipo del sistema se terminó y se aplicaron pruebas de aceptación, con la idea de verificar que cumplía con los requerimientos estipulados por el cliente.

En este apéndice se desarrolla un plan de pruebas de acuerdo a la guía de pruebas propuesta con el objetivo de ampliar los resultados de la primera evaluación del sistema SCART.

E.1. Plan de pruebas

En el grupo de ingeniería de software del Departamento de Ciencias de la Computación de la División de Física Aplicada del CICESE se desarrolló el sistema SCART "Sistema Colaborativo de Apoyo a Revisiones Técnicas" como parte de un trabajo de tesis de maestría.

SCART permite realizar reuniones de revisiones técnicas síncronas con grupos de revisores colocalizados y/o distribuidos, similares a las reuniones de revisión tradicionales (cara a cara), apoyados en la tecnología de cómputo, este tipo de reuniones electrónicas están orientadas a la revisión de documentos, no a la persona que los realiza, la idea es reducir el estrés de los participantes durante la reunión y los costos de viaje, cuando se trata de la participación de revisores distribuidos geográficamente.

Las pruebas a realizarse tienen la finalidad de encontrar la mayor cantidad de defectos en los cuales el sistema tenga una respuesta diferente a la esperada de acuerdo a su funcionalidad específica.

Objetivos de las pruebas

Las pruebas realizadas deben cumplir los requerimientos planteados en el análisis del sistema, mediante el mapeo supervisado de cada uno de éstos hacia las demás etapas del ciclo de desarrollo del sistema.

Descripción del sistema

SCART divide la reunión de revisión tres etapas: planeación , preparación y reunión de revisión.

La etapa de planeación consiste en registrar agendas, documentos a revisar, los defectos encontrados y guías para revisión de documentos.

En la etapa de preparación el moderador proporciona a los revisores los documentos que serán revisados y los revisores se preparan individualmente para la revisión llenando listas de defectos.

En la reunión de revisión el moderador conduce la reunión presenta a los participantes y sus respectivos roles y asigna el control de la reunión al productor para la presentación del material.

Elementos de la prueba

Análisis del sistema. Los elementos que componen el análisis del sistema son el análisis de requerimientos y su especificación. El análisis de requerimientos es la descripción de las características que debe cubrir el sistema y la especificación es la representación de éstos en tres modelos diferentes: el modelo de objetos (diccionario de datos, relaciones entre clases y diagramas de objetos), el modelo dinámico y el modelo funcional.

Diseño del sistema. Los elementos que componen el diseño son el tipo de arquitectura, los subsistemas, el almacenamiento y el manejo de los datos y la metodología para modelar los requerimientos.

Sistema de software. Los elementos a probar en el sistema son seguridad, instalación, el sistema de ayuda y la funcionalidad de la interfaz.

Descripción de la prueba

P1. Análisis: Consiste en probar la descripción de los requerimientos y los diagramas correspondientes a la especificación de estos.

P2. Diseño: Consiste en probar si se realizó correctamente la traducción del análisis del sistema al diseño del mismo, y si se definieron los suficientes detalles para permitir su codificación.

P3. Sistema: consiste en probar la instalación, ayuda, seguridad, funcionalidad e interfaz del sistema.

Los errores se van a plasmar en listas de errores, cuyo contenido es: descripción del defecto, ubicación, tipo, entre otros.

Documentación del sistema

Los documentos utilizados para el desarrollo de este plan de pruebas son la tesis de "Sistema Colaborativo de Apoyo para Revisiones Técnicas en Salas Electrónicas de Reuniones" (Gómez, 1998) y el reporte técnico "Sistema Colaborativo de Apoyo para Revisiones Técnicas "SCART": Análisis y Diseño (Gómez, 1998).

Documentos de la prueba

Los documentos que resultan de este proceso de pruebas son plan de pruebas, documentos de la aplicación de la prueba, reportes, seguimiento y métricas.

Mecanismos de prueba

El mecanismo de pruebas principalmente utilizado son las listas de revisión de errores. Entre las cuales se encuentran listas de revisión del análisis, del diseño, de la instalación entre otros.

Presupuesto

Debido a la naturaleza de este trabajo, no se asignó capital a la realización de la prueba.

Necesidades de personal y entrenamiento

Para el desarrollo de la prueba se requirió del apoyo de la creadora de SCART y de dos estudiantes de la maestría en Ciencias de la computación del CICESE. Para la selección del personal de apoyo a la aplicación de la prueba se le pidió que tenga experiencia en el uso de una computadora personal, conocimientos de las metodologías de modelado OMT y UML, conocimientos generales en computación o áreas afines y experiencia en revisiones técnicas.

Requisitos de software y hardware

Los requisitos de software y hardware para la realización de las pruebas a SCART son:

Hardware. Computadora pentium II o superior con 64 Mbytes en RAM (mínimo) y 50 Mbytes de en disco duro.

Software. Un navegador web como el netscape communicator ó internet explorer.

Calendario

La tabla V muestra el calendario de pruebas a realizar al sistema SCART en el periodo del 14 de mayo al 23 de julio. El tiempo en que se llevaron a cabo las pruebas del análisis fueron cuatro semanas, en el diseño se tomaron tres semanas, por último en el sistema se tomaron cuatro semanas.

Tabla V. Calendario de pruebas.

Tipos de pruebas	**Mayo**			**Junio**				**Julio**			
	14	21	28	4	11	18	25	2	9	16	23
P1. Análisis	X	X	X	X							
P2. Diseño					X	X	X				
P3. Sistema							X	X	X	X	X

E.2. Aplicación de las pruebas

La aplicación de la prueba es el desglose del plan de pruebas en el diseño de la prueba, casos y procedimientos de prueba. A continuación, se muestra una descripción de los documentos que surgieron como resultado de este proceso.

E.2.1. Diseño de las pruebas

Requerimientos funcionales

Los requerimientos funcionales del sistema se encuentran en el reporte técnico correspondiente. Algunos de ellos se mencionan a continuación:

- El sistema requiere realizar el *registro de proyectos* de desarrollo de software.
- El sistema deberá apoyar las tres primeras etapas del proceso de revisiones técnicas, que son*: planeación, preparación, y reunión de revisión*.
- En la etapa de *planeación* deberá proporcionar los mecanismos para la calendarización de la reunión a través de una *agenda*. El sistema permitirá asignar privilegios y responsabilidades a un usuario o grupo de usuarios dependiendo del *rol* que vaya a desempeñar durante la reunión.
- La etapa de *preparación* deberá permitir que los miembros del grupo de revisores trabajen de manera independiente, en el registro de documentos, la revisión individual de documentos y elaboración de las *listas de defectos* encontrados en dichos documentos.
- El sistema deberá proporcionar capacidad *consultar la agenda de la reunión*, el *listado de documentos* y el *listado* defectos.
- En la tercera etapa conocida como *reunión de revisión*, las sesiones deben ser formales, de manera que la interacción entre los participantes sea estructurada y flexible mediante el seguimiento de una secuencia de acciones planteadas en una agenda.
- Los usuarios podrán revisar el (los) documento(s) especificado(s) en la agenda y hacer anotaciones sobre el (los) mismo(s).

Funciones estructurales

Los usuarios pueden realizar una serie de actividades en el sistema dependiendo del rol que tengan. Las actividades que se pueden realizar se mencionan a continuación:

- Abrir una sesión.
- Registro del administrador del proyecto.
- Registro del proyecto y sus participantes.
- Registro de una agenda.
- Registro de un documento.
- Registro de guías de revisión
- Registro de reporte de revisión
- Registro de lista de defectos
- Presentación de un documento.
- Control de Piso.
- Anotaciones sobre elementos de un documento.

Características a ser probadas

Se van a probar los elementos que corresponden a cada pruebas. Las características a evaluar se mencionan a continuación:

Accesibilidad. Libertad para introducirse, navegar y salirse del sistema.

Coherencia. Se evalúa si el sistema realiza la funcionalidad que se describe en la interfaz.

Compatibilidad. El sistema pueda ejecutarse en una variedad de computadoras.

Comprensibilidad. Se entienden la estructura del producto, la ayuda del sistema y la documentacion.

Configuración. El sistema opera adecuadamente cuando el software o hardware está configurado de manera requerida.

Disponibilidad. El sistema proporciona la información solicitada en cualquier momento.

Instalabilidad. Los procedimientos de instalación llevan a resultados correctos.

Recuperación. El sistema reúne los requisitos para la recuperación después de una falla.

Seguridad. El sistema pueda restringuir los accesos a personal autorizado.

Tensión. El sistema puede atender cargas máximas hasta que falle.

Volumen. El sistema puede atender una gran cantidad de demandas de los usuarios.

Enfoque

En el análisis se toman los requerimientos y se prueban con la lista de revisión de defectos R1 (ver apéndice C.1) de tal manera que se identifiquen redundancias, omisiones, entre otros. A continuación se hace un mapeo hacia las especificaciones verificando que todos estén considerados en el diagramado de los modelos de objetos, dinámico y funcional. Cada diagrama es probado individualmente y los errores se anotan en el formato de registro de defectos (ver apéndice D).

En el diseño se prueba la eficiencia de la traducción de los modelos de análisis a diseño utilizando la lista de revisión de defectos DA1 (ver apéndice C.3). Se prueba la división en subsistemas y la estructura de los módulos e interfaces utilizando la lista de revisión de defectos DB1 (ver apéndice C.4). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

El sistema se prueba en tres partes. Primero, se prueba la instalación del sistema utilizando la lista de revisión IN1(ver apéndice C.19). En segundo lugar, se prueba la ayuda del sistema para esto, se utiliza la lista de revisión AY1 (ver apéndice C.20). Por último, se prueba la interfaz utilizando la lista de revisión I1(ver apéndice C.14). Los errores se anotan en el formato de registro de defectos (ver apéndice D).

Por último, se ejecutan casos de prueba que prueban la funcionalidad del sistema, se realizan en el orden especificado en el procedimiento de pruebas.

E.2.2. Procedimiento de pruebas

Propósito

El procedimiento describe los requerimientos para preparar el ambiente de pruebas y el orden de ejecución de las pruebas especificadas en el diseño de las pruebas.

Requerimientos especiales

El servidor debe estar configurado y funcionando según lo dispuesto en la guía de instalación.

El cliente debe tener un navegador de web que contenga aplicaciones que soporten la ejecución de programas java.

Pasos del procedimiento

Especificar la bitácora en donde se registra la ejecución de la prueba

Ejecutar los casos de prueba en forma secuencial. Apoyarse del manual de usuario para realizar la prueba.

Anotar los sucesos inesperados y errores presentados al ejecutar los casos de prueba en el reporte de incidentes.

Realizar el resumen de las pruebas y el reporte de transmisión

E.2.3. Casos de prueba

En esta sección se describen los casos de prueba esenciales para evaluar la funcionalidad del sistema y los resultados que se esperan obtener al ejecutarlos.

Caso de prueba 1: Abrir una sesión.

Descripción del caso de prueba

Consiste en que el participante proporcione al sistema, su nombre de usuario y clave de acceso, el proyecto al que pertenece y la reunión en la cual participa.

Especificaciones de entrada

Nombre de usuario = Yuridia Hernández
Clave de acceso = yhernan
Proyecto = Guía
Reunión = Doce

Especificaciones de salida

El usuario debe poder entrar al sistema, este le muestra las opciones de trabajo e información que le corresponden según el rol que tenga asignado para esa revisión. Por ejemplo, si su rol es *moderador* el sistema mostrará la interface con las opciones disponibles para el participante como moderador.

Caso de prueba 2: Registro del proyecto y sus participantes.

Descripción del caso de prueba

Consiste en que el participante proporcione al sistema los datos del proyecto y sus participantes. Del proyecto proporciona nombre, descripción, fecha de inicio, fecha final e idioma y de los participantes proporciona el nombre y su función.

Especificaciones de entrada

El participante deberá proporcionar al sistema, los siguientes datos del proyecto:

Nombre = Evaluación de la guía de pruebas.
Descripción = Evaluación de la guía de valoración de sistemas de software durante el ciclo de vida.
Fecha de inicio = 01/01/02
Fecha Final = 01/09/02
Idioma = Español

Datos de los participantes:

Nombre	Función
José Cárdenas	Ing. de Pruebas
Brenda Flores	Diseñador
Raúl Romero	Analista
Jorge Ibarra	Ing. de Validación y Verificación

Especificaciones de salida

El usuario deberá poder registrar la información del proyecto. El sistema guardará los datos satisfactoriamente (sin mostrar mensajes de error), y al finalizar limpiará los campos de la interface de registro de proyectos.

Caso de prueba 3: Registro de una agenda.

Descripción del caso de prueba

Para realizar esta prueba, es necesario, que el usuario entre al sistema con el rol de moderador. Posteriormente, el participante deberá registrar la agenda proporcionando los datos de la agenda, los participantes y el calendario de actividades.

Especificaciones de entrada

El participante deberá proporcionar al sistema, los siguientes datos de la agenda:

Título = Reunión para revisión del plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida.
Objetivo = Aceptar y congelar el plan de pruebas.
Fecha = 20/02/02
Duración = 1 1/2 hrs
Tipo de Revisión = Caminata

Datos de los participantes:

Nombre	Rol
Jose Cárdenas	Presentador
Jorge Ibarra	Revisor

Brenda Flores	Revisor
Raúl Romero	Moderador

Datos del calendario de actividades:

Horario	**Actividad**	**Responsable**	**Tiempo Asignado**
8:00-8:10 am	Introducción	Moderador	10 min.
8:10-8:30 am	Presentación del Plan	Presentador	20 min.
8:30-9:00 am	Revisión del Plan	Revisores	30 min.

Comentarios = Todos los revisores deben asistir preparados a la reunión.

Especificaciones de salida

El usuario podrá realizar el registro de la información de la agenda. Por lo tanto, el sistema deberá guardar los datos en la base de datos sin mostrar mensajes de error, y al finalizar limpiará los campos de la interface de registro de agendas.

Caso de prueba 4: Registro de un documento.

Descripción del caso de prueba

El participante debe entrar al sistema con el rol de moderador, presentador, revisor o administrador. Enseguida, deberá registrar el documento que estará en revisión durante la agenda calendarizada en la sección anterior.

Especificaciones de entrada

El participante deberá proporcionar al sistema, los siguientes datos del documento:

Documento = Plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida.
Descripción = Contiene la descripción de las pruebas de aceptación que se aplicarán a la guía.
UrlDocumento = http://yhernand.cicese.mx/plan_pruebas.html

Para completar la actividad el usuario deberá registrar el autor del documento:

Autor (es)
Yuridia Hernández

Especificaciones de salida

El usuario podrá registrar la información del documento. El sistema deberá guardar los datos satisfactoriamente en la base de datos, y al finalizar limpiará los campos de la interface de registro de documentos. El documento estará relacionado con la agenda de la reunión que eligió el usuario al accesar al sistema.

Caso de prueba 5: Reporte de lista de defectos

Descripción del caso de prueba

El participante debe entrar al sistema con el rol de revisor. Enseguida, deberá registrar una lista de defectos para el documento de plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida.

Especificaciones de entrada

El participante deberá proporcionar al sistema, los siguientes datos de la lista de defectos:

Documento = Plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida.
Fecha = 17/07/02
Disposición = Documento condicionado
Tipo de Revisión = Inspección
Tiempo de Revisión = 30 min.

Datos de los defectos:

Descripción	**Ubicación**	**Tipo**	**Característica**	**Severidad**

El texto no es claro.	Pág. 5, párrafo 2	Documentación	Claridad	Menor
El formato del texto no cumple el estándar	Pág. 1-8	Estándares	Omisión	Menor

Comentarios = Corregir los errores encontrados.

Especificaciones de salida

El usuario podrá registrar la información de la lista de defectos en la interface. El sistema deberá guardar los datos en la base de datos sin marcar mensajes de error, y al finalizar limpiará los campos de la interface para registro de defectos.

Caso de prueba 6: Registro de guías de revisión

Descripción del caso de prueba

El participante debe entrar al sistema con el rol de moderador. Enseguida, deberá registrar una guía para la revisión del documento "Plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida".

Especificaciones de entrada

El participante deberá proporcionar al sistema, los siguientes datos de la guía de revisión:

Documento = Plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida.

Datos de los puntos a revisar:

Descripción
Se entiende en general el propósito del plan?
Es clara y precisa la redacción?
Cumple con los estándares definidos por la IEEE?

Especificaciones de salida

El usuario podrá registrar la información de la guía para la revisión del documento Plan de pruebas de la guía de valoración de sistemas de software durante el ciclo de vida. El sistema guardará los datos satisfactoriamente en la base de datos, y al finalizar limpiará los campos de la interface.

Caso de prueba 7: Presentación de un documento.

Descripción del caso de prueba

El participante entrará al área de presentación de documentos y solicitará el turno.

Especificaciones de entrada

Para solicitar el turno se deberá presionar un botón que representa la solicitud y que se encuentra en la barra de presentación. Posteriormente. el moderador le asignará el turno. Cuando el participante obtenga el turno, abrirá la ventana de documentos y seleccionará algún documento de la lista.

Especificaciones de salida

El participante podrá entrar al área de presentaciones, se cargará la barra de herramientas disponibles para apoyar la reunión. El documento seleccionado por el participante será mostrado en la interface para presentaciones de todos los participantes de la reunión

Caso de prueba 8: Control de Piso.

Descripción del caso de prueba

El participante con el rol de moderador asignará el turno a un participante de la lista en espera.

Especificaciones de entrada

Durante la reunión, el moderador seleccionará el participante en la lista de espera. Segundo, presionará el botón Activo.

Especificaciones de salida

La ventana mostrará la lista de los participantes en espera, quién tiene el turno y quién ya participó. La interface de control de turnos del moderador tendrá activado el botón Activo.

El moderador puede presionar el botón y asignar satisfactoriamente la palabra. Por último, el participante seleccionado aparece en la sección de *participante activo.*

Caso de prueba 9: Anotaciones sobre elementos de un documento.

Descripción del caso de prueba

El participante deberá cargar la interface de anotaciones gráficas para un párrafo del documento plan de pruebas de aceptación de la guía de valoración de sistemas de software durante el ciclo de vida.

Especificaciones de entrada

Una vez que el elemento haya sido cargado en la interface, el participante realizará trazos sobre el elemento.

Especificaciones de salida

El sistema obtiene los datos del trazo paloma, posteriormente, envía esos valores a cada interface de los participantes conectados a la reunión. El sistema dibuja una paloma en cada interface activa para anotaciones. Cada participante visualiza el trazo desplegado por el sistema.

E.3. Realizar reportes de las pruebas

El registro de los resultados de la aplicación de la prueba del sistema SCART es mostrado en esta sección.

E.3.1. Bitácora de prueba

Descripción

La bitácora describe la ejecución completa de las pruebas del sistema SCART durante el período que abarca del 15 de mayo al 25 de julio que tuvo lugar en el laboratorio de colaborativos del departamento de computación del CICESE. En esta bitácora se prueban los documentos de análisis y diseño del sistema SCART y su implementación.

Para la realización de la prueba se requirieron dos computadoras (el servidor y el cliente) que cumplen con los requerimientos establecidos en el plan de pruebas del sistema .

Actividades

La tabla VI muestra el inicio y la terminación de las actividades que corresponden al período de pruebas que abarcan del 15 de mayo al 25 de julio del 2001.

Tabla VI. Tabla de ejecución de actividades

Fecha	Hora	Actividad	Incidente
Mayo 15	8:00 A.M.	Inicia la prueba requerimientos	
Mayo 16	3:00 P.M.	Termina la prueba	
Mayo 17	8:00 A.M.	Inicia la prueba del diccionario de datos	
Mayo 21	5:50 P.M.	Termina la prueba	
Mayo 22	8:00 A.M.	Inicia la prueba del diagramas de objetos	
Mayo 23	5:45 P.M.	Termina la prueba.	
Mayo 24	8:00 A.M.	Inicia la prueba del modelo dinámico	
Mayo 28	3:00 P.M.	Termina la prueba.	
Mayo 29	8:00 A.M.	Inicia la prueba del modelo funcional	
Mayo 31	8:20 P.M.	Termina la prueba.	
Junio 4	8:00 A.M.	Inicia la prueba de la arquitectura	
Junio 4	1:00 P.M.	Termina la prueba.	
Junio 5	8:00 A.M.	Inicia la prueba de subsistemas	
Junio 6	1:00 P.M.	Termina la prueba.	
Junio 7	8:00 A.M.	Inicia la prueba del diseño de la base de datos	
Junio 11	6:30 P.M.	Termina la prueba.	
Junio 12	8:00 A.M.	Inicia la prueba del diseño de objetos	
Junio 18	3:00 P.M.	Termina la prueba.	
Junio 19	8:00 A.M.	Inicia la prueba del modelo dinámico	
Junio 22	8:00 P.M.	Termina la prueba.	
Junio 25	8:00 A.M.	Inicia la prueba de instalación	
Junio 25	11:30 A.M.	Se detiene la prueba	El manejador no es el adecuado
Julio 2	8:00 A.M.	Se actualiza el manejador y se compilan los archivos	
Julio 3	1:00 A.M.	Se termina la compilación de archivos	
Julio 4	8:00 A.M.	Se inicia la prueba de ayuda	
Julio 5	1:30 P.M.	Termina la prueba.	
Julio 6	8:00 A.M.	Inicia la prueba del interfaz	
Julio 9	1:00 P.M.	Termina la prueba.	
Julio 10	8:00 A.M.	Inicia la ejecución de casos de prueba	
Julio 12	12:30 A.M.	Se detiene la prueba	Errores en las tablas de la base de datos
Julio 16	8:00 A.M.	Se reanuda la prueba	
Julio 18	10:00 A.M.	Se suspende la prueba	El módulo de reunión de revisión no funciona
Julio 19	1:00 P.M.	Se termina la prueba	
Julio 20	8:00 A.M.	Se analizan resultados	
Julio 25	1:00 P.M.	Se evalua el sistema	

Descripción de la ejecución

La ejecución de la prueba se llevó a cabo en el periodo planeado. Se presentaron tres incidentes que retrasaron la ejecución de la prueba. El primer incidente se presentó el día 25 de junio durante las pruebas de instalación, el problema fue que el sistema no reconoció el manejador de acceso a la base de datos de ORACLE. El segundo incidente se presentó el 12 de julio en donde se presentaron algunos problemas con la definición de las tablas de la base de datos. El tercer incidente se presentó el día 18 de julio, el sistema se bloqueó al entrar a la reunión de revisión. Con este incidente se dió por terminada la ejecución de la prueba . Se analizaron resultados y se hizo un diagnóstico.

Resultados del procedimiento

El procedimiento utilizado permitió que las pruebas se realizaran adecuadamente y no se presentaran retrasos largos en el calendario. Se realizaron el 80 % de los casos de prueba diseñados.

E.3.2. Reportes de incidentes de pruebas

Reporte 1: Manejador inadecuado

Resumen

Se presentó un problema en la instalación del sistema, después de realizar los pasos listados en la guía de instalación se negó el acceso al sistema. El incidente se reporta en la bitácora de prueba.

Descripción del incidente

El 25 de junio a las 11:30 A.M al terminar la instalación del sistema SCART se presentó un problema al intentar accesar al sistema, el manejador de la base de datos de ORACLE era diferente al utilizado originalmente por el sistema. La instalación se realizó siguiendo los pasos del archivo de instalación que es parte del código del sistema.

Impacto

Se suspendió la prueba, durante este tiempo se bajaron varios manejadores de internet para encontrar el adecuado. Cuando se tuvo al alcance se hicieron varias pruebas con el nuevo driver. El retraso fué de una semana.

Reporte 2: Definición de tablas en la base de datos

Resumen

Se presentaron problemas durante la ejecución de casos de prueba. Durante la prueba se presentaron varias inconsistencias entre el sistema y la definición de las tablas en la base de datos que impedían que se realizaran las transacciones. El incidente se reporta en la bitácora de prueba.

Descripción del incidente

El 12 de julio a las 12:30 A.M se presentaron varios problemas al hacer las transacciones con la base de datos como omisión de columnas y atributos en las tablas, inconsistencia del tamaño de las columnas.

Impacto

Se detuvo la prueba, durante este tiempo se revisaron los documentos de análisis y diseño, el código del sistema y las tablas de la base de datos. Se hicieron las modificaciones respectivas siguiendo los documentos de análisis y diseño. El retraso fué 3 días.

Reporte 3: Falla del módulo reunión de revisión

Resumen

Se presentó un problema durante la ejecución de casos de prueba. El sistema no permita la entrada a la reunión de revisión. El incidente se reporta en la bitácora de prueba.

Descripción de incidentes

El 18 de julio a las 10:00 el sistema se bloqueó al intentar accesar la reunión de revisión, el proceso se repitió varias veces en circunstancias diferentes y no se obtuvo ninguna respuesta.

Impacto

Se suspendió la prueba y posteriormente se dió por terminada la prueba.

E.3.3. Reporte de transmisión

Elementos

P1. Análisis del sistema. Se probó el análisis de requerimientos y los tres modelos de su especificación: el modelo de objetos (diccionario de datos, relaciones entre clases y diagramas de objetos), el modelo dinámico y el modelo funcional.

P2. Diseño del sistema. Se probó la arquitectura, los subsistemas, el almacenamiento y el manejo de los datos y la metodología de modelado.

P.3. Sistema. Se probó la instalación, ayuda, interfaz y funcionalidad del sistema.

Localización

El resultado de las pruebas se presenta en este trabajo que lleva por nombre Pruebas: Una guía de valoración de sistemas de software durante el ciclo de vida.

Estado

El estado es terminado.

E.3.4. Resumen de reportes de pruebas

Resumen

Este reporte muestra los resultados de la ejecución de las pruebas al sistema SCART, se prueban los documentos de análisis y diseño y el sistema.

Varianzas

En el análisis se observó que los requerimientos y su especificación estuvieron descritos en el detalle suficiente y no se encontraron indicadores que dieran lugar a desvíos en la etapa de diseño. Por otro lado, fue notoria la ausencia de los requerimientos no funcionales.

En el diseño se encontraron ambigüedades en los subsistemas que pudieron haber desviado y dificultado el modelado y la traducción a un lenguaje de codificación.

Las pruebas al sistema no se realizaron en su totalidad. El sistema falló varias veces al entrar al módulo de reunión de revisión.

Evaluación de la completitud

El análisis y el diseño se probaron totalmente y de acuerdo a su nivel de granularidad. El sistema se probó parcialmente, esto es, se ejecutaron satisfactoriamente seis casos de prueba y tres de ellos fueron suspendidos.

Resumen de resultados

Se probaron los elementos del análisis del sistema: descripción de requerimientos, diccionario de datos, diagramas de objetos, modelo dinámico y funcional.

Se probaron los elementos del diseño del sistema: arquitectura, subsistemas almacenamiento y el manejo de datos y la metodología de modelado.

Se probó la instalación, ayuda, interfaz y funcionalidad del sistema SCART..

Evaluación

Descripción de requerimientos. La mayoría de los requerimientos estuvieron bien definidos, aunque faltaron definir los requerimientos de desempeño, tiempo real, precisión y de exactitud.

Diccionario de datos. No se encontraron errores en las definiciones de las clases.

Diagramas de objetos. Existen algunas inconsistencias en el diagrama general de objetos.

Modelo dinámico. En el modelo dinámico se omitieron varios diagramas de eventos, tiene algunas inconsistencias y ambigüedades.

Modelo funcional. En el modelo funcional se omitieron varios diagramas.

Arquitectura. La arquitectura que se utilizó fue cliente/ servidor y es la apropiada para un sistema de esta naturaleza. El cliente solicita servicios y el servidor atiende peticiones.

Además el lenguaje java utilizado para su implementación también es el adecuado ya que permite que se satisfagan los requerimientos de comunicación vía internet e independencia de la plataforma

Subsistemas. Existe ambigüedad en el subsistema de servicios de colaboración particularmente en el módulo administración de reunión.

Estructura de la base de datos. La omisión de las llaves primarias y foráneas en las tablas que forman la estructura de la base de datos dificultó su comprensibilidad, además de que la relación entre las tablas es ambigua.

Diseño de Objetos. No se definieron los atributos y métodos de las clases del modelo de objetos del diseño.

Modelo dinámico. En el modelo dinámico se omitieron diagramas de eventos: registro del administrador, registro de acuerdos, registro de la minuta, registro de resumen de defectos.

Interfaz. Se encontraron errores variados en la presentación de las pantallas como definición de botones default, desorden en las listas de desplegado, falta de letras subrayadas en los botones, entre otros.

Instalación. El procedimiento de instalación es manual. Se realizaron los pasos indicados en el archivo de instalación y el sistema no se instaló satisfactoriamente .

Seguridad. La funcionalidad fue tal como se indicaba en el diseño de las pruebas.

Ayuda. La ayuda es pobre, lejos de ayudar al usuario lo confunde por la cantidad de inconsistencias que tiene. El contenido muestra que actividades deben hacerse más no cómo deben hacerse que, realmente es lo que le interesa al usuario.

Resumen de actividades

Las actividades se realizaron en el orden dispuesto en el calendario de actividades y fueron las siguientes: pruebas al análisis, diseño e interfaz, análisis del resultados y evaluación del sistema.

E.4. Seguimiento de las pruebas

El seguimiento de las pruebas es la administración de las pruebas mediante una supervisión adecuada basada en reportes. A continuación se muestran los reportes que se obtuvieron como resultado de llevar a cabo este paso.

Elementos probados a la fecha contra estimados

La tabla VII muestra el avance que tuvo la prueba durante el calendario de actividades comparando los elementos probados contra los estimados. Se observa que las pruebas del análisis y diseño del sistema SCART se realizaron de acuerdo a lo establecido en el plan pruebas. Sin embargo, se presentaron problemas al final del período durante las pruebas del sistema en donde se estimaba probar 8 elementos y sólo se probaron 5 de éstos. La ejecución se realizó en el orden previsto, es decir no hubo saltos que ocasionaran que se dejaran de probar alguno de los elementos.

Tabla VII. Tabla de elementos probados a la fecha vs estimados.

Fecha	**Elementos**	**Total de elementos**	**Probados sin errores**	**Probados con errores**	**Sin probar**	**Estimados**
14/ mayo	Requerimien - tos	1	1	0	0	1
17/ mayo	Modelo de objetos	3	1	2	0	3
24/ mayo	Modelo dinámico	10	5	5	0	10

29/ mayo	Modelo Funcional	5	5	0	0	5
4/ junio	Arquitectura	1	1	0	0	1
5/ junio	Subsistemas	4	0	4	0	4
7/ junio	Estructura de la base de datos	12	0	12	0	12
12/ junio	Diseño de Objetos	1	1	0	0	1
19/ junio	Modelo dinámico	8	5	3	0	8
25/ junio	Instalación	1	0	1	0	1
4/ julio	Ayuda	1	0	1	0	1
6/ julio	Interfaz	7	0	7	0	7
10/ julio	Funcional	8	5	0	3	8

Gráfica del estado de los elementos

La gráfica del estado de los elementos muestra el estado de la prueba, se divide en elementos libres de errores, con errores y sin probar. La figura 4 indica el estado de los elementos del análisis, diseño y sistema que se probaron. La cual muestra que se probaron todos los elementos del análisis y diseño y que faltaron elementos que probar en el sistema.

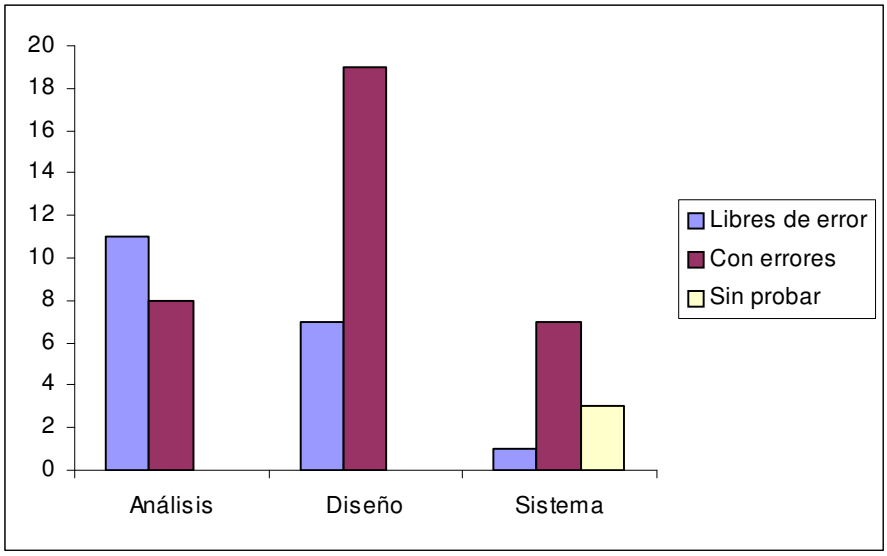


Figura 4. Gráfica del estado de los elementos

La tabla VIII muestra la información de los 56 elementos que se probaron: 19 del análisis, 26 del diseño y 11 del sistema. De los cuales 19 (35%) estuvieron libres de error, 34 (60%) tuvieron errores y 3 no se probaron (5%). Este resultado indica que la prueba fue exitosa, ya que se probaron 53 de los 56 estimados.

Tabla VIII. Tabla de datos del estado de los elementos.

Elementos	Libres de errores	Con errores	Sin probar	Total elementos	Estimados
Análisis del sistema	11	8	0	19	19
Diseño del sistema	7	19	0	26	26
Sistema	1	7	3	11	11
TOTAL	19	34	3	56	56

Gráfica de distribución de defectos en los elementos

La Figura 5 muestra la distribución de los defectos en los elementos del análisis del sistema, entre los cuales se consideran el análisis de requerimientos, modelo de objetos, modelo dinámico y modelo funcional. La cantidad de errores encontrada en el análisis del sistema se considera baja, el elemento con mayor cantidad de errores es el modelo dinámico.

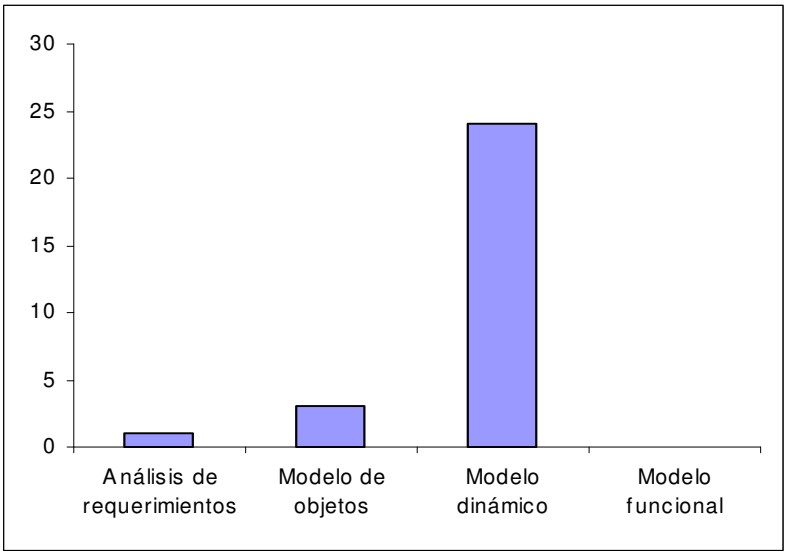


Figura 5. Gráfica de distribución de defectos en el análisis del sistema

La tabla IX muestra la información desglosada de los defectos encontrados en el análisis del sistema

Tabla IX. Tabla de defectos del análisis del sistema.

Análisis del sistema	Defectos
Análisis de requerimientos	1
Modelo de Objetos (3)	
Diccionario de datos	0
Relaciones entre clases	2
Diagrama de objetos	1
Modelo dinámico (10)	
Registro de proyectos	1
Registro de participantes	0
Registro de agendas	2
Registro de documentos	0
Registro de defectos	0
Entrar a sesión	0
Diagrama de estados caminata	0
Diagrama de estados inspección	4
Diagrama de estados documento	3
Otros errores	14
Modelo funcional (5)	
Registro de proyectos	0
Registro de participantes	0
Registro de la agenda	0
Registro de documentos	0
Anotaciones gráficas	0
TOTAL	28 errores

La figura 6 muestra la distribución de los defectos en los elementos del diseño del sistema, entre los cuales se consideran la arquitectura, subsistemas, estructura de la base de datos y modelo dinámico. La cantidad de errores encontrados en el diseño del sistema es baja. Los elementos que presentaron la mayor cantidad de errores son la estructura de la base de datos y subsistemas. Al parecer, el modelo dinámico se probó parcialmente ya que no tuvo

la proporción de defectos esperada, esto es, se detectaron menos errores que en el modelo dinámico del análisis este es un indicador de que quedaron errores sin detectar.

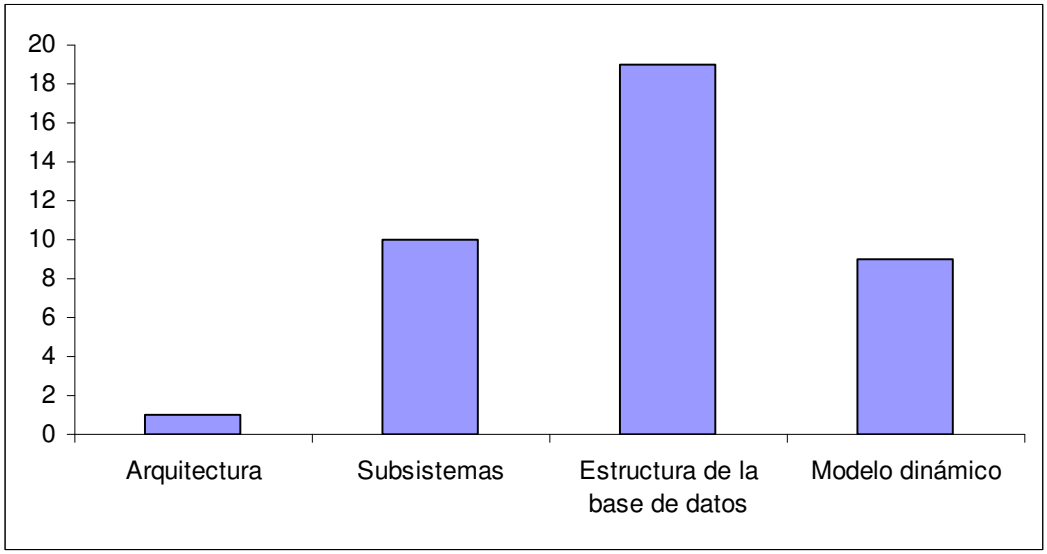


Figura 6. Gráfica de distribución de defectos en el diseño del sistema

La tabla X muestra la información desglosada de los defectos encontrados en el diseño del sistema.

Tabla X. Tabla de defectos del diseño del sistema.

Diseño del sistema	Defectos
Arquitectura	1
Subsistemas (3)	
Interfaces	3
Asíncrono	2
Síncrono	2
Colaboración	3
Estructura de la base de datos (12)	
Actividad_Agenda_Scart	1
Agenda_Scart	1
Autor_Docto_Scart	1
Checklist_Scart	1
Datos_Defectos_Scart	2
Documento_Scart	2
Lista_Defecto_Scart	3
Participante_Scart	1
Participante_Agenda_Scart	1
Participa_Proy_Scart	1
Proyectos_Scart	2
Reporte_Revisión_Scart	3
Diseño de objetos	0
Modelo dinámico (8)	
Abrir una sesión	1
Registro del administrador	0
Registro de proyectos	0
Registro de una agenda	0
Registro de un documento	3
Presentación de un documento	0
Anotaciones sobre elementos	0
Otros	5
TOTAL	38 errores

La figura 7 muestra la distribución de los defectos en los elementos del sistema, se probó la instalación, ayuda e interfaz. La interfaz tuvo la mayor cantidad de errores, los cuales se distribuyen de forma uniforme en sus elementos.

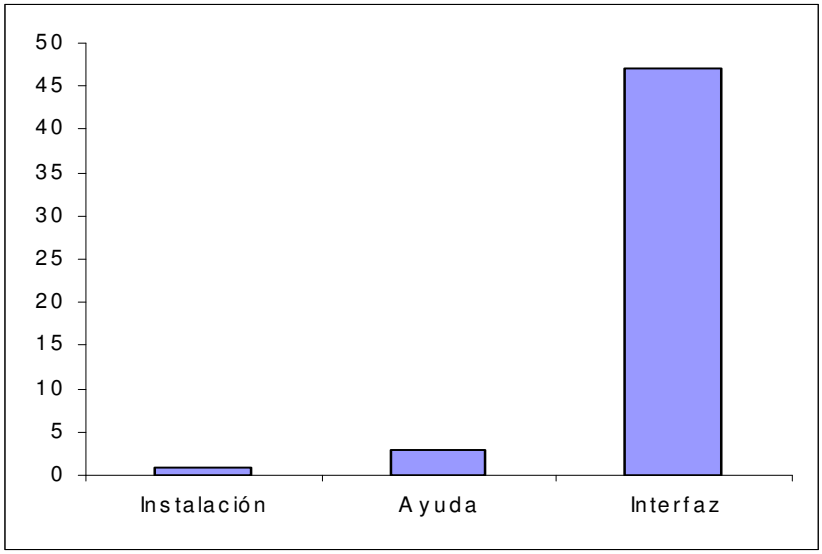


Figura 7. Gráfica de distribución de defectos en el sistema

La tabla XI muestra la información desglosada de los defectos encontrados en el sistema.

Tabla XI. Tabla de defectos del sistema.

Sistema	Defectos
Elementos	
Instalación	1
Seguridad	0
Ayuda	3
Interfaz	
Registro del admionistrador	5
Registro del proyecto	10
Registro de documentos	5
Registro de agenda	9
Registro de guías	5
Registro de reporte	5
Registro de lista de defectos	8
TOTAL	51 errores

Gráfica de defectos corregidos por tipo

La figura 8 muestra la distribución de defectos críticos que se corrigieron durante la ejecución de la prueba. La cual nos indica que el diseño fue la fase en donde se presentaron la mayor cantidad de defectos que se corrigieron en su mayoría, estos estaban relacionados directamente con la estructura de la base de datos como nombre y/o tamaño de las columnas de las tablas de la base de datos. El propósito de corregirlos fue continuar con la ejecución de las pruebas. Por otra parte, en la implementación se presentaron algunos errores también relacionados con la base de datos que fueron corregidos en el momento.

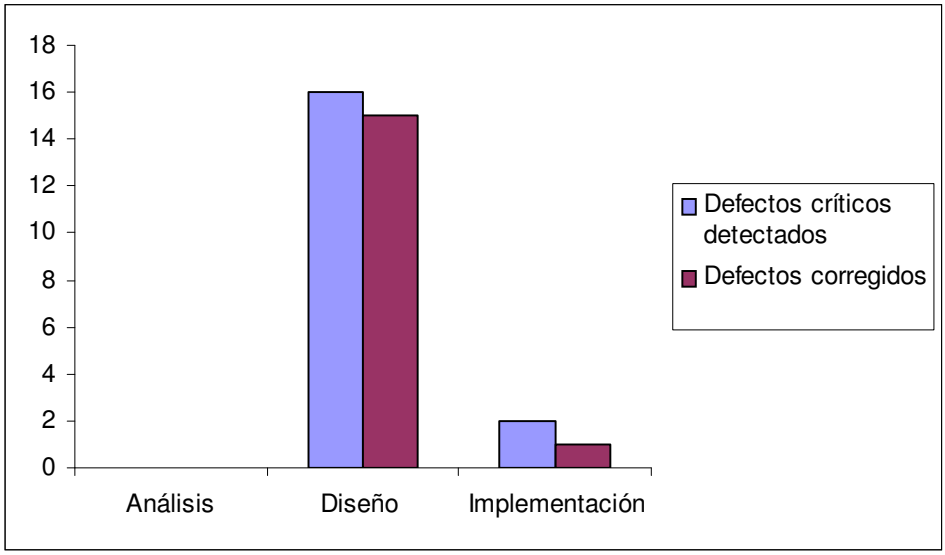


Figura 8. Gráfica de distribución de defectos críticos

La siguiente gráfica muestra la distribución de defectos mayores que se corrigieron durante la ejecución de la prueba. La figura 9 muestra que en el análisis se concentran la mayoría de los defectos mayores y sólo unos pocos en el diseño. Por otro lado, se aprecia que ninguno de los errores fue corregido esto se debe a que no se considera conveniente realizar dichos cambios ya que no afectaban el avance de la ejecución de las pruebas.

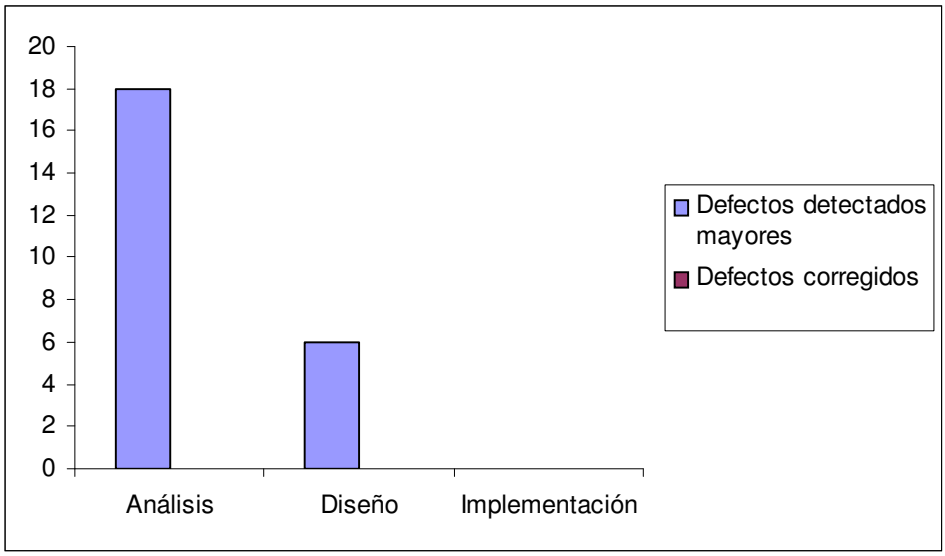


Figura 9. Gráfica de distribución de defectos corregidos mayores

La siguiente gráfica muestra la distribución de defectos menores que se corrigieron durante la ejecución . La figura 10 indica que en la implementación se encontraron la mayoría de los defectos. Además, muestra que no se corrigieron los errores por los motivos mencionados anteriormente.

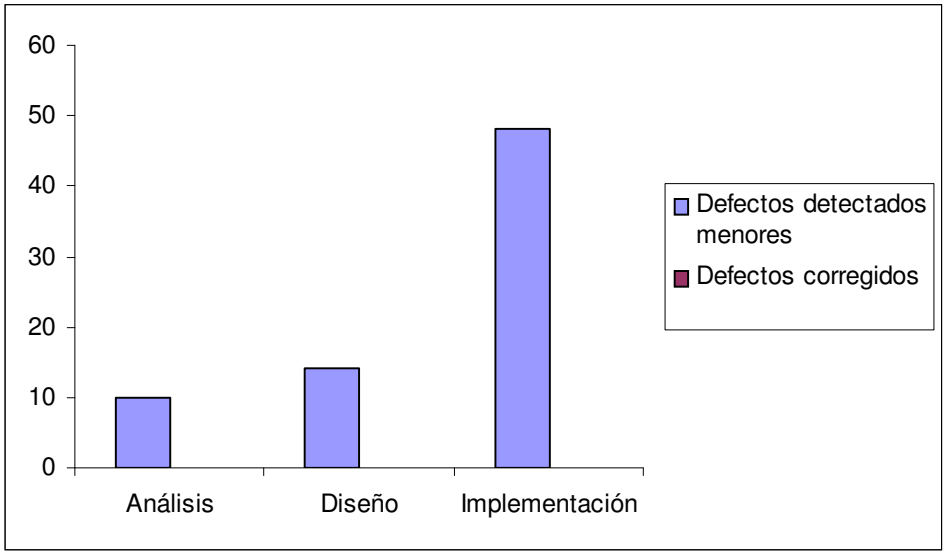


Figura 10. Gráfica de distribución de defectos corregidos menores

La tabla XII muestra la información desglosada de los defectos corregidos por tipo en tres categorías críticos, mayores y menores.

Tabla XII. Tabla de distribución de defectos corregidos por tipo.

Defectos Críticos	**Defectos Críticos Detectados/ Corregidos**	**Defectos mayores Detectados/ Corregidos**	**Defectos menores Detectados/ Corregidos**
Análisis del sistema			
Análisis de requerimientos	0/0	0/0	1/0
Modelo de objetos	0/0	1/0	2/0
Modelo dinámico	0/0	11/0	6/0
Modelo funcional	0/0	6/0	1/0
TOTAL	0/0	18/0	10/0
Diseño del sistema			
Arquitectura	1/0	0/0	0/0
Subsistemas	0/0	3/0	6/0
Estructura de la base de datos	15/15	0/0	3/0
Diseño de objetos	0/0	0/0	0/0
Modelo dinámico	0/0	3/0	5/0
TOTAL	16/15	6/0	14/0
Implementación			
Instalación	1/1	0/0	0/0
Seguridad	0/0	0/0	0/0
Ayuda	1/0	0/0	0/0
Interfaz	0/0	0/0	48/0
	2/1	0/0	48/0

Reporte de pruebas

En el análisis del sistema se probaron 19 elementos. De los cuales 11 están libres de error y 8 tienen defectos. De los defectos detectados 18 son defectos mayores y 10 son defectos menores.

En el diseño del sistema se probaron 26 elementos. De los cuales 7 están libres de error y 19 tienen defectos. De los defectos detectados 16 son críticos, 6 son mayores y 14 son menores.

En el sistema se consideraron 11 elementos. De los cuales un elemento está libre de error, 7 tienen defectos y 3 no se probaron. De los defectos detectados 2 son críticos y 48 son menores.

La mayoría de los defectos no se arreglaron, únicamente aquellos que se requerían para continuar el proceso.

Quedó pendiente de probar los elementos del módulo asíncrono de SCART que corresponden a la presentación de la revisión, control de piso y anotaciones sobre el documento.

E.5. Métricas

Distribución de defectos por clase

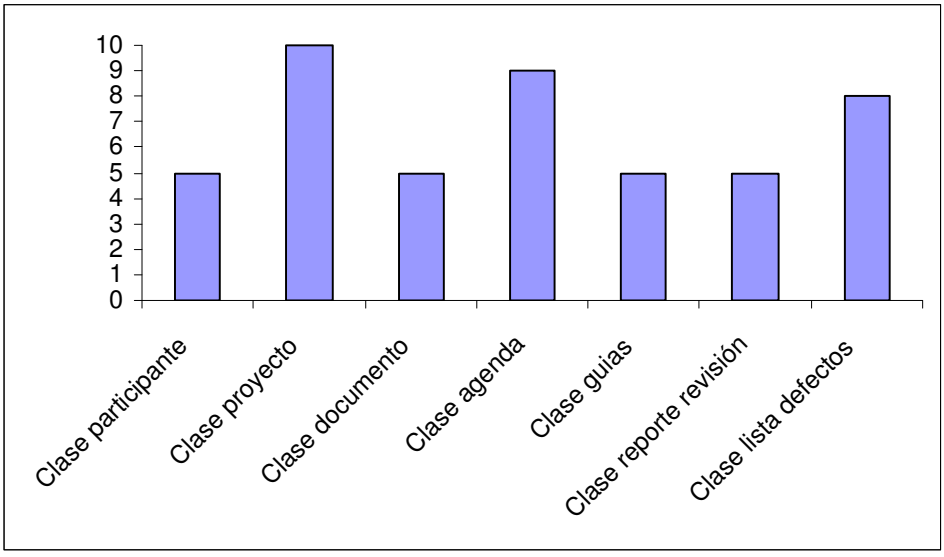


Figura 11. Distribución de defectos por clase

La gráfica de distribución de defectos por clase muestra la cantidad de errores detectados por clase. En la figura 11 se visualizan los defectos en las clases que se probaron en el

sistema SCART. Se observa que las clases con mayor proporción de defectos son proyecto, agenda y lista de defectos.

Distribución de defectos por fase en el ciclo de vida

La gráfica de distribución de defectos por fase en el ciclo de vida muestra los errores que se detectaron en cada etapa. En la figura 12 se observa que los errores fueron incrementándose conforme se avanzaba en la ejecución de las pruebas. Idealmente los errores deben ir disminuyendo en cada etapa cuando se realiza la prueba.

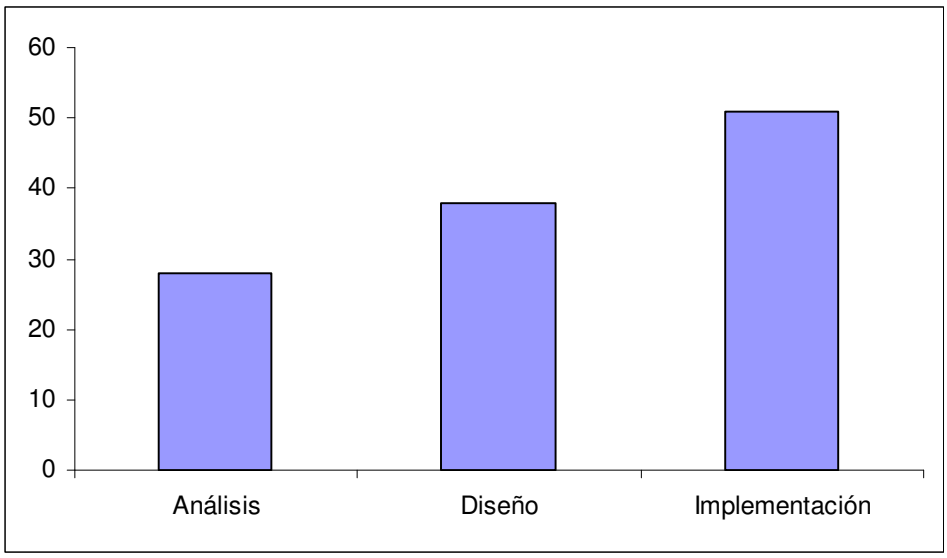


Figura 12. Distribución de defectos por fase en el ciclo de vida

Defectos descubiertos debido a las pruebas

Los defectos descubiertos debido a las pruebas se calculan dividiendo el número de defectos detectados con las pruebas entre los defectos del sistema total.

$$D_{DP} = \frac{D_P}{D_T}$$

Donde: D_{DP} = defectos descubiertos debido a las pruebas,, D_P = número de defectos detectados con las pruebas y D_T = defectos del sistema total.

D_{DP} =117/ 122= 0.95

Esta formula proporciona el porcentaje de defectos obtenidos por la aplicación de las pruebas al sistema SCART respecto a los defectos totales. Las pruebas muestran el 95% de efectividad.

Eficiencia de la selección de las pruebas

La eficacia de la selección de las pruebas se calcula dividiendo el número de pruebas entre los defectos del sistema total.

$$E_P = \frac{N_P}{D_T}$$

Donde: E_P = Eficacia de la selección de pruebas, N_P = número de pruebas y D_T = defectos del sistema total

E_P = 13/122= 0.11

La fórmula muestra la habilidad de seleccionar y aplicar el plan de pruebas. En este caso podemos ver cada defecto de una prueba representa el 11% de los defectos que se detectaron.

Defectos encontrados en relación al código

Los defectos encontrados se calculan dividiendo el número de los defectos encontrados entre el tamaño del sistema

$$D_E = \frac{N_D}{T_S}$$

Donde: D_E *=Defectos encontrados,* N_D *=Número de defectos encontrados y* T_S *= el tamaño de sistema.*

$D_E = 117/6000 = 0.0195$

La fórmula muestra la proporción del código promedio necesario para poder encontrar un defecto. Cuando el valor está muy cerca de 0 es un indicador de la existencia de errores ocultos que deben detectarse. En base a esto se obtuvo que aproximadamente el 2% del código tuvo defectos. Es decir, tenemos un error por cada 60 líneas de código.

Cuestionario de evaluación

Cuestionario para la evaluación de la propuesta de pruebas de la guía de valoración de sistemas durante el ciclo de vida

Nombre del proyecto _____Versión _____

Persona que lo aplica _____Puesto que ocupa_____

El rango de evaluación para las siguientes preguntas es de **1**. Fácil – **5** Difícil

1. Establecimiento del plan de pruebas
    a) Crear el plan de pruebas fue:

        1 2 3 4 5

2. Aplicación de la prueba
    a) Crear diseños de prueba fue:

        1 2 3 4 5

    b) Crear casos de prueba fue:

        1 2 3 4 5

    c) Crear procedimientos de prueba fue:

        1 2 3 4 5

3. Reportes de pruebas
    a) Crear bitácoras de pruebas fue:

        1 2 3 4 5

    b) Crear reportes de incidentes fue:

        1 2 3 4 5

c) Crear reportes de transmisión fue:

1 2 3 4 5

d) Crear resumen de reportes de prueba fue:

1 2 3 4 5

4. Seguimiento de la prueba

a) Crear la gráfica de elementos probados a la fecha vs estimados fue:

1 2 3 4 5

b) Crear gráfica de estado de elementos fue:

1 2 3 4 5

c) Crear gráfica de distribución de defectos en los elementos fue:

1 2 3 4 5

d) Crear gráfica de defectos corregidos por tipo fue:

1 2 3 4 5

e) Crear reporte de pruebas fue:

1 2 3 4 5

5. Aplicación de métricas

a) Crear la gráfica de distribución de defectos por módulo fue:

1 2 3 4 5

b) Crear la gráfica relativa de distribución de defectos fue:

1 2 3 4 5

c) Crear la gráfica de defectos por el ambiente:

1 2 3 4 5

d) Crear la gráfica de defectos por tipo:

1 2 3 4 5

e) Crear la gráfica de distribución de defectos por quién lo encontró fue:

1 2 3 4 5

f) Crear la gráfica de distribución de defectos por fase en el ciclo de vida fue:

1 2 3 4 5

g) Crear la gráfica de causas de defectos fue:

1 2 3 4 5

h) Aplicar e interpretar la fórmula de caminos ejecutados fue:

1 2 3 4 5

i) Aplicar e interpretar la formula de instrucciones ejercitadas fue:

1 2 3 4 5

j) Aplicar e interpretar la fórmula de defectos descubiertos debido a las pruebas fue:

1 2 3 4 5

k) Aplicar e interpretar la fórmula de la eficacia de selección de pruebas fue:

1 2 3 4 5

l) Aplicar e interpretar la fórmula de defectos encontrados en relación al código fue:

1 2 3 4 5

m) Aplicar e interpretar la fórmula de tiempo para corregir defectos

1 2 3 4 5

n) Aplicar e interpretar la fórmula de costo de la etapa de pruebas fue:

1 2 3 4 5

o) Aplicar e interpretar la fórmula de presupuesto alcanzado fue:

1 2 3 4 5

p) Aplicar e interpretar la fórmula de prueba de automatización fue:

1 2 3 4 5

Preguntas Generales

1. ¿Fué fácil la aplicación dela guía de valoración de sistemas de software durante el ciclo de vida?
2. ¿Consideras adecuado el diseño de cada etapa de la guía?
3. ¿Cuál etapa fue la más complicada de desarrollar?
4. ¿Crees que el modelo está completo?
5. ¿Qué le hace falta?
6. ¿Qué problemas tuviste al realizarlo?
7. ¿Enriqueció tu aprendizaje sobre el desarrollo de las pruebas?
8. ¿Crees que se debería dar una explicación más amplia en alguna de las etapas?
9. ¿Recomiendas que otros usuarios lo utilizen?
10. ¿Qué sugieres para mejorar el sistema?