Universidad de las Ciencias Informáticas

Facultad # 3

Sistema Integrado de Gestión Estadística. Rol de ingeniero de prueba, módulo Generador de Modelos.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Arianna Valdés Martínez

Tutores: Misael Salvador González Rodríguez.

Ridosbey Milán Iglesias.

La Habana, Junio 2007.

DECLARACIÓN DE AUTORÍA

Yo: **Arianna Valdés Martínez** declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas(UCI) a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma del Autor Firma del Tutor

"Semejante a la muerte y a los impuestos, la prueba es desagradable e inevitable."

ED Yourdon.

Agradecimientos.

En primer lugar agradezco a nuestro Comandante en Jefe Fidel Castro Ruz, la oportunidad de haber creado esta universidad de las Ciencias Informáticas, que nos ha proporcionado a miles de jóvenes la posibilidad de realizar esta carrera del futuro.

Agradezco también a mis tutores y profesores, Eugenia Genoveva Muñiz Lodos, Ridosbey Milán Iglesias, Misael Salvador González Rodríguez, que con su ayuda incondicional y esmerada contribuyeron a que este trabajo saliera adelante a pesar de todos los problemas que se presentaron en el camino siempre me dieron ánimos para seguir esforzándome.

Agradezco a todos mis compañeros de proyecto, en especial a Julio Ernesto Ortiz, Armando Lobo y todos los demás, por su cooperación en mi trabajo.

Además mis agradecimientos especiales a mis seres queridos, en particular a mi querida madre Graciela Martínez Urquiza, mi novio Michel Molina Ríos ,mi suegra Ana Del Rosario Ríos Vázquez y mis amigos que no por ser los últimos menos importantes, Mairelys Plaza Matos , Elaine y Javier .

Dedicatoria

Dedico este trabajo de diploma a mi familia y en especial a mi madre y mi hermano, que siempre han confiado en mí y me dieron las fuerzas para seguir adelante y a todas las personas que de una forma u otra siempre me ayudaron y apoyaron en los momentos más difíciles de mi carrera y en especial en estos últimos meses y que sin ellos no hubiese podido llegar al final.

RESUMEN

El desarrollo de sistemas de software implica una serie de actividades de la producción y las posibilidades de que aparezca un fallo humano son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos …pueden estar especificados de forma errónea o imperfecta, así como en posteriores pasos de diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad. Las pruebas de software son un elemento crítico para la garantía de la calidad. Citando a Pressman. El presente trabajo tiene entre sus principales objetivos desarrollar las pruebas al módulo Generador de Modelos que se encuentra formando parte del software Sistema Integrado de Gestión Estadística, como línea de producción de la Facultad 3, perteneciente a la Universidad de las Ciencias Informáticas. Se reorganizaron los casos de uso de prueba de caja negra, utilizando las técnicas de partición equivalente, de análisis de valores limites, pruebas a la documentación, además de la utilización de estrategias de validación funcional a dicho módulo. Las conclusiones generales se mueven en las teorías y prácticas que definen la efectividad y evaluación del software. En tal sentido, nuestros resultados permiten aseverar la aplicación de pruebas de caja negra, mostrándose como resultado algunos errores que no habían sido detectado por los desarrolladores y que era necesario eliminar debido a que fueron requisitos funcionales declarados por los clientes.

TABLA DE CONTENIDOS.

INTRODUCCIÓN.

En las últimas décadas, el software ha pasado de ser una resolución de problemas especializada y una herramienta de análisis de información, a ser una industria por si misma, creciendo de forma acelerada y con ello la satisfacción a los clientes en menor tiempo y costo posible.

En nuestro país del mismo modo, se han establecido líneas de acción de interés para el Estado Cubano, que repercutirán en la economía, política y la sociedad, buscándose beneficios que traigan consigo no solo un desarrollo de productos de software, sino también su entrada al mercado a escala mundial aprovechando su perspectiva económica. En tal sentido nuestro comandante en jefe Fidel Castro Ruz expresó… “La idea fundamental es que se convierta en la rama más productiva, aportadora de recurso para la nación…”

Asumiendo estos planteamientos y motivados por la necesidad de nuestro trabajo de diploma, hemos experimentado en las teoría y prácticas del *software,* en la que se destacan algunas de sus principales características: confiables, precisos, rápidos de utilizar, flexibles, etc. Según Pressman, R 2005, un producto cumple con estos requisitos cuando se le han aplicado las técnicas de Ingeniería de *Software* adecuadas, se han utilizados los roles apropiados para el desarrollo de las tareas en la empresa de *software* y en su etapa de comprobación se le han aplicado las pruebas necesarias para lograr el nivel de calidad requerido**.** Atendiendo a estas exigencias**,** es que se aplican las pruebas requeridas a los productos en su etapa de desarrollo y entonces conseguir que salgan al mercado con un mínimo de errores y tengan una buena aceptación por parte de los clientes.

Con la introducción de las microcomputadoras en el antiguo Comité Estatal de Estadísticas (CEE), en noviembre de 1984, se comenzó el estudio, análisis y diseño de un sistema que permitiera procesar modelos de información estadística, sin que fuera necesario realizar una programación específica para cada uno y que a la vez brindara las mayores posibilidades, tanto en la validación, como en las ediciones, de acuerdo con las características particulares de cada modelo. Así surge la primera versión del MICROSET, la cual en septiembre de 1987 es sustituida por el MICROSET II, resultado de un nuevo diseño del sistema y que obtiene amplia difusión debido a su eficiencia y sencillez, teniendo en cuenta la introducción, en 1986, de las microcomputadoras en toda la red de Centros de Cálculo del Comité Estatal de Estadísticas.

Situación problemática.

Al analizar el contexto actual de la Oficina Nacional de Estadísticas, se pudo observar como las mismas trabajan fundamentalmente con el software llamado MICROSET NT, a este producto lo antecedieron las versiones I, II, III de MICROSET, programas que se originan, dada la necesidad primordial de lograr un sistema general integral para el procesamiento de datos y las ediciones de tablas, a través de las que se proporcionen y optimicen los procesamientos de la información en estas oficinas. Sus especialistas en convenio con la Universidad de las Ciencias Informáticas, expusieron sus limitaciones existentes, entre las que tenemos: no posee una interfaz de fácil interacción con los usuarios de éste sistema, el mismo está elaborado para trabajar sobre el sistema Windows 98, siendo incompatible con los sistemas operativos de la familia Windows NT, además de que no posibilita gestionar las nomenclaturas de los modelos estadísticos, tampoco admite la gestión de los modelos y la gestión de los cuadres es compleja. Por todas estas desventajas presentadas, se específica la sustitución del software MICROSET NT, para agregar nuevas funcionalidades importantes a los clientes. A partir de este momento se crea el proyecto Sistema Integrado de Gestión Estadística, el cual está integrado por cuatro módulos fundamentales, de estos el Generador de Modelos, es el que forma parte del objetivo de nuestro trabajo. Para verificar el cumplimiento de los requisitos funcionales, señalados por los clientes de la Oficina Nacional de Estadísticas, es necesario realizar pruebas de caja negra a dicho módulo, antes de que el producto sea entregado a los usuarios. Por lo que surge la necesidad de contar con un ingeniero de prueba que sea el responsable de la realización de las mismas.

Estos argumentos nos llevan a plantear el siguiente **Problema científico:**

- No se cuenta con un proceso de aplicación de pruebas de caja negra , al módulo Generador de Modelos, perteneciente al Sistema Integrado de Gestión Estadísticas, impidiendo la verificación de los requisitos funcionales planteados por los clientes.

La problemática manifestada permite establecer el **Objeto de estudio,** de la presente investigación, la cual se dirige al proceso de aplicación de pruebas de caja negra, en software de gestión.

Antes el problema científico y el objeto de estudio que orientan la presente investigación, se abren las puertas al **Campo de acción,** el cual queda enmarcado en la aplicación de pruebas de caja negra, al módulo Generador de Modelos.

Esta afirmación aunque es bastante abarcadora, no debe interpretarse como algo rígido o categórico, pero sí permite suponer que se puede aplicar a casos particulares donde se refleje con mayor fuerza la necesidad de una solución científica, por lo que hemos determinado a manera de **Hipótesis** la siguiente afirmación:

- Si realizamos un adecuado proceso de aplicación de pruebas de caja negra al módulo Generador de Modelos del Sistema Integrado de Gestión Estadísticas, se obtendrá un producto que cumpla con los requisitos definidos previamente por el cliente.

Una vez planteada la hipótesis que sitúa esta investigación, nos proponemos darle solución a los mismos, a través del siguiente **Objetivo general:**

- Aplicar las pruebas de caja negra, al módulo Generador de Modelos perteneciente al software Sistema Integrado de Gestión Estadísticas, en su primera iteración.

A partir de lo antes señalado exponemos los siguientes **Objetivos específicos** de la investigación.

- Realizar la selección y aplicación de estrategias y técnicas de prueba de caja negra al módulo Generador de Modelos.
- Hacer un plan de pruebas que detalle todo el proceso de realización de las pruebas.
- Definir los procedimientos de prueba de caja blanca y caja negra.
- Diseñar e implementar las pruebas, mediante la creación de los casos de uso de prueba de caja negra.
- Realizar pruebas de entornos especializados, específicamente prueba a la documentación del módulo Generador de Modelos.
- Hacer un resumen de la evaluación de las pruebas aplicadas.

Metodología utilizada.

En la primera etapa del estudio, se procedió a la revisión bibliográfica nacional e internacional, ordenando el conocimiento ya existente y satisfaciendo las necesidades de búsqueda, basándonos en los métodos teóricos a través del análisis o estudio de documentos de interés. El método analítico - sintético nos permitió descomponer el tema, además el enfoque sistémico nos posibilitó caracterizar todas las concepciónes, donde el método, Inductivo – deductivo nos facilitó interpretar los análisis de datos, así como descubrir las regularidades y relaciones entre los distintos componentes de la investigación para poder realizar las generalizaciones que parten por un lado de los análisis histórico – lógico a través del cual se profundizó en las tendencias, regularidades y cualidades del objeto de estudio, así como los argumentos que antecedieron al problema científico, en este sentido a traves del método comparativo se establecieron las semejanzas y diferencias con relación a las particularidades que precisamos con relación a los indicadores observables. Además se utilizó el método de expertos para darle validación externa a los resultados expuestos.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA.

1. Introducción.

En este capítulo brindaremos la fundamentación teórica sobre la que vamos a sentar las bases a partir de definir el concepto general del software estadísticos, haciendo una especial mención a la estructura funcional, a las características generales y a los modelos explicativos de los mismos.

Además profundizamos en las bibliografías más contemporáneas acerca del tema, protagonista principal de nuestro estudio, lo que nos permitió conocer los antecedentes sobre las investigaciones y trabajos relacionados con nuestro estudio enfatizando en la calidad y procedimientos del software.

También hacemos mención a los modelos de calidad y su relación con el software, analizándose trabajos relacionados con nuestro estudio, de los cuales se alcanzaron referencias importantes que definen el rol de ingeniero de prueba como elemento necesario dentro del análisis del proceso, para abordar finalmente las pruebas del software y de ellas sus principios, fundamentos, herramientas para la automatización de las mismas, presentamos el diseño de caso de prueba, como elemento esencial del proceso, en el que se encuentran las pruebas de caja blanca, caja negra, sus técnicas de aplicación y uso y además nos referiremos a las estrategias de prueba del software.

1.1. Análisis de software estadísticos.

En este punto se establece un análisis acerca de software estadísticos, permitiéndonos una comprensión lógica que nos asegura un mejor acercamiento, para poder entender su funcionamiento y así poder aplicar las pruebas con la mayor información posible.

1.1.1. Microset NT. Software estadístico.

El MICROSET NT es un sistema que permite procesar modelos de información sin que sea necesario realizar una programación específica para cada uno y que a la vez brinda grandes posibilidades, tanto en la validación de los datos primarios, como en las ediciones de tablas, de acuerdo con las características particulares de cada uno. Las entradas al sistema deben estar representadas por un modelo compuesto de una matriz simple con sus filas y columnas, debiendo tener un número de modelo

con cuatro caracteres numéricos y un código de fila que puede ser un número desde el 1 hasta el 99999999. El tamaño de la matriz del modelo debe tener como máximo 8000 celdas de datos, teniendo como límites de filas y columnas 3000 y 32 respectivamente. El control de la información se sustenta sobre los códigos del modelo, el centro informante, dos tipos de variantes opcionales, el número de la página y el código de la fila. El control de la información se sustenta sobre los códigos del modelo, el centro informante, dos tipos de variantes opcionales, el número de la página y el código de la fila.(PIQUERA. 1997).

Hacer un estudio acerca de los software estadísticos existentes en la actualidad, nos permitió construir importantes ideas entre las que se muestran similitudes en cuanto a la funcionalidad y operatividad de los mismos, por ello aparece a continuación las descripciones más novedosas de estos productos, atendiendo a su nivel de semejanza.

1.1.2. SPSS Base 15.0: El software estadístico integral.

Este software resuelve problemas de negocio e investigación para Windows, es un software estadístico y de gestión de datos para analistas e investigadores. A diferencia de otros paquetes estadísticos, SPSS es más fácil de utilizar, tiene un menor coste total de propiedad y abarca todas las etapas del proceso analítico. SPSS Base es una parte integral de este proceso, pues provee funciones para el acceso, gestión, preparación y análisis de los datos, y presentación de informes de resultados. Permite trabajar con módulos adicionales y otros productos de la familia SPSS, que ofrecen la funcionalidad necesaria para la planificación, recopilación de datos, puesta en marcha y adicionalmente incrementa la funcionalidad para manejar otras áreas que también abarca SPSS. A continuación se describen las características más resaltantes de SPSS Base, detallando la lista de *estadísticos*, los procedimientos para la gestión de datos y otras características que hacen de SPSS Base el *software estadístico* y de gestión de datos para el escritorio, líder en el mercado. SPSS para Windows. Facilita el rápido acceso, manejo y análisis de cualquier tipo de conjunto de datos, incluso datos de encuestas, bases de datos corporativas o datos descargados de la Web. Va más allá de las estadísticas resumidas y la matemática de filas y columnas. SPSS ofrece un amplio rango de procedimientos estadísticos para el análisis básico incluyendo: recuentos, tablas de contingencia, conglomerados, estadísticos descriptivos, análisis de factor, regresión lineal, análisis de conglomerados y regresión ordinal. Una vez completado el análisis, incluye datos en su base de datos con el “Asistente para exportar a bases de datos”. Para una

mayor capacidad de análisis, además cuenta con una amplia variedad de módulos que puede añadir el SPSS, como “SPSS Modelos de Regresión” y “SPSS Modelos Avanzados” que se centran en el análisis de los datos. (*Software SPSS.* 2006)

Como resultado del análisis bibliográfico se llegó a reconstruir las necesidades de aplicación de un nuevo sistema de interacción que permitiera una mejora a los tratamientos estadísticos llevados a cabo en nuestro país, desde este enfoque, en el que se partió de la situación actual, llegamos a la situación deseada a través del Sistema Integrado de Gestión Estadísticas.

1.1.3. Sistema Integrado de Gestión Estadística.

El Sistema Integrado de Gestión Estadística, es un sistema encargado del almacenamiento, control, gestión y manipulación de información estadísticas en Cuba. El proyecto cuenta con 4 módulos fundamentales encargados del control, funcionamiento y automatización de los principales flujos de trabajos que lleva a cabo la oficina nacional de estadísticas, permitiéndole a la misma la gestión de información de todos sus centros informantes así como el almacenamiento y manejo de la misma facilitando la entrega de partes y análisis estadísticos a los diferentes niveles del estado y/o del gobierno.

Una vez desarrollados los estudios de en los más novedosos controles y evaluadores de un software estadísticos, aparece el proceso y análisis de calidad, como indicador fundamental que destacaremos a continuación.

1.2. Calidad del software.

Empezaremos por definir algunos conceptos que se encuentran asociados a la calidad del software.

“Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente” R. S. Pressman (1992).

“El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas” ISO 8402 (UNE 66-001-92).

La calidad del *software* es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del *software*. El aseguramiento de la calidad de software se ha convertido en una necesidad prioritaria para las organizaciones que desarrollan software, ya sea para uso interno o para implementaciones externas en clientes, porque cada vez más los errores en el software repercuten directa o indirectamente en graves consecuencias para la organización.

1.3. Modelos de calidad.

En la actualidad, ha crecido la complejidad con la que se desarrollan sistemas de software para la industria, por lo que resulta difícil generar productos que cumplan completamente con las expectativas del cliente. Para equilibrar a esta circunstancias, han surgido una serie de modelos que facilitan a las organizaciones, generar productos que cumplan las expectativas del cliente e incluso las propasen, modelos que prometen ser la solución a los problemas de calidad, costo y tiempos de desarrollo; de éstos podemos mencionar a los modelos de calidad como: la norma ISO 9000-2000, la ISO/IEC TR 15504 y el modelo Capability Maturity Model (CMM).Estos modelos presentan una serie de ventajas y desventajas que a continuación, mencionaremos brevemente.(ALARCÓN Enero de 2004)

1.3.1. Norma ISO 9000-2000.

Es una norma internacional destinada a evaluar la capacidad de la organización para cumplir los requisitos del cliente, los reglamentarios y los propios de la organización.

Ventajas.

- Tiene un mecanismo de certificación bien establecido.
- Está disponible y es conocida.

Desventajas.

- No es específica para la industria de software.
- No es fácil de entender.
- No está definida como un conjunto de procesos.
- No es fácil de aplicar.

1.3.2. Capability Maturity Model (CMM).

Es un marco evolutivo organizado en cinco niveles para lograr la mejora continua de procesos.

Ventajas.

- Específico para el desarrollo y mantenimiento de software.
- Definido como un conjunto de áreas clave de procesos.
- Tiene un modelo de evaluación.
- Desde 1998 empezó a popularizarse en México.
- Existen organizaciones evaluadas.

Desventajas.

- Es un modelo extranjero, no internacional.
- No es fácil de entender.
- No es fácil de aplicar (pensado para organizaciones grandes).
- La mejora no está enfocada directamente a los objetivos de negocio.
- La evaluación es costosa y no tiene periodo de vigencia.
- Se está abandonando a favor de CMM-I.

1.3.3. ISO/IEC TR 15504.

Define el modelo de referencia de procesos de software y de capacidades de procesos que constituyen la base para la evaluación de procesos de software.

Ventajas.

- Específico para el desarrollo y mantenimiento de software.
- Fácil de entender (24 procesos, 16 páginas).
- Definido como un conjunto de procesos.
- Orientado a mejorar los procesos para contribuir a los objetivos del negocio.

Desventajas.

- No es práctico ni fácil de aplicar.
- Tiene solamente lineamientos para un mecanismo de evaluación.
- Todavía no es norma internacional. (ALARCÓN Enero de 2004)

Dada la importancia de obtener productos con la calidad esperada por los clientes, movida por la necesidad que se impone en el mundo de las nuevas tecnologías y productos de mayor complejidad, no solo podemos conformarnos con aplicar estos modelos que hemos visto y muchos más que existen en la actualidad, a pesar de ver todas las ventajas que nos proporcionan. Otro aspecto importante y sin restarle valor a dichos modelos, son las pruebas del software, que nos ayudan a lograr un producto con la menor cantidad de defectos antes de que el producto sea entregado a los clientes, ya que un error que desde el

punto de vista de codificación puede ser relativamente simple de corregir, pero puede resultar muy difícil y costoso de detectar y pueden llegar a tener graves efectos en la organización. Normalmente el equipo de desarrollo del proyecto se encuentra presionado por la necesidad de cumplir con las fechas establecidas en el cronograma y el *proceso de prueba,* no se cumple o se ejecuta de una manera desorganizada, sin método y sin considerar los tiempos establecidos para esta fase. El resultado es un software sin las pruebas mínimas requeridas y sin el nivel de calidad esperado.

1.4. Rol de ingeniero de pruebas.

Por todas estas razones surge la necesidad de contar con un grupo especializado en procesos de aseguramiento de calidad y específicamente en pruebas de software, para que pueda ayudar a alcanzar el nivel de calidad esperado del software y conseguir el cumplimiento exitoso del proyecto. El responsable de la realización de las pruebas y otras actividades es el ingeniero de pruebas de software. *El Ingeniero de pruebas de software*, *es el* responsable de ejecutar el conjunto de pruebas establecidas para el software, registrar los defectos hallados y ejecutar las pruebas que garanticen la solución de los defectos y la estabilidad del resto del software. También existe responsabilidad compartida en el mejoramiento de procesos, junto con el equipo de desarrollo, quienes deben realizar la identificación de las causas de no conformidades y la definición de estrategias para el mejoramiento de procesos y por consecuencia del producto.

El ingeniero de prueba es responsable de planear las pruebas, es decir que deciden los objetivos de prueba adecuados, las planifican, también detallan los procedimientos de prueba correspondientes y son responsables de la evaluación de las pruebas escogidas para su realización cuando estas se ejecutan.

1.5. Introducción a las pruebas del software.

Pruebas de software no es lo mismo que aseguramiento de calidad de software. Las pruebas de software son una parte del proceso de aseguramiento de calidad; realizar pruebas a un sistema de información no significa necesariamente que el proceso de desarrollo esté asegurado y tampoco que de manera directa esté mejorando.

Autores como Krutchen, Pressman, Pfleger, Cardoso y Sommerville, entre otros afirman que el proceso de aplicación de pruebas debe ser considerado a lo largo de todo el ciclo de vida del proyecto para así obtener un producto de alta calidad, que cumpla con los objetivos y necesidades de los clientes. La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado, podemos decir que no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: podemos probar la funcionalidad de los primeros prototipos; probar la estabilidad, cobertura y rendimiento de la arquitectura o probar el producto final, etc. La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. Para la realización de las pruebas se establecen normas que contribuyen a lograr acertadamente sus objetivos entre los que tenemos: La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. Es una técnica experimental para la búsqueda de errores en los programas. Con La prueba no se puede asegurar la ausencia de defectos; sólo se puede demostrar que existen defectos en el software.

1.6. Fundamentos y principios de las pruebas del software.

Antes de aplicar los diferentes métodos para el diseño de casos de prueba efectivos, siempre los ingenieros de prueba deben tener en cuenta los principios básicos que guían el buen desempeño de las pruebas del software.

Principios de las pruebas de software.

- A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos del cliente, para medir su trazabilidad.
- Las pruebas deberían planificarse antes de comenzar.
- El principio de Pareto es aplicable a la prueba del software “donde hay un defecto, hay otos”.
- Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
- No son posibles las pruebas exhaustivas.
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente.
- Se deben evitar los casos de prueba no documentados ni diseñados con cuidado.
- No deben suponerse planes de prueba suponiendo que prácticamente no hay defectos en los programas y por tanto dedicar pocos recursos a las pruebas.(S.PRESSMAN and INCE).

El principal objetivo dentro de la etapa de pruebas es certificar la calidad del producto desarrollado. Esta etapa implica además los que veremos a continuación:

- Verificar que todos los requisitos se han implementado correctamente de acuerdo a las expectativas de los usuarios.
- Verificar la interacción y la integración adecuada de los componentes.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo

1.7. Diseño de casos de uso de prueba.

El diseño de pruebas para el software o para cualquier producto de ingeniería puede necesitar un esfuerzo igual que el propio diseño inicial del producto. A pesar de toda la importancia que requiere la realización de las pruebas, que se tratado anteriormente. A menudo los ingenieros de prueba de software tratan las pruebas como algo sin importancia y le restan valor y casi siempre desarrollan casos de prueba, que aparentan ser adecuadas, pero que tienen poca garantía y no son totalmente completos y se olvida

uno de los principios fundamentales de las pruebas, de diseñar pruebas que posean una gran probabilidad de encontrar errores con el mínimo esfuerzo y tiempo posibles. Todo producto de ingeniaría y de cualquier otro campo se puede probar mediante una de estas formas: *prueba de caja blanca o prueba de caja negra.* Ver Fig. 1.1. En esta figura se representa gráficamente la filosofía de las pruebas de caja blanca y caja negra. Como se puede observar las pruebas de caja blanca necesitan conocer los detalles del código, mientras que las de caja negra únicamente necesitan saber el objetivo o funcionalidad que el código ha de proporcionar.

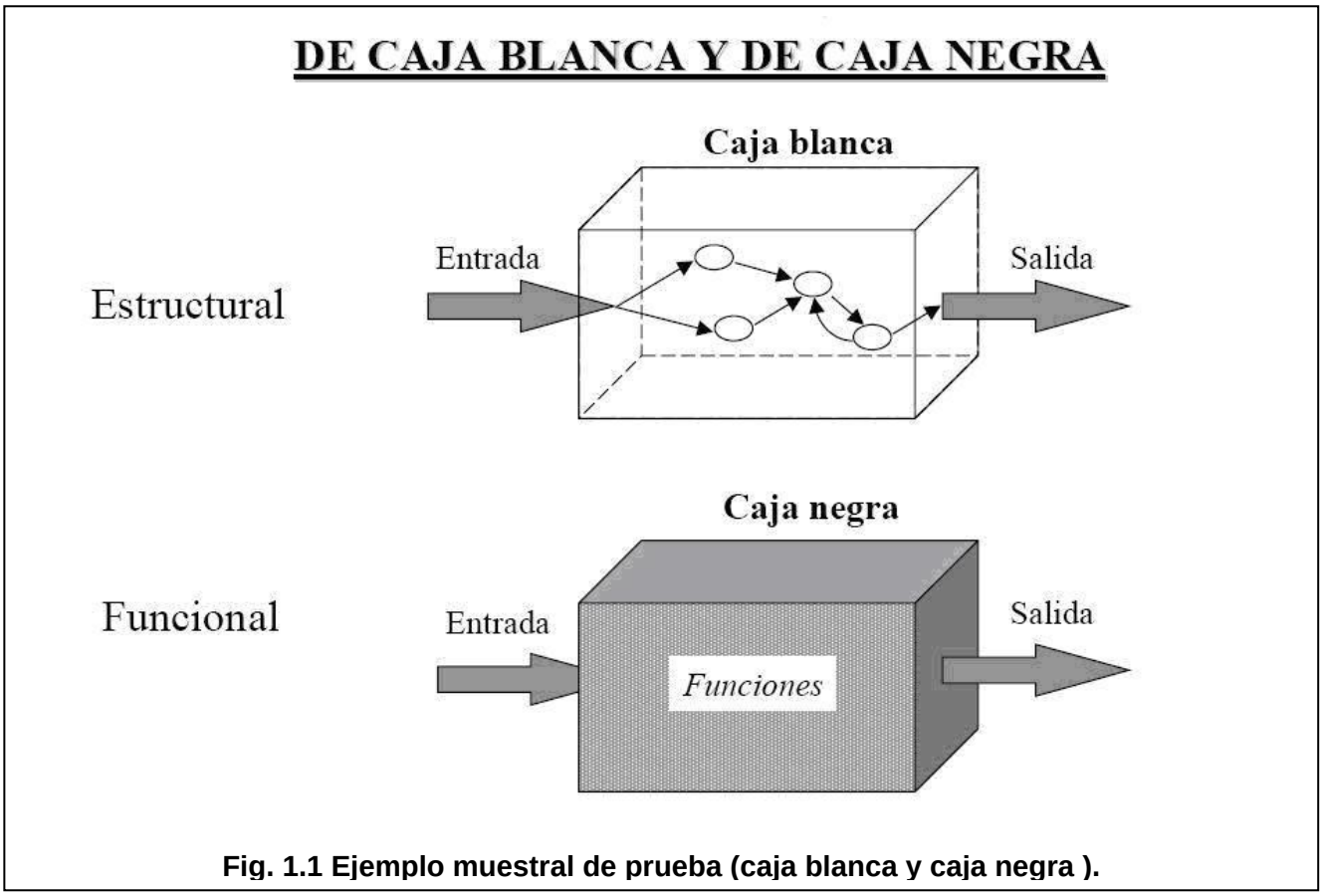

Fig. 1.1 Ejemplo muestral de prueba (caja blanca y caja negra).

1.8. Prueba de caja blanca.

Las pruebas de caja blanca realizan un seguimiento del código fuente según se van ejecutando los casos de prueba, de manera que se determinan de forma concreta las instrucciones, bloques, en los que existen errores. Conociendo el funcionamiento del producto se pueden desarrollar pruebas *prueba de caja blanca*, para asegurar que todas las piezas del programa concuerdan, es decir que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada, es decir analiza los caminos de ejecución. Mediante los casos de prueba de caja blanca se pueden derivar casos de prueba que garanticen que:

- Se ejecutan por lo menos una vez todos los caminos independientes de cada módulo.
- Que se ejercitan todas las decisiones lógicas en sus vértices verdaderos y falsos.
- Que se ejecutan todos los bucles en sus límites y con sus límites operacionales.
- Que se ejecutan las estructuras de datos internas para asegurar su validez.

En ocasiones resultan necesarias realizar las pruebas de caja blanca para detectar errores que se producen cuando nos encontramos ante las siguientes situaciones:

- Los errores lógicos y suposiciones, los cuales son proporcionales a la probabilidad de que se ejecute algún cambio en el programa.
- Cuando pensamos que un camino lógico tiene pocas posibilidades de ejecutarse y de hecho se ejecuta de forma normal y esto significa o conlleva a que nuestras suposiciones sobre el flujo de control y datos conlleven a errores de diseño y se encuentran solamente cuando comienza la prueba de *camino básico.*
- También podemos encontrar errores tipográficos mediante estas pruebas, lo cual es casi imposible de detectar mediante otro tipo de prueba. .(PRESSMAN 2005).

1.9. Técnicas de prueba de caja blanca.

A continuación se describen varias técnicas de caja blanca mediante las cuales podemos realizar los casos de uso de prueba según nos seas de mayor utilidad, en nuestro proyecto.

1.9.1. Prueba de camino básico.

Anteriormente se hablaba sobre ***prueba de camino básico*****,** la cual es una técnica de la prueba de *caja blanca* y será descrita a continuación:

Esta técnica es de gran importancia, ya que permite obtener una medida de la complejidad lógica de un diseño y utilizar esta medida como guía para la definición de los caminos básicos posible de ejecución.

Para facilitar el trabajo de la prueba de *camino básico*, puede ser de gran utilidad la herramienta estructura de datos denominada ***matriz de grafo****.* Una matriz de grafo es una matriz cuadrada cuyo tamaño de filas y columnas es igual al número de nodos del grafo de flujo, cada fila y cada columna corresponde a un nodo específico y las entradas de la matriz corresponden a las aristas que establecen las conexiones entre nodos, mediante esta técnica ,el análisis requerido para el diseño de los casos de prueba se pudiera automatizar de forma parcial o totalmente, siendo esto de gran ayuda para evitarnos trabajo y lograr mayor eficiencia en el desempeño de aplicación de las pruebas. La técnica de prueba del camino básico que se analizó, es una de las técnicas de caja blanca existentes, y aunque presenta muchas facilidades, es sencilla y efectiva, pero no es suficiente por si sola, a continuación se trataran otras variantes de la prueba de caja blanca, que amplían su responsabilidad y su calidad.

1.9.2. Prueba de condición.

La ***prueba de condición*** es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. Por tanto, los tipos posibles de componentes en una condición pueden ser: un operador lógico, una variable lógica, un par de paréntesis lógicos (que rodean a una condición simple o compuesta), un operador relacional o una expresión aritmética.

Esta prueba se lleva a cabo si una condición es incorrecta, entonces es incorrecto al menos un componente de la condición .Así, los tipos de errores de una condición pueden ser los siguientes:

- Error en un operador lógico.
- Errores en variables lógicas.
- Errores en paréntesis lógicos.
- Errores en operadores relacionales.
- Errores en una expresión aritmética.

Las pruebas de condición centran las pruebas en cada una de las condiciones del programa. Estas pruebas tienen como objetivo la detección de no solo los errores en las condiciones de un programa, sino también otros errores en dicho producto.

1.9.3. Prueba del flujo de datos.

El método de ***prueba de flujo de datos*** selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. La prueba de flujo de datos puede ser usada en áreas del software que son sospechosos, pero casi imposible de usar para probar grandes sistemas. Una estrategia sencilla de prueba de flujo de datos se basa en que se cubran al menos una vez cada cadena de definición-uso (o cadena DU), sin embargo se ha demostrado que dicha prueba no nos garantiza el cubrimiento de todas las ramificaciones de un programa, sin embargo, solamente no se garantizan el cubrimiento de una ramificación en situaciones raras como: if-then-else, en las que el then no tiene ninguna variable definida y no existe la parte else, pero tienen mucha utilidad para seleccionar cambios de prueba de un programa que contenga sentencias ***if o de bucles*** anidados.

1.9.4. Pruebas de bucles.

Los ***bucles*** son la piedra angular de la inmensa mayoría de los algoritmos implementados en software. A pesar de su importancia, les prestamos habitualmente poca atención cuando realizamos las pruebas del software. Los bucles no son más que segmentos controlados por decisiones. Así, la cobertura de ramas cubre plenamente la esencia de los bucles. Pero eso es simplemente la teoría, pues la práctica descubre que los bucles son una fuente inagotable de errores, todos triviales, algunos mortales. Un bucle se ejecuta un cierto número de veces; pero ese número de veces debe ser muy preciso, y lo más normal es que ejecutarlo una vez de menos o una vez de más tenga consecuencias indeseables. Y, sin embargo, es extremadamente fácil equivocarse y redactar un bucle que se ejecuta 1 vez de más o de menos.

Las pruebas ***de bucles*** es una técnica de prueba de caja blanca que se centra únicamente en la validez de las construcciones de bucles [BEI90]. Se pueden definir cuatro clases distintas de bucles: bucles simples, bucles concatenados, bucles anidados y bucles no estructurados. Las estructuras de bucles complejas es un lugar expuesto a errores, por lo que es de vital importancia realizar diseños de pruebas que ejerciten completamente las estructuras bucles. Veamos un ejemplo para ilustrar mejor lo antes expuesto:

Para un bucle de tipo WHILE hay que pasar 3 pruebas

- 0 ejecuciones
- 1 ejecución
- Más de 1 ejecución

Para un bucle de tipo REPEAT hay que pasar 2 pruebas

- 1 ejecución
- Más de 1 ejecución

1.10. Prueba de caja negra.

Conociendo la función específica para la que se diseñó el producto, se puede llevar a cabo pruebas que nos demuestren que cada una de las funciones es totalmente operativa y a la vez buscar errores en cada función, este tipo de prueba se le llama *prueba de caja negra,* también conocidas *como prueba de comportamiento*. Además permite obtener un conjunto de condiciones de entradas que permitan ejercitar totalmente todos los requisitos funcionales del programa. La prueba de caja negra no es una alternativa a las técnicas de *prueba de caja banca*, es más bien un complemento que intenta descubrir diferentes tipos de errores que los caja blanca .Se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Mediante los casos de prueba de la caja negra podemos indicar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

La prueba de la caja negra puede descubrir errores de Funciones incorrectas o ausentes entre las que encontramos:

- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

La prueba de caja blanca tiende a aplicarse durante las fases posteriores de la prueba a diferencia de la de caja negra que se realiza previamente en el proceso de prueba, además la prueba de caja negra ignora intencionadamente la estructura de control, centra su atención en el campo de información. Mediante las técnicas prueba de caja negra obtenemos un conjunto de casos de prueba que satisfacen los siguientes criterios:

1. Casos de prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable.
2. Casos de prueba que nos digan algo sobre la presencia o ausencia de clases de errores en lugar de errores asociados solamente con la prueba que estamos realizando.

1.11. Técnicas de prueba de caja negra.

A continuación se mostrarán las técnicas que se utilizan para el diseño de casos de uso de prueba de caja negra.

1.11.1. Partición equivalente.

La ***partición equivalente*** es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de uso ideal descubre de forma inmediata una clase de errores que de otro modo, requerirán la ejecución de muchos casos de prueba para detectar un error genérico. La prueba de partición equivalente reduce el número total de casos de prueba que hay que desarrollar, ya que se dirige a la definición de casos de prueba que descubran clases de errores. Las clases de entrada se conocen relativamente temprano en el proceso de

creación del producto de software, dado esta razón podemos pensar en utilizar la partición equivalente cuando se haya creado el diseño. Las clases de equivalencia se pueden definir de acuerdo con los siguientes criterios:

- Si una condición de entrada específica un rango***,*** se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia valida y dos no válidas.
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida
- Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Los casos de prueba se seleccionan de forma que se ejercite el mayor número de atributos de cada clase de equivalencia a la vez.

1.11.2. Análisis de valores límite.

Los errores por lo general tienden a presentarse con mayor frecuencia en los límites del campo de entrada que en el centro, para esto se utiliza la técnica de ***análisis de valores límites*** (AVL) para las pruebas, ya que esta nos lleva a una elección de casos de prueba que ejerciten los valores límites. Esta técnica de diseño de casos de prueba es un complemento de la partición equivalente. La AVL lleva a la elección de casos de prueba en los extremos de la clase y no se centra solamente en las condiciones de entrada, obteniendo casos de prueba para el campo de salida.

Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente, pero la prueba de límites será más completa y por tanto tendrá una mayor posibilidad de detectar errores, por lo que es más efectiva.

- Si una condición de entrada específica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justo por debajo y justo por encima de a y b, respectivamente.
- Si una condición de entrada específica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.

- Aplicar las directrices 1 y 2 a las condiciones de salida. Por ejemplo, supongamos que se requiere una tabla de (temperatura/presión) como salida de un programa de análisis de ingeniería. Se deben diseñar casos de prueba que creen un informe de salida que produzca el máximo (y el mínimo) numero permitido de entradas en la tabla.
- Si las estructuras de datos internas tienen límites preestablecidos (por ejemplo, una matriz que tenga un límite definido de 100 entradas), hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

1.11.3. Prueba de comparación.

Hay situaciones en las que la fiabilidad del software es algo absolutamente crítico. En ese tipo de aplicaciones a menudo se utiliza hardware y software redundante para minimizar la posibilidad de error. Cuando se desarrolla un software redundante varios equipos de ingeniería del software separados desarrollan versiones independientes de una aplicación, usando las mismas especificaciones .luego se prueban todas las versiones con los mismos datos de prueba, para verificar si todas proporcionan una salida idéntica, luego se ejecutan todas al mismo tiempo y se comparan los resultados en tiempo real. Estas versiones desarrolladas independientemente son la base de la técnica de prueba de caja negra denomina ***prueba de comparación***. Después de que producen varias implementaciones de la misma especificación, a cada versión del software se le proporciona como entrada los casos de prueba diseñados mediante cualquiera otra técnica de caja negra, se compararan las salidas y se verifican que sean idénticas, se asume que todas las implementaciones son correctas. Si la salida es distinta, se investigan todas las aplicaciones para determinar el defecto responsable de la diferencia en una o varias versiones.

1.11.4. Prueba de tabla ortogonal.

La prueba de la ***tabla ortogonal*** puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas. Este método es particularmente útil para encontrar errores asociados con fallos localizados, una categoría de error asociada con defectos de la lógica dentro de un componente software.

Tabla ortogonal.

Caso de prueba	Parámetros de prueba			
	A1	A2	A3	A4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	1
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Para lograr un mayor entendimiento del uso de la tabla ortogonal, consideremos el ejemplo de la función enviar para una aplicación de fax, los cuatro parámetros A1, A2, A3, A4 se le pasan a dicha función y cada uno tiene tres valores diferentes, por ejemplo A1 tona lo valores:

A1= 1, enviar ahora.
A2=2, enviar dentro de 2 horas.
A3=3 enviar a las 5.00 pm.
A2, A3 y A4, podrán tomar también los valores de 1, 2, 3 representando otras funciones enviar. Dado un número relativamente pequeño de parámetros de entrada y valores diferentes, es posible realizar una prueba exhaustiva, la cantidad de pruebas necesarias es $3^4 = 81$.

Con esta prueba podemos diseñar casos de prueba que faciliten una cobertura máxima de prueba con un número razonable de casos de prueba. La prueba de la tabla ortogonal nos permite proporcionar una buena cobertura de prueba con bastantes menos casos de prueba que en la estrategia exhaustiva.

Esta prueba según [PHA97] valora el resultado de las pruebas utilizando la tabla ortogonal de la siguiente manera:

1 Detecta y aísla todos los defectos de modalidad simple.
2 Detecta todos los fallos de modalidad doble.
3 Detecta los fallos multimodales.

1.12. Pruebas de entornos especializados. Arquitecturas y Aplicaciones.

A medida que el software de computadora se ha hecho más complejo, también ha aumentado la necesidad de realizar pruebas especializadas. Los métodos de prueba de caja blanca y de caja negra son adaptables a todos los entornos, arquitecturas y aplicaciones pero a veces se necesitan criterios y enfoques específicos para las pruebas. A continuación analizaremos diferentes criterios para la aplicar las pruebas a entornos, arquitecturas y aplicaciones especializadas.

1.13. Prueba de interfaces gráficas de usuario (IGUs).

A medida que el software se ha hecho más complicado, ha aumentado también la complejidad de las interfaces graficas de usuario, originando mayores dificultades en su diseño y para la ejecución de los casos de uso de prueba a la vez que también se requieren de menos tiempo y posen mayor exactitud, para garantizar que las pruebas vayan dirigidas a los datos específicos y programas objetos, se pudiera utilizar, *los grafos de modelado de estado finito,* pero debido al gran número de permutaciones asociadas a las operaciones de las ***interfaces gráficas de usuario***, sería de gran ayuda utilizar herramientas automáticas.

1.14. Prueba de la documentación y facilidades de ayuda.

La presencia de errores en la documentación puede ser tan dañina para la aceptación de los programas por los clientes, como los errores de código, ya que es de vital importancia que el manual de usuario y los resultados obtenidos tengan una correspondencia entre ellos. Por esta razón la ***prueba de documentación*** debería formar parte imprescindible de cualquier plan de prueba del software. Para

implantar esta prueba se pueden usar varios métodos de caja negra como: el análisis de la partición equivalente o de los valores límites para definir varias clases de entradas e interacciones asociadas.

1.15. Prueba de sistemas de tiempo-real.

La estrecha relación que existe entre el software ***de tiempo-real*** y su entono de hardware también puede introducir problemas en la prueba. Las pruebas del software deben considerar el impacto de los fallos del hardware sobre el proceso del software. Puede ser muy difícil simular de forma realista esos fallos, sin embargo, se puede proponer una estrategia en cuatro pasos:

1 Prueba de tareas.
2 Prueba de comportamiento.
3 Prueba ínter tareas.
4 Prueba del sistema.

A modo de resumen podemos decir que los métodos de prueba especializados comprenden una amplia gama de capacidades del software y áreas de aplicación. Las interfaces gráficas de usuario, las arquitecturas cliente/servidor, la documentación y facilidades de ayuda, y los sistemas de tiempo real requieren directrices y técnicas especializadas para la prueba del software.(PRESSMAN 2005)

1.16. Estrategias de prueba de software.

Una estrategia de prueba de software, es una unificación de las técnicas de diseño de casos de prueba y sus pasos bien detallados, que posibilitan el correcto cumplimiento de todas las especificaciones del cliente y dan como resultado una correcta construcción del software. También se describen, el objetivo de la prueba, los métodos que se van a necesitar para poder desarrollar cada una de esas pruebas. De ser posible una estrategia de prueba del software debe ser suficientemente manejable, para promover la creatividad y la adaptabilidad para que pueda ser usada en nuevos sistemas, con esto no estamos contradiciendo la posibilidad de que a la hora de plantearla también debe ser lo suficientemente rígida y que nos permita promover y seguir una planificación razonable a medida que progrese el proyecto. Existen diferentes estrategias de prueba que se llevan a cabo en la actualidad entre las que podemos mencionar :

:

- ***Validación y Verificación***. La validación se consigue mediante una serie de pruebas de la caja negra que demuestran la conformidad con los requisitos, la *validación* no es más que el conjunto de actividades que aseguran que el software construido se ajusta a los requisitos del cliente y se comparan con el sistema que ha sido construido y la *verificación* es el conjunto de actividades que aseguran que el software implementa correctamente una función específica. Una correcta estrategia de Validación y Verificación nos ayudará a mejorar y asegurar la calidad de los productos y además permitirá reducir los costes de corrección de errores.

- La prueba de ***Unidad***, que se centra en el proceso de verificación en la menor unidad del diseño del software EL MÓDULO. Algunas consideraciones que debemos tener en cuenta para esta prueba es que:
  1 Se debe probar la interfaz del módulo para asegurar que le información fluye de forma adecuada hacia y desde la unidad que se esta probando.
  2 Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente conservando su integridad durante todos los pasos de ejecución del algoritmo.
  3 Se prueban las condiciones límites establecidas como restricciones, para asegurar que el módulo funciona correctamente.

- La prueba de ***Integración***. Se basa en la integración de todos lo módulos para comprobar sus interfaces en el trabajo conjunto.

- La prueba ***Funcional***. Cuando el software esta totalmente ensamblado se prueba como un conjunto para comprobar si cumple o no los requisitos funcionales, además de los requisitos de rendimiento, seguridad, etc.

- La prueba de ***Sistema***. El software ya validado se entrega con el resto del sistema para probar su funcionamiento conjunto.

- La prueba de ***Aceptación***. El producto final se pasa a la prueba de aceptación para que el usuario compruebe en su propio entorno de explotación, si lo acepta en las condiciones que esta o no. Entre otras existentes.

  La estrategia para producir grandes programas de gran calidad es, en primer lugar, eliminar todos los defectos de los módulos, que forman estos grandes proyectos, lo cual se pone de manifiesto en el software analizado en este trabajo. Muchas veces se plantea que muchos de los defectos en los programas grandes están en el sistema y no en los módulos, esto sin embargo no es cierto. Con pocas excepciones, los defectos en los grandes sistemas están en los módulos que lo constituyen. Estos defectos se pueden dividir en dos clases: aquellos que implican solamente un módulo y los que implican interacciones entre módulos. La calidad de los grandes programas depende de la calidad de los pequeños programas que lo forman. (PRESSMAN 2005).

1.17. Herramientas para el entorno de pruebas.

Las herramientas que dan soporte a las actividades de calidad específicamente a las pruebas del software, ya sean *pruebas de integración*, *pruebas de sistema*, *diagnóstico o afinado de las aplicaciones,* en los entornos de preproducción o certificación y los de producción, que habitualmente son difíciles de realizar de una forma integrada, comprenden herramientas de análisis estático, automatización de pruebas *funcionales*, de *carga*, de *rendimiento*, de *estrés*... el *depurado del código*, etc. Entre las herramientas de mayor utilidad para los entornos de pruebas tenemos, entre otras:

1.17.1. ChecKing de ALS.

ChecKing es la herramienta de monitorización del proceso de desarrollo software y sus resultados, que cubre las necesidades de organizaciones que desean controlar la calidad del software antes de su puesta en producción. Para ello la herramienta sigue un modelo de métricas en el que se integran medidas obtenidas automáticamente del proceso de desarrollo (actividad, requisitos, defectos y cambios) y de elementos analizables del software: código fuente, documentación del proyecto, scripts de construcción, y scripts de pruebas. ChecKing añade transparencia al ciclo de desarrollo, capturando, integrando y presentando de forma automática dichas métricas, obtenidas automáticamente través de diversas herramientas y sistemas externos: analizadores, gestores de defectos, sistemas de control de versiones, herramientas de pruebas y otras herramientas usadas por los desarrolladores.

1.17.2. JTest de Parasoft.

Herramienta de Parasoft, que ayuda a prevenir errores por medio de su capacidad de análisis estático personalizable, que permite hacer cumplir automáticamente alrededor de 300 estándares de codificación de la industria, crear y hacer cumplir estándares propios de codificación, o construir estándares destinados a un proyecto o grupo en particular. Jtest ofrece varios avances para las industrias desarrolladoras de software de alta calidad. Estos avances tecnológicos están concentrados en la realización de pruebas, para ayudar a los equipos a verificar de manera automática la funcionabilidad de aplicaciones cada vez mas complejas, en empresas con sistemas en permanente cambio, todo esto para generar un incremento en la satisfacción del cliente, una reducción en tiempo de entrega del software así como una disminución del riesgo de generar software defectuoso o con problemas de vulnerabilidad.

1.17.3. SOATest de Parasoft.

SOATest proporciona verificación instantáneas de servicios Web, simplifica el desarrollo del equipo de calidad, automatiza las pruebas funcionales de cliente/servidor, pruebas de regresión, pruebas de carga/rendimiento y más

1.17.4. TEST de Parasoft.

TEST es una unidad de pruebas automatizada y productos de análisis de código estándar, que trabaja sobre clases escritas en la plataforma Microsoft .NET, sin requerir que los desarrolladores realicen un solo caso de prueba.

1.17.5. Herramienta más potente y flexible de modelado en UML, Enterprise Architect (EA).

Enterprise Architect combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación. EA puede equipar a su equipo entero, incluyendo analistas, evaluadores, administradores de proyectos, personal del control de calidad, equipo de desarrollo y más. Enterprise Architect es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, ***pruebas*** y mantenimiento. EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. Todo este conjunto de características han hecho de EA la herramienta de modelado de elección para usuarios en más de 60 países en todo el mundo. Estas son algunas de sus características que lo hacen ser una herramienta de alta calidad.(*Enterprise Architect - Herramienta de diseño UML* 2007)

Modelado basado en el equipo.

- Archivos compartibles o modelos basados del repositorio.
- Seguridad incorporada y administración de permisos.

Rápido y escalable.

- Desde un sólo a usuario a grandes equipos.
- Soporte de repositorio.
- Rápido para descargar y usar – aún con grandes modelos.

Modelado visual por excelencia.

- Ingeniería reversa de código fuente en 10 lenguajes.
- Importar esquemas de bases de datos.

Soporte para Prueba.

Soporta ***pruebas*** de Unidad, pruebas de Integración, pruebas de sistema, pruebas de aceptación, escenarios, reporte de todos los detalles soportados en el generador del documento Revisiones Técnicas Formales.

1.17.6. Herramienta de prueba NUnit.

NUnit es una herramienta que se encarga de analizar ensamblados generados por .NET, interpretar las pruebas inmersas en ellos y ejecutarlas. Es una herramienta Open Source. Utiliza atributos personalizados para interpretar las pruebas y provee además métodos para implementarlas. En general, NUnit compara valores esperados y valores generados, si estos son diferentes la prueba no pasa, caso contrario la prueba es exitosa. NUnit carga en su entorno un ensamblado y cada vez que lo ejecuta, o mejor, ejecuta las pruebas que contiene, lo recarga. Esto es útil porque se pueden tener ciclos de codificación y ejecución de pruebas simultáneamente, así cada vez que se compile no tiene que volver a cargar el ensamblado al entorno de NUnit si no que este siempre obtiene la última versión del mismo. NUnit ofrece una interface simple que informa si una prueba o un conjunto de pruebas fallaron, pasó o fue ignorada.(*NUnit.* 2006)

CAPÍTULO 2

PLANIFICACIÓN DEL PROCESO DE PRUEBA.

2. Introducción.

En este capítulo se desarrollarán una serie de procedimientos, los cuales fueron obtenidos luego de aplicar toda la fundamentación teórica descrita en el capítulo 1, que nos posibilitarán su posterior aplicación, para realizar los diferentes artefactos, que nos proporcionan los resultados esperados como cumplimiento de los objetivos planteados en este trabajo de diploma, posibilitando el buen desempeño de las pruebas que se realizarán al módulo Generador de Modelos. Para la confección de los diferentes procedimientos y artefactos de este trabajo de tesis se utilizaron las plantillas definidas por el Proceso Unificado del Software (RUP), ya que mediante estas plantillas logramos una mayor organización de nuestro trabajo, queda constancia de todo lo que se hizo, le posibilitamos el trabajo al personal del equipo de calidad, para que se hagan las revisiones y contribuir a obtener un producto de alta calidad. A continuación se muestra todo el proceso que se lleva a cabo para la realización de las pruebas.

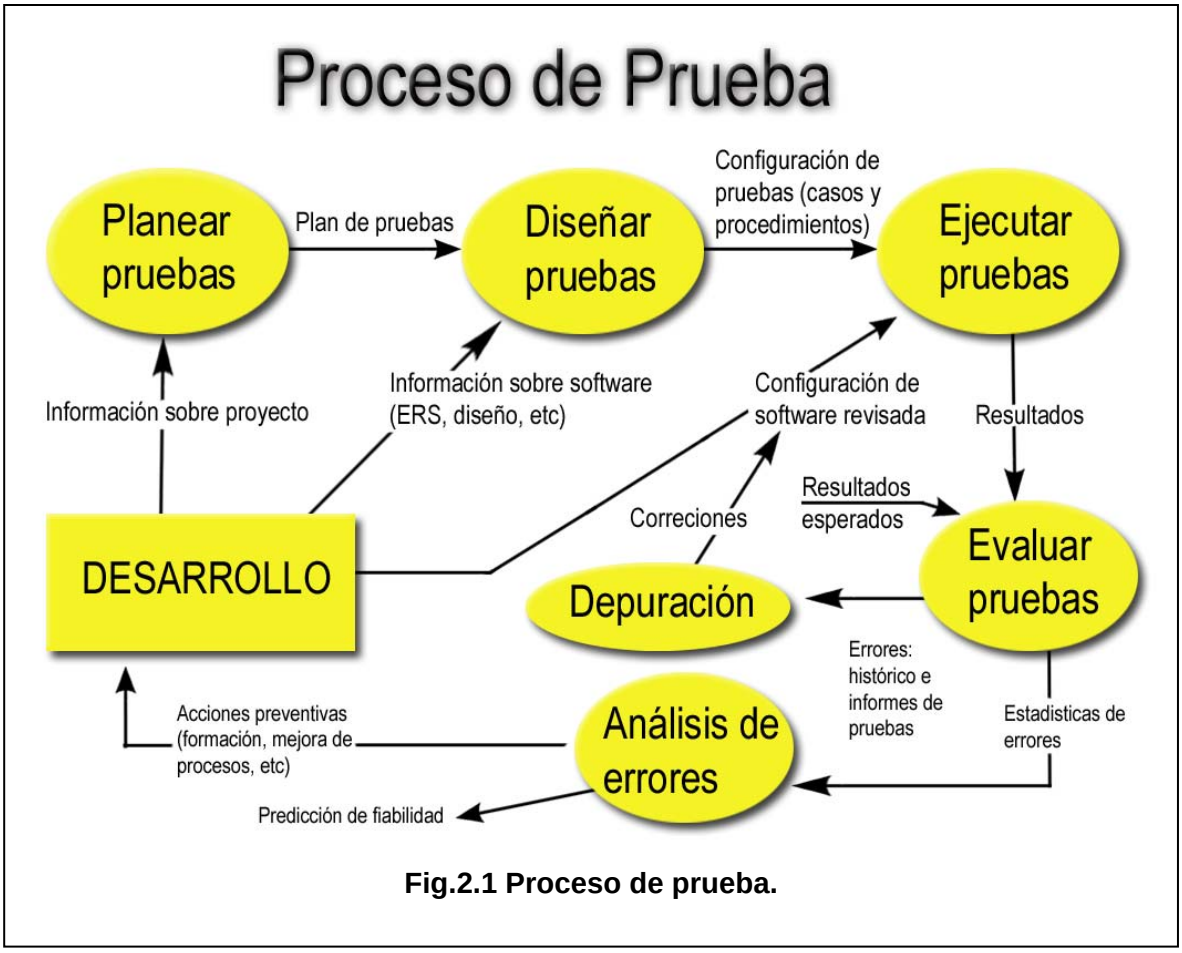

Fig.2.1 Proceso de prueba.

Dentro del proceso de prueba como se muestra en la Fig.2.1 se deben realizar varias actividades para que esta etapa se desarrolle con la calidad que requiere la misma. Bueno primeramente antes de comenzar a planear cualquier prueba debemos conocer el proyecto,(los requisitos planteados por los clientes, como funciona), informarnos antes de diseñar los casos de uso, no podemos hacerlo sin antes tener conocimiento sobre el funcionamiento del producto, ya una vez que contamos con la mayor información acumulada pasamos a planear y diseñar las pruebas, previamente debe de hacerse la selección de las técnicas que serán usadas para la aplicación de las mismas, ya previsto todo lo antes planteado se procede a la ejecución de las pruebas, luego se evalúan los resultados obtenidos, es decir el porcentaje de errores encontrados, etc. De inmediato se produce el proceso de corrección, depuración y análisis de los errores por parte de los desarrolladores y el equipo de ingenieros de pruebas, logrando

implantar y llevar a cabo todos los pasos incluidos en el proceso de prueba, obtendremos productos con el éxito esperado y que satisfaga las necesidades de los clientes .

A continuación se hace una breve descripción de la funcionalidad del módulo Generador de Modelos al cuaL se le aplicaron las pruebas y es objeto de estudio para los diferentes artefactos realizados. El mismo tiene el objetivo de automatizar el proceso relacionado con la gestión de modelos, bases metodológicas, cuadres y nomenclaturas correspondientes, a las oficinas estadísticas. Una vez que el usuario ingresa al sistema, puede realizar varias actividades, como son gestionar, guardar, exportar, imprimir y enviar los modelos, las bases metodológicas, los cuadres y nomenclaturas correspondientes, además tiene opciones para importar las bases metodológicas y nomenclaturas de los modelos, en el caso de las gestiones se puede crear, modificar o eliminar algún documento. En la Fig. 2.2 se muestra su interfaz correspondiente.

Fig.2.2 .Interfaz (Sistema Integrado de Gestión Estadística), módulo Generador de Modelos.

2.1. Procedimiento de prueba.

Un procedimiento de prueba específica como realizar uno o varios casos de prueba o partes de estos, un procedimiento de prueba puede ser tan simple como por ejemplo una instrucción para un usuario sobre como ha de realizar una caso de prueba manualmente o puede ser una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de prueba. Un procedimiento de prueba específica el cómo llevar a cabo un caso de prueba y a menudo se puede reutilizar un procedimiento de prueba para diversos casos de prueba y reutilizar varios procedimientos de prueba para un caso de uso de prueba. Ver Fig.2.3.

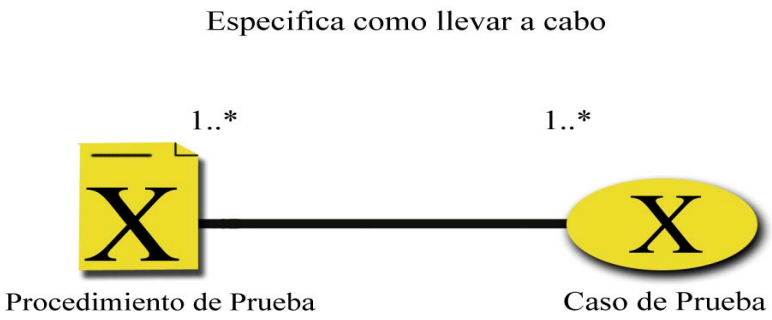

Fig.2.3. Ejemplo de asociación entre procediendo de prueba y caso de uso de prueba.

2.1.2. Procedimientos de pruebas de caja negra, para los casos de uso Gestionar Modelos y Gestionar Nomenclatura.

Se necesita un procedimiento de prueba para que el usuario lleve a cabo el caso de uso de prueba, Gestionar Modelo, este caso de uso brinda la posibilidad de crear, modificar o eliminar los modelos que necesite para el control estadístico, además del caso de uso Gestionar Nomenclatura, en el cual se pueden crear, actualizar y eliminar las nomenclaturas, en cada caso de uso se atendió de manera separada el flujo central del alterno, lo cual detallaremos a continuación en la descripción de los procedimientos de prueba para dichos casos de uso.

No.	Prueba	Acciones a realizar	Resultado esperado
1	Creando un Nuevo modelo.	Seleccione Nuevo Modelo a Crear. Debe escoger el tipo de modelo a Crear de: Talón Abierto o Talón Cerrado .Deben introducirse los datos del modelo en el **Anexo 1**. Obvie concientemente alguno de los campos: número, subnúmero, descripción, día de captación, periodicidad. Introduzca valores fuera de los rangos mostrados en el **Anexo 2**. Finalmente introduzca correctamente los datos y presione siguiente.	Se comprueba la obligatoriedad de los campos número, subnúmero, descripción, día de captación, periodicidad. y el chequeo de rango que sobre estos campos se realiza.
1.1	Creando un Nuevo Modelo.	Cree un modelo, que ya se encuentre registrado.	Se muestra un mensaje de error y le da La posibilidad de cambiar el nombre del modelo.
2	Modificando Modelos	Seleccione la opción de abrir y escoja el modelo, modifique los campos que desee y acepte la acción.	El sistema verifica los datos y si están correctos modifica el modelo
2.1	Modificando Modelos	Modifique algún dato de manera incorrecta.	Se indican los campos incorrectos, para su corrección.
3	Eliminando Modelos.	Seleccione la opción de eliminar el modelo en el que esta trabajando.	Se elimina el modelo.
3.1	Eliminando Modelos.	Seleccione la opción de eliminar. y no se esta trabajando en ningún modelo.	Se muestra un mensaje de error.

4	Creando la nomenclatura.	Seleccione Nueva Nomenclatura a Crear, el tipo de modelo a Crear debe ser: Talón Abierto .Deben introducirse los datos de la nomenclatura en el Anexo 3. Obvie concientemente alguno de los campos: tipo de indicador, código, indicador, código, unidad de medida. Introduzca valores fuera de los rangos mostrados en el Anexo 4. Finalmente introduzca correctamente los datos y presione siguiente.	Se comprueba la obligatoriedad de los campos tipo de indicador, código, indicador, código, unidad de medida y el chequeo de rango que sobre estos campos se realiza.
4.1	Creando la nomenclatura.	Cree una nomenclatura que se haya registrada anteriormente	Se indica que ya existe esa nomenclatura y debe cambiar el nombre
5	Actualizando la nomenclatura.	Seleccione la opción de abrir y escoja el modelo, actualice la nomenclatura que desee y acepte la acción.	El sistema verifica los datos y si están correctos actualiza la nomenclatura.
5.1	Actualizando la nomenclatura.	Actualice alguna nomenclatura de manera que esté incorrecta.	Se indican las nomenclaturas incorrectas, para su corrección.
6	Eliminando la Nomenclatura	Seleccione la opción de eliminar la nomenclatura en el modelo que este trabajando.	Se elimina la nomenclatura
6.1	Eliminando la Nomenclatura	Seleccione un modelo no tiene nomenclatura.	Se muestra un mensaje de error

2.1.3. Procedimiento de prueba de caja blanca.

A continuación ejemplificamos mediante el grafo de la Fig.2.4, el procedimiento a realizarse para una prueba de caja blanca, utilizando la técnica de camino básico y explicamos los pasos a seguir para su ejecución

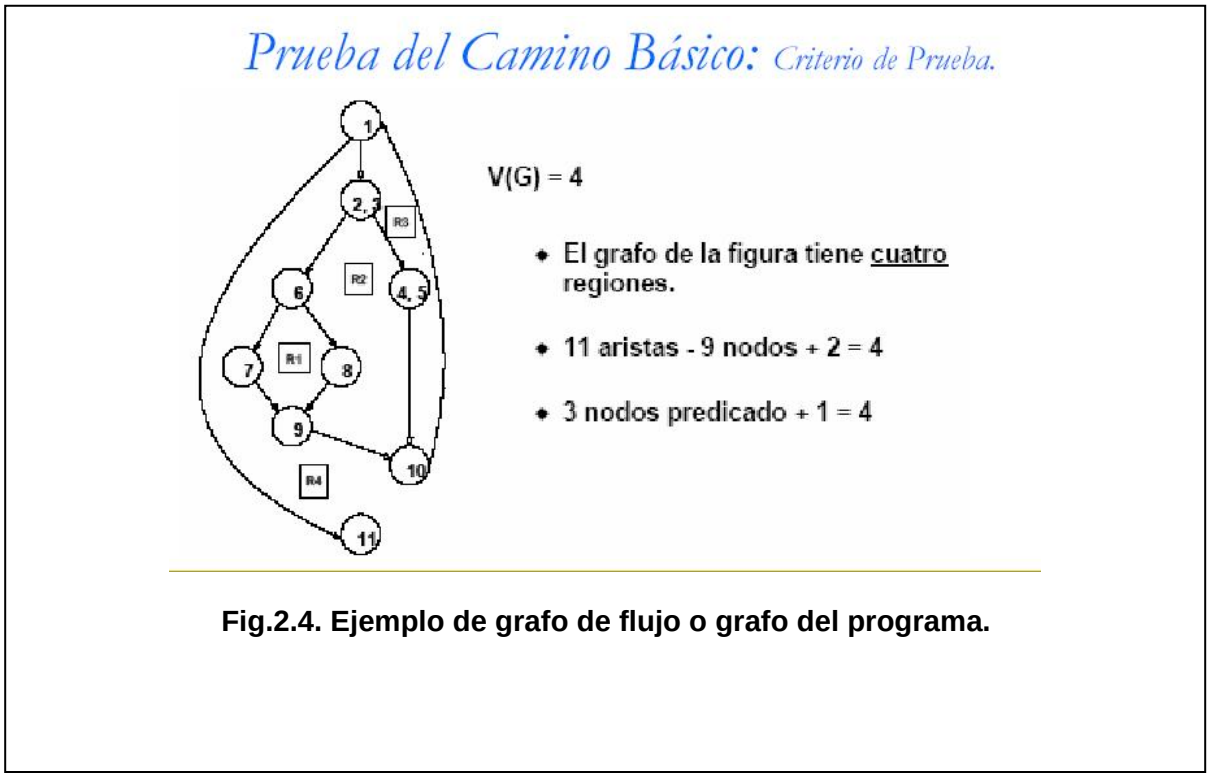

Fig.2.4. Ejemplo de grafo de flujo o grafo del programa.

Componentes del Grafo de Flujo:

Arista: Representa flujo de control.

Nodo: representa un conjunto de sentencias.

Región: Área limitada por asitas y nodos.

Se considera como un área + lo que esta fuera del grafos (otra región).

Pasos a seguir.

1 Debe introducir una sencilla notación, para representar el flujo de control (grafo de flujo o grafo del programa), utilizando el código o el diseño como base.

2 Se debe calcular la complejita ciclomática, de una de estas formas fundamentales:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

V (G)=A-N+2

Donde: A=al número de aristas del grafo.

N=al número de nodos del grafo.

V (G)=P+1

Donde: P= es el numero de nodos predicados contenidos en el grafo.

3 Debe definir todos los caminos lógicos independientes del conjunto básico del programa.

4 Desarrollar todos los casos de prueba que ejerciten la ejecución de cada camino del conjunto básico de sentencias al menos una vez.

5 Se deben escoger los datos de forma que las condiciones de los nodos predicados estén adecuados.

6 Se ejecutan cada uno de los casos de uso de prueba y se comparan los resultados obtenidos con los esperados.

7 Se evalúan los resultados.

2.2. Estrategia de prueba.

Mediante esta estrategia de prueba pretendemos garantizar la integración de las técnicas de prueba del software y de los distintos métodos que seleccionamos para logar que el módulo Generador de Modelos sea un producto con la menor cantidad de errores y que se pueda en un futuro reutilizar para futuros proyectos. Luego de hacer un estudio y análisis de las estrategias del software existentes, las cuales fueron brevemente descritas en el capitulo anterior decidimos utilizar la siguiente, la cual se ajusta a nuestros objetivos planteados anteriormente como se muestra a continuación en la Fig. 2.5.

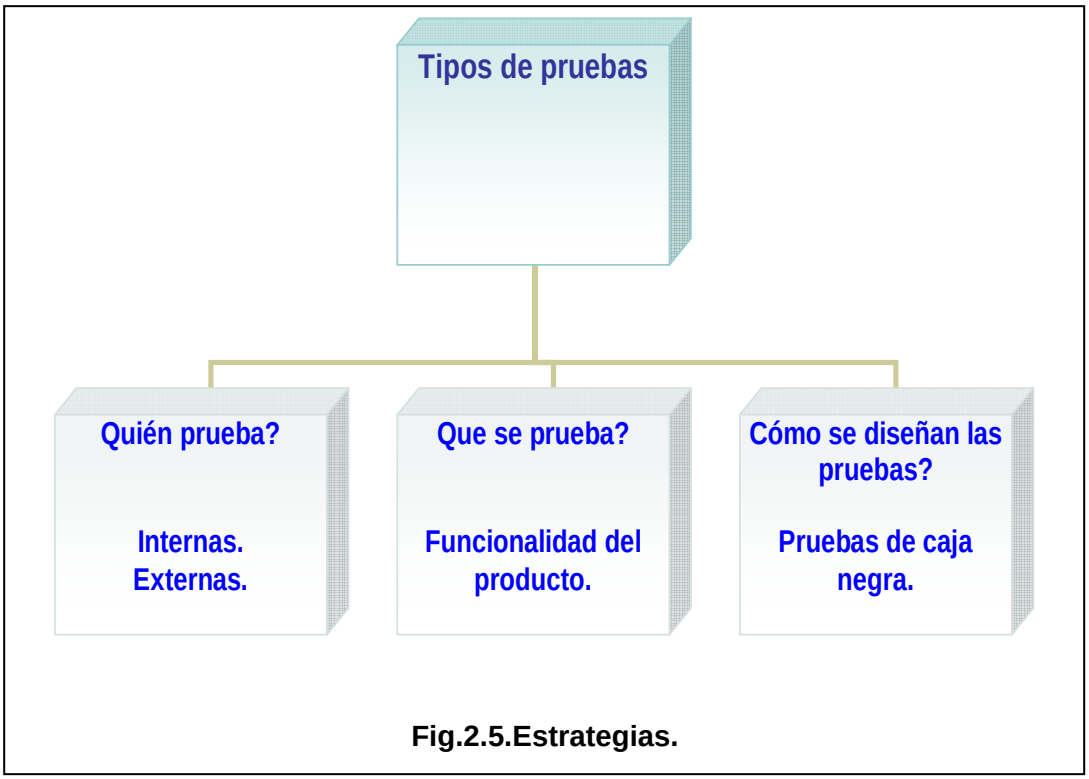

Fig.2.5.Estrategias.

- Dependiendo de quién sea el personal encargado de probar el módulo Generador de Modelos: se realizarán pruebas internas por la ingeniera de prueba perteneciente al equipo de calidad y externas por los clientes de las Oficinas de estadísticas ya sean Nacional, Provincial o municipal

- Dependiendo de cómo se van a diseñar las pruebas, se usará el diseño de casos de uso de prueba, *de caja negra,* tomando en cuanta la funcionalidad y los requisitos plantados por los clientes. Se realizarán las pruebas de validación al módulo Generador de Modelos. De las diferentes técnicas de prueba de caja negra se hizo un análisis de las existentes, de estas las seleccionadas fueron las de partición equivalente, la de análisis de valores limites, prueba de la documentación y facilidades de ayuda. Mediante el uso de los casos de prueba de la caja negra podemos demostrar que las funciones del software son operativas, la entrada se acepta de forma adecuada, se produce una salida correcta y además que la integridad de la información externa se mantiene, como podemos ver la prueba de caja negra se ajusta a los objetivos de este trabajo, lo cual se centraliza en la comprobación de la correspondencia de los requisitos planteado por los clientes.

2.3. Configuración del entorno de prueba.

El propósito de esta tarea es brindar una información que es de gran importancia ya que se describen los requerimientos de software y de hardware necesarios para realizar las pruebas con la mayor calidad posible. Para la realización de las pruebas escogidas para ser aplicadas al módulo Generador de Modelos, se necesitarán una máquina con los requerimientos mínimos siguientes como se muestra en la Fig. 2.5.

Software mínimos necesarios*:*

- Plataforma Windows.
- Visual Studio .NET 2005.
- Framework.NET.2.0

Hardwares mínimos necesarios:

- Microprocesador de 2.4 GHz como mínimo.
- Memoria RAM de 256 MB como mínimo soportado, aunque se recomienda + 512 MB.
- Capacidad de disco duro de 60 GB como mínimo.

- El procesador puede ser de la familia Intel Pentium/Celeron, AMD K6/Athlon/Duron u otro procesador compatible recomendado.
- PC.

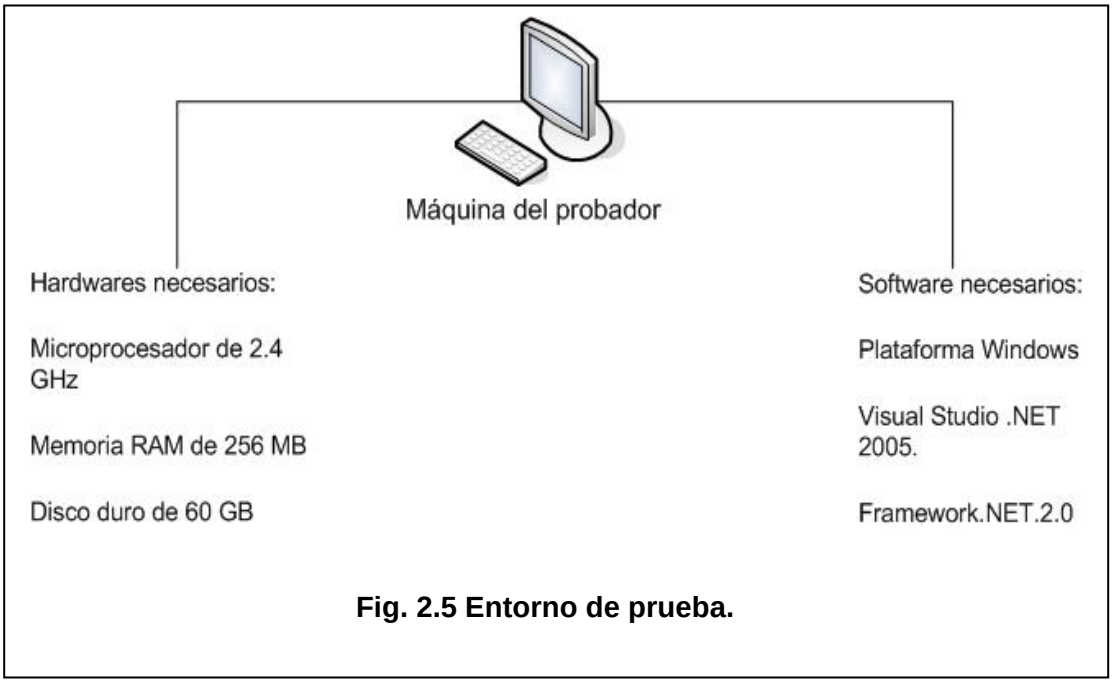

Fig. 2.5 Entorno de prueba.

2.4. Datos reales de prueba.

Para la realización de las pruebas al módulo Generador de Modelos se tuvieron en cuenta modelos del Sistema de Información de Estadística Nacional (SIEN), entre los que tenemos: M0006 ,el M0102, entre otros, para tener una correspondencia con lo que se realiza en las oficinas de estadísticas del país.

OFICINA NACIONAL DE ESTADISTICAS Sistema de Información Estadística Nacional (SIEN)	**PRODUCCIÓN DE PIEZAS DE REPUESTO**	**MODELO No.0102-01** Página 1 de 2
UNIDAD DE MEDIDA: MILES DE PESOS (MP) **ENTERO CON UN DECIMAL**	Informe acumulado hasta: Junio 30 ☐ Diciembre 31 ☐ Año:	**INFORME SEMESTRAL**
CENTRO INFORMANTE :	CÓDIGO CENTRO INFORMANTE:	

		FABRICACIÓN			**RECUPERACION**		
INDICADORES SELECCIONADOS	Código	PLAN	REAL	AÑO ANTERIOR	PLAN	REAL	AÑO ANTERIOR
A	B	1	2	3	4	5	6
FABRICACION Y/O RECUPERACION TOTAL	111						
MAQUINARIAS Y EQUIPOS DE LA INDUSTRIA AZUCARERA	112						
DE ELLO:							
COMBINADAS CAÑERAS	113						
ALZADORAS Y REMOLQUES DE ZAFRA	114						
MAQUINARIAS, EQUIPOS E IMPLEMENTOS AGROPECUARIOS	115						
DE ELLO:							
COMBINADAS DE ARROZ	116						
MAQUINARIAS Y EQUIPOS DE LOS SISTEMAS DE RIEGO	117						
TRACTORES Y BULLDOZERS AGRICOLAS	118						
TRANSPORTE AUTOMOTOR	119						
MAQUINARIAS Y EQUIPOS DE TRANSPORTE FERROVIARIO	120						
DE ELLO:							
LOCOMOTORAS	121						
EQUIPOS DE USO DOMESTICO	122						
DE ELLO:							
MOTOCOMPRESORES DE REFRIGERACION	123						
COCINAS	124						
MAQUINARIAS Y EQUIPOS DE LA INDUSTRIA ALIMENTICIA	125						
MAQUINARIAS Y EQUIPOS DE LA INDUSTRIA SIDERO-METALURGICA	126						
EQUIPOS PESADOS DE LA CONSTRUCCION Y LA MINERIA	127						
MAQUINARIAS Y EQUIPOS DE LA INDUSTRIA QUIMICA	128						
DE ELLO:							
MAQUINARIA Y EQUIPOS DE LA INDUSTRIA DEL:							
PLASTICO Y GOMA	129						
VIDRIO Y LA CERAMICA	130						
SUMA DE CONTROL (Página 1 de 2)	999						

Fig.2.6. Modelo0102, de talón cerrado.

OFICINA NACIONAL DE ESTADISTICAS		Informe acumulado hasta:	Modelo No. 0006-00 Página 1 de 2
	INDICADORES SELECCIONADOS	Mes:_____	MENSUAL
Sistema de Información Estadística Nacional (SIEN)		Año:_____	Unidad de medida: Entero con un decimal

CENTRO INFORMANTE:	CODIGO CENTRO INFORMANTE:

CONCEPTO	UNIDAD DE MEDIDA	CODIGO	AÑO ACTUAL		REAL AÑO ANTERIOR
			PLAN	REAL	
A	B	C	1	2	3

Fig.2.6. Modelo0006, de talón abierto.

2.5. Plan de prueba.

A continuación describiremos de forma general, como esta conformado un plan de prueba y en el capítulo siguiente lo haremos de manera específica para el módulo Generador de Modelos. El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas. Detalla quién es el responsable de cada una de las tareas previstas en el plan. Indica el tipo de prueba y las propiedades/elementos del software a ser probado. Identifica el cronograma de las tareas necesarias para preparar y ejecutar las pruebas, así como cualquier habilidad especial que se requiera. Un plan de pruebas deberá cumplir con ciertos puntos, en primer lugar se deberá tener en claro que tipo de pruebas se van a aplicar y su correcto orden y diseño. Diseñar correctamente las pruebas. Se analizarán los resultados de las pruebas mediante la cobertura de las mismas realizadas mediante las siguientes ecuaciones:

- Cobertura de pruebas (planeadas) =TP/ RfT
- Cobertura de pruebas (ejecutadas)= TX/ RfT
- Cobertura de pruebas (exitosas) = TS/ RfT

Notación

- Tp-pruebas planeadas.
- Tx-pruebas ejecutadas (tanto satisfactorias o no satisfactorias).
- Ts-pruebas exitosas (satisfactorias).
- RfT- requisitos de prueba.

CAPÍTULO 3

RESULTADOS OBTENIDOS.

3. Introducción.

En este capítulo presentamos los casos de uso de prueba, que le fueron aplicados al módulo generador de Modelos, como resultado de lo analizado en los capítulos anteriores, que fueron la base para la realización de los mismos, se mostrarán aspectos importantes del plan de pruebas, se hará un resumen de la evaluación de las pruebas, además de hacer un análisis de sobre la repercusión de los errores encontrados.

3.1. Diseño de casos de uso de prueba de caja negra.

A continuación mostramos los casos de usos de prueba de caja negra que se realizaron. Se utilizaron las técnicas de partición equivalente y análisis de valores límites. Para la elaboración de los mismos se utilizaron las descripciones de los casos de uso del sistema, ejecutados por el analista del módulo Generador de Modelos.

3.2. CU Prueba: Crear Modelos. Descripción de la Funcionalidad:

A través de este caso de uso el sistema nos da la posibilidad de crear un modelo, mediante su interfaz correspondiente.

3.2.1. Iteraciones. Caso de uso, descrito en el Anexo 5.

Clases válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 123456. Subnúmero: 897. Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto.		Se crea el modelo y se muestra un mensaje indicando que el modelo ha sido guardado satisfactoriamente.	Satisfactorio.

Día de captación: 15 Periodicidad: Semanal Nombre del archivo: Modelo1			
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 879512 Subnúmero: 456 Unidad de Medida: Miles de pesos. Descripción: Ingresos por ventas de bienes de consumo y servicios a la población Día de captación: 2 Periodicidad: Trimestral Nombre del archivo: Modelo1	El sistema da la posibilidad de cambiarle el nombre al archivo o reescribirlo.	Satisfactorio.
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 12345678999 Subnúmero: 7 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 18 Periodicidad: ocasional(mes de marzo)	El sistema muestra un mensaje de error indicando que se excedido la cantidad de dígitos, admitidos en el campo número.	Insatisfactorio.
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 5246 Subnúmero: 7456987278 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 21 Periodicidad: Mensual.	El sistema muestra un mensaje de error indicando que se excedido la cantidad de dígitos, admitidos en el campo subnúmero.	Insatisfactorio.

	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: vació Subnúmero:123 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 8 Periodicidad: Trimestral.	Es sistema muestra un mensaje indicando que existen campos obligatorios vacíos.	Satisfactorio.
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: q12 Subnúmero:789 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 9 Periodicidad: Anual	El sistema muestra un mensaje indicando que el campo número no se admiten letras ni signos.	Satisfactorio.
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 321578 Subnúmero:564 Unidad de Medida: Miles de pesos. Descripción: vació Día de captación: 5 Periodicidad: Mensual	El sistema muestra un mensaje indicando que existen campos obligatorios vacíos.	Insatisfactorio. Ver anexo **7.**
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 012365 Subnúmero:36 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 45 Periodicidad :Semestral y Anual	El sistema muestra un mensaje indicando que el día de captación que escribió no se encuentra dentro del rango de indicado.	Insatisfactorio. Ver anexo **8.**

	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 011478 Subnúmero:025 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: vació Periodicidad: Trimestral	El sistema indica que existen campos obligatorios vacíos.	Satisfactorio.
	Seleccione crear el modelo de talón cerrado/abierto. Introduzca los siguientes datos: Número: 356489 Subnúmero:312 Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 15 Periodicidad: vació	El sistema indica que no se ha seleccionado ninguna periodicidad.	Satisfactorio.

3.3. CU Prueba: Modificar Modelos. Descripción de la Funcionalidad:

A través de este caso de uso el sistema nos da la posibilidad de modificar un modelo, mediante su interfaz correspondiente.

3.3.1. Iteraciones. Caso de uso, descrito en el Anexo 5.

Clases válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
Seleccione abrir, se muestran los modelos de talón cerrado/abierto y modifique el número del modelo 1 Número: 123456. por 78956 Subnúmero: 897. Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 15 Periodicidad: Semanal Nombre del archivo: Modelo1		El sistema indica que la operación de modificar se realizo correctamente y que los datos están correctos.	Satisfactorio.
	Seleccione abrir, se muestran los modelos de talón cerrado/abierto y modifique el número del modelo 1 Número: 123456. por as344444 Subnúmero: 897. Unidad de Medida: Miles de pesos. Descripción: Producción de piezas de repuesto. Día de captación: 15 Periodicidad: Semanal Nombre del archivo: Modelo1	El sistema muestra un mensaje de error indicando, qué datos están incorrectos de los modificados.	Insatisfactorio.

3.4. CU Prueba: Eliminar Modelos. Descripción de la Funcionalidad:

A través de este caso de uso el sistema nos da la posibilidad de eliminar un modelo, mediante su interfaz correspondiente.

3.4.1. Iteraciones. Caso de uso, descrito en el Anexo 5.

Clases válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
Seleccione eliminar en el modelo que este trabajando. Seleccione eliminar el archivo: X, en cual esta trabajando		El sistema indica que el modelo se elimino correctamente.	Satisfactorio.
	Seleccione eliminar Seleccione eliminar y no hay ningún archivo abierto	El sistema muestra un mensaje de error, indicando que no hay modelo en el área de trabajo	Insatisfactorio.

3.5. Registro de defectos y dificultades encontradas. Caso de uso de prueba Gestionar Modelos.

Elemento de configuración	No	No conformidad	Aspecto correspondiente	Importancia	Recomendación
Interfaz	1	Cuando se va al menú Archivo/ /nuevo/Modelo de Estadística Continua /Talón Cerrado o / Talón Abierto, Se cierra la Aplicación.	Se encuentra en la Interfaz Crear modelos ,perteneciente al caso de uso Gestionar Modelos. Ver anexo **12.**	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	2	La Opción de ayuda, no esta disponible esta funcionalidad.	Se encuentra en la interfaz principal	Alta	Se debe de entregar en la segunda iteración
interfaz	3	Cuando se va a: Archivo/ /nuevo/Modelo de Estadística Parcial, no esta disponible esta funcionalidad.	Se encuentra en la interfaz principal	Alta	Debe estar disponible para la segunda iteración

Interfaz	5	No cuenta con autentificación, no esta disponible esta funcionalidad.	Debe aparecer en la interfaz principal	Alta	Debe estar disponible para la segunda iteración
Interfaz	6	Cuando se va a modificar el campo número por un dato incorrecto no muestra ningún mensaje de error	Se encuentra en la Interfaz Modificar Modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	7	En el campo de número de modelo, se admiten números de mayor de 6 cifras.	Se encuentra en la Interfaz de de crear modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	8	En el campo de subnúmero de modelo, se admiten números de mayor de 3 cifras.	Se encuentra en la Interfaz de de crear modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	9	Cuando se deja el campo de descripción vació, no se muestra ningún mensaje de error	Se encuentra en la Interfaz de de crear modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	10	En el campo día de captación se admiten números mayores de 31 y además cuando de selecciona el botón atrás para modificar alguna cosa, ha cambiado totalmente el día que se había escrito y sin mostrar ningún mensaje de error.	Se encuentra en la Interfaz de de crear modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	11	Cuando se va a modificar el campo número del modelo, por algún dato incorrecto ,no se muestra ningún mensaje de error	Se encuentra en la Interfaz Modificar Modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba
Interfaz	12	Cuando se va a eliminar un modelo sin que exista algún modelo en creación, la aplicación da un error y no permite hacer ninguna otra operación.	Se encuentra en la Interfaz eliminar modelos, perteneciente al caso de uso Gestionar Modelos.	Alta	Se debe de arreglar para la segunda iteración

3.6. CU Prueba: Crear nomenclatura. Descripción de la Funcionalidad:

A través de este caso de uso el sistema nos da la posibilidad de crear un modelo, mediante su interfaz correspondiente.

3.6.1. Iteraciones. Caso de uso, descrito en el Anexo 6.

Clases válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
	Seleccione crear el modelo de talón cerrado, y creé la nomenclatura del mismo Introduzca los siguientes datos: Tipo de indicador: maquinarias y equipos de la industria azucarera Código: 112 Indicador: combinadas cañeras Código: 113 alzadoras y remolques de zafra Código:113 Unidad de medida: Miles de p.	El sistema muestra un mensaje de error indicando que no se pueden tener dos indicadores con el mismo código.	Insatisfactorio. Ver anexo **9.**
Seleccione crear el modelo de talón cerrado, y creé la nomenclatura del mismo Introduzca los siguientes datos: Tipo de indicador: maquinarias y equipos de la industria azucarera Código: 112 Indicador: combinadas cañeras Código: 114 alzadoras y remolques de zafra Código:115 Unidad de medida: Miles de p.		El sistema muestra un mensaje indicando que se la operación de creación de nomenclatura se realizó correctamente.	Satisfactorio.
	Seleccione crear el modelo de talón cerrado, y creé la nomenclatura del mismo Introduzca los siguientes datos: Tipo de indicador: maquinarias	El sistema muestra un mensaje de error , indicando que en el código del tipo de indicador, solo se admiten hasta 6 dígitos	Insatisfactorio. Ver anexo **10.**

	y equipos de la industria azucarera Código: 112333333 Indicador: combinadas cañeras Código: 114 alzadoras y remolques de zafra Código:115 Unidad de medida: Miles de p.		
	Seleccione crear el modelo de talón cerrado, y creé la nomenclatura del mismo Introduzca los siguientes datos: Tipo de indicador: maquinarias y equipos de la industria azucarera Código: 112sgsysy Indicador: combinadas cañeras Código: 11456 alzadoras y remolques de zafra Código:11578 Unidad de medida: Miles de p.	El sistema muestra un mensaje indicando que en el código del tipo de indicador no se admiten, letras ni signos.	Insatisfactorio. Ver anexo **11.**

3.7. CU Prueba: Actualizar nomenclatura. Descripción de la Funcionalidad:

EL sistema nos da la posibilidad de actualizar la nomenclatura, mediante su interfaz correspondiente.

3.7.2. Iteraciones. Caso de uso, descritos en el Anexo 6.

Clases válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
Seleccione la opción de abrir y escoja de los modelos registrados, el que se le va a cambiar la nomenclatura y cambie Tipo de indicador: maquinarias y equipos de la industria azucarera Código: 11256 a 012365 Indicador: combinadas cañeras Código: 11456 alzadoras y remolques de zafra Código:11578 Unidad de medida: Miles de p.		El sistema verifica los datos y muestra un mensaje indicando que la nomenclatura ha sido actualizada	Satisfactorio
	Seleccione la opción de abrir y escoja de los modelos registrados, el que se le va a cambiar la nomenclatura y cambie Tipo de indicador: maquinarias y equipos de la industria azucarera Código: 11256 a asjsjj123 Indicador: combinadas cañeras Código: 11456 alzadoras y remolques de zafra Código:11578 Unidad de medida: Miles de p.	El sistema muestra un mensaje de error indicando que la nomenclatura no pudo ser modificada, porque los datos están incorrectos	Insatisfactorio.

3.8. CU Prueba: Eliminar Nomenclatura. Descripción de la Funcionalidad:

A través de este caso de uso el sistema nos da la posibilidad de eliminar la nomenclatura seleccionada mediante su interfaz correspondiente.

3.8.1. Iteraciones. Caso de uso, descrito en el Anexo 6.

Clases válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
Seleccione la opción de eliminar en la nomenclatura del modelo en el que este trabajando y elimine. Tipo de indicador: maquinarias y equipos de la industria azucarera Código: 11256 Indicador: combinadas cañeras---eliminar Código: 11456 alzadoras y remolques de zafra Código:11578 Unidad de medida: Miles de p.		El sistema muestra un mensaje indicando que la nomenclatura ha sido eliminada satisfactoriamente.	Satisfactorio
	Seleccione la opción de eliminar en la nomenclatura del modelo en el que este trabajando, este modelo no cuenta con nomenclatura creada.	El sistema muestra un mensaje de error indicando que no existe nomenclatura en el modelo.	Satisfactorio.

3.9. Registro de defectos y dificultades encontradas. Caso de uso de prueba Gestionar Nomenclatura.

Elemento de configuración	No	No conformidad	Aspecto correspondiente	Importancia	Recomendación
Interfaz	1	Cuando se crean dos indicadores con el mismo código, no se muestra ningún mensaje de error	Se encuentra en la interfaz Crear Nomenclatura, perteneciente a caso de uso Gestionar Nomenclatura.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	2	Cuando se introducen en el código del indicador números mayores de 6 cifras, no se muestra ningún mensaje de error.	Se encuentra en la interfaz Crear Nomenclatura, perteneciente a caso de uso Gestionar Nomenclatura.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	3	Cuando se introducen en el código del indicador letras y signos , no se muestra ningún mensaje de error.	Se encuentra en la interfaz Crear Nomenclatura, perteneciente a caso de uso Gestionar Nomenclatura.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.
Interfaz	4	Cuando se va actualizar algún dato de la nomenclatura y se introduce algún dato incorrecto, no se muestra ningún mensaje de error.	Se encuentra en la interfaz Actualizar Nomenclatura, perteneciente a caso de uso Gestionar Nomenclatura.	Alta	Debe solucionarse para la próxima revisión del ingeniero de prueba.

3.10. Pruebas a la documentación.

Se realizaron pruebas a la documentación del módulo Generador de Modelos, tomando entre sus documentos más significativos el Manual de usuarios, siendo este unos de los aspectos de mayor importancia a la hora de la entrega de un producto a los clientes, se tomaron en cuenta una serie de aspectos para la revisión del mismo entre los que tenemos:

1 Legibilidad del contenido.
2 Faltas de ortografía.
3 Forma de instalación de la aplicación.
4 Requisitos de hardware y software.
5 Mensajes de error.
6 Acciones a tomar enumerados en caso de que ocurra un error.
7 Imágenes ilustrativas de los pasos.

Ante lo antes expuesto podemos indicar que se detectaron dificultades en dicho manual, señalando que solo se hace una descripción muy superficial de cómo se realizan las actividades de crear, modificar, eliminar y actualizar modelos y nomenclaturas y sus respectivas imágenes, aunque si hay que destacar que todo esta muy bien explicado y sin faltas de ortografía, en un lenguaje técnico, pero a la vez entendible para los usuarios de la aplicación, pero no se cuentan con aspectos importantes dentro del mismo, como lo son los requisitos de hardware y software, que son de necesario conocimiento para el buen funcionamiento del producto , además de que no se explican los posibles errores que se le pueden presentar y las posibles soluciones a los mismo, tampoco se expone la forma de instalación, que a pesar de parecer una simpleza realmente es de gran utilidad ya que no todos los que van a trabajar con la aplicación son informáticos, ni especialistas en el tema. Los errores en la documentación pueden llegar a ser tan perjudiciales para la aprobación del producto, como los errores en los datos o en el código fuente, para los usuarios debe ser algo frustrante, seguir paso por paso el manual de usuario y obtener resultados o comportamientos contrarios a lo adelantado en el documento. Por eso es de vital importancia tener presente la prueba documentación, ya que es un elemento clave para la aceptación del programa, por parte de los usuarios.

3.11. Plan de prueba del módulo Generador de Modelos.

En este plan de prueba mostraremos una lista que recoge todos los requerimientos funcionales, para los casos de uso Gestionar Modelos y Gestionar Nomenclatura del módulo Generador de Modelos, terminados en la primera iteración del producto. Cualquier requerimiento que no se haya contenido en esta lista estará fuera del alcance de las pruebas. Esto nos libera de responsabilidad en caso de que existan problemas con la funcionalidad del sistema que se relacione con un requisito no incluido en este listado. El encargado de toda la realización del plan de prueba fue el ingeniero de prueba de este módulo.

Requisitos Funcionales.

A continuación se muestran los requisitos funcionales que se tuvieron en cuenta para la realización de los casos de uso de pruebas, ejecutados al módulo Generador de Modelos, verificando la funcionalidad de los mismos de acuerdo a lo planteado por los clientes de la Oficina Nacional de Estadísticas .

Caso de uso Gestionar Modelos.

1 El sistema debe dar la posibilidad de Crear los Modelos.
2 El sistema debe dar la posibilidad de Modificar los Modelos.
3 El sistema debe dar la posibilidad de Eliminar los Modelos.

Caso de uso Gestionar Nomenclatura.

1 El sistema debe dar la posibilidad de Crear la Nomenclatura de un modelo determinado.
2 El sistema debe dar la posibilidad de Modificar la Nomenclatura de un modelo determinado.
3 El sistema permite eliminar la nomenclatura de un modelo determinado.

3.11.1. Análisis de los resultados de las pruebas.

En la Fig.3.1, se muestra el por ciento que representan cada tipo de los errores encontrados mediante la aplicación de las pruebas, en sus distintas clasificaciones, según la afectación que causaban a la aplicación, del módulo Generador de Modelos.

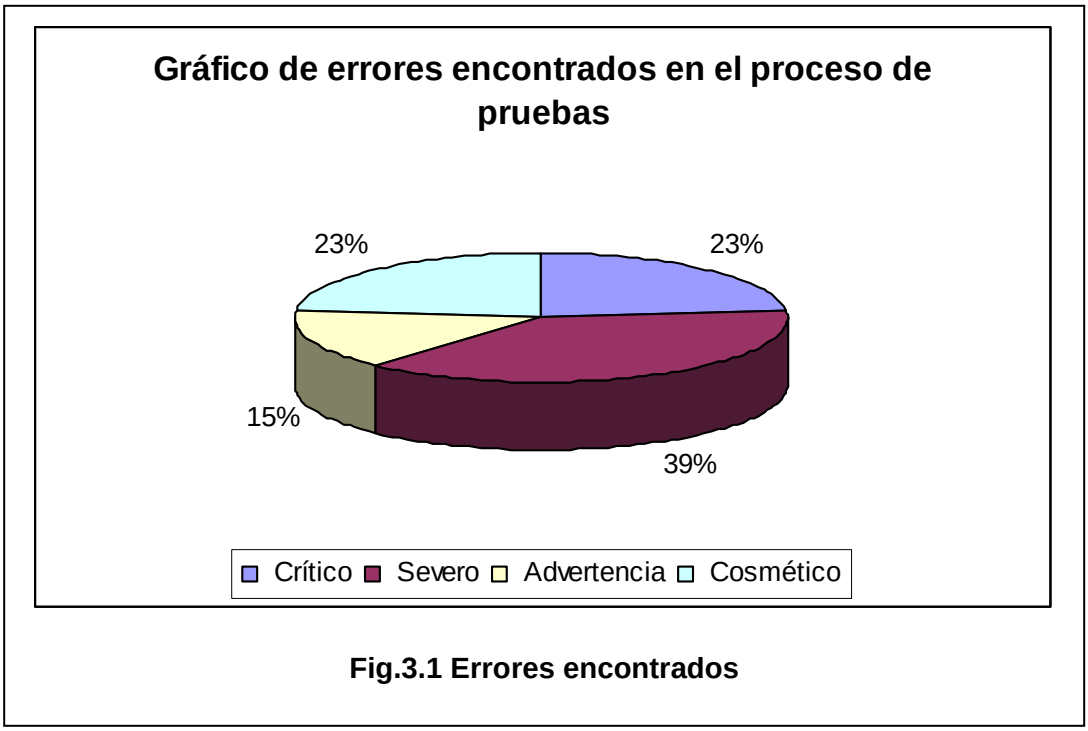

Fig.3.1 Errores encontrados

Las siguientes son categorías, para priorizar o calificar los defectos encontrados :

- **Crítico**: Denota una función inutilizable que causa un término anormal o un fallo general.
- **Severo**: Una función no actúa como fue requerida o diseñada, o un objeto de interfaz no trabaja como se muestra.
- **Advertencia**: La función trabaja, pero no tan rápidamente como se espera.
- **Cosmético**: No es un error crítico para el funcionamiento de sistema: palabras con mala ortografía y mensajes de error o advertencias, confusos.

3.11.2 Cobertura de las pruebas.

Se planificaron 6 casos de uso de pruebas en correspondencia con los casos de uso Gestionar Modelos y Gestionar Nomenclaturas, todos fueron ejecutados y de estos se logro un 16 % de pruebas exitosas como se muestra en la Fig.3.2. Podemos concluir, que se probaron todos los casos de uso, logrando verificar todos los requisitos planteados por los clientes ,se encontraron errores que se fueron informando a los desarrolladores, a medida que se realizaba el proceso de aplicación de pruebas, para su

posterior corrección y así lograr que la entrega a los usuarios se realizará con la menor cantidad de dificultades, esto queda demostrado en las cartas de Avales que fueron presentadas por los compañeros de las oficinas de estadísticas de todas las provincias del país , de estas se tomaron como muestra las representadas en los Anexos **13, 14, 15, 16,** donde ellos manifiestan la aprobación del producto.

Cobertura de pruebas (planeadas) = 6/ 6=1*100=100%

Cobertura de pruebas (ejecutadas)= 66=1*100=100%

Cobertura de pruebas (exitosas) = 1/6=0.16*100=16%

Notación

RfT- requisitos de prueba=6

Tp-pruebas planeadas=6

Tx-pruebas ejecutadas (satisfactorias o insatisfactorias)=6

Ts-pruebas exitosas (satisfactorias)=6

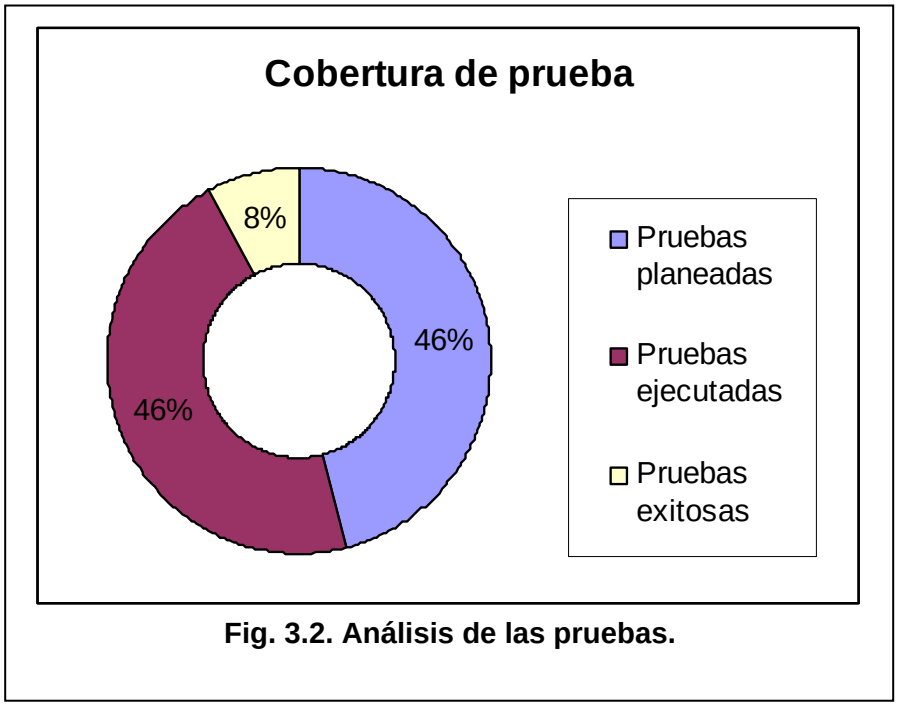

Fig. 3.2. Análisis de las pruebas.

3.12. Resumen de evaluación de las pruebas.

Las pruebas fueron realizadas al módulo Generador de Modelos el cual tiene el objetivo de automatizar el proceso relacionado con la gestión de modelos, bases metodológicas, cuadres y nomenclaturas correspondientes, a las oficinas estadísticas. La realización de las pruebas de calidad al software antes referido son indiscutiblemente la mejor forma de conocer el grado con que se manifiesta este indicador antes de ponerlo en manos de los clientes. En este caso estas pruebas fueron de gran utilidad debido a que arrojaron como resultado algunos errores que no se habían detectado por los desarrolladores y que era necesario eliminar debido a que fueron requisitos funcionales declarados por los clientes. Errores tales como:

- Problemas a la hora de almacenar el “día de captación”.
- Problemas con la cantidad de dígitos aceptados por algunos campos como Número del modelo, Subnúmero, entre otros.
- El campo Descripción del modelo no estaba valorado como obligatorio cuando si lo era.
- En el caso de los indicadores existía el problema de que el software no verificaba si existían dos indicadores con el mismo número, entre otros.

Como se demuestra la realización de las pruebas fueron un eslabón importante para que la entrega de la primera versión estuviera en correspondencia con los requisitos planteados por los clientes. Las pruebas fueron realizadas en tres etapas en las que fueron detectando errores y a la vez informando al equipo de desarrolladores para su corrección y luego realizarse la próxima revisión, para ir refinando la aplicación .Como se puede observar en la Fig. 3.3, a medida que se avanzaba en las etapas del proceso de prueba se logró una disminución bastante notable de los errores, obteniéndose en la primera etapa un 54 % de errores, en la segunda un 32 %, mientras que en la tercera un 14 % de errores.

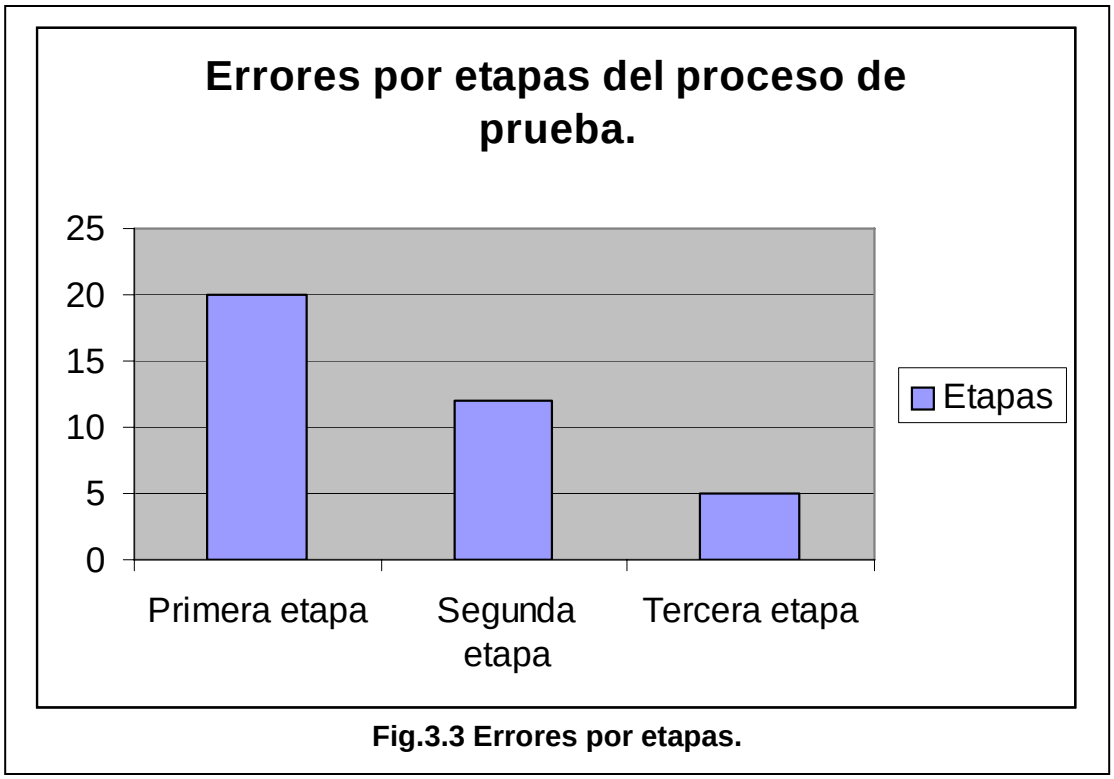

Fig.3.3 Errores por etapas.

3.13. Propuesta de herramienta de automatización de prueba.

Luego de hacer un estudio del arte de las diferentes herramientas para automatización de las pruebas existentes en la actualidad y definir las que se ajustaban a las necesidades y características de nuestro producto de Gestión Estadísticas, como una aplicación de Escritorio y muchos otros aspectos que se tuvieron en cuenta, para realizar dicha fundamentación nos basamos en la Ayuda del Enterprise Arquitect. Se propone la integración de las herramientas CASE , Enterprise Arquitect(EA), con la herramienta de prueba NUnit por las razones que a continuación plantearemos.

Enterprise Architect 6.1, es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, provee la capacidad de compilar, *probar*, depurar y ejecutar scripts de despliegue - todo dentro del entorno de desarrollo de EA. Suministra a los desarrolladores con herramientas para integrar sus desarrollos y modelados UML (Unified Modeling Language) con su desarrollo fuente y compilación. Con la capacidad de generar clases de ***prueba*** NUnit e integrar los procesos de pruebas directamente en la interfaz de EA, es posible ahora integrar el UML y modelado en el proceso de compilar/probar/ejecutar/desplegar.

3.14. Validación de especialistas. Método Delphi.

En los objetivos de la investigación, nos vemos obligados a realizar una validación externa del resultado alcanzado, por lo que solicitamos de cada experto que nos responda la siguiente interrogante. ¿Considera usted, que las técnicas utilizadas en la aplicación de las pruebas al módulo Generador de Modelos , fueron las correctas?

Selección de los expertos que trabajaron en la investigación.

Indicadores para la primera selección:

- Años de experiencia: Más de 5 años como Ingeniero Informático.
- Años trabajando en el campo de la informática: Más de 5 años
- Calificación: Ingeniero Informático.
- Criterio de evaluación.

Se tomaron como posibles especialistas (5), aquellos profesionales que por más de cinco años habían estado trabajando en el área de la informática en diferentes proyectos de investigación o productivos, se realizaron talleres, donde se analizaron las características de la aplicación de las pruebas, caja negra y caja blanca, su estructura, fundamentos para su aplicación, observación y detección de dificultades.

Estas actividades se realizaron de forma teórica y práctica, la primera selección (4) se efectúa basada en un criterio de evaluación de los conocimientos adquiridos (más del 80% de los conocimientos), de estos fueron escogidos 3 bajo el criterio de evaluación en las habilidades para la aplicación de las pruebas; los que posteriormente se perfeccionaron en el trabajo de la aplicación de estas técnicas, teniendo en cuenta el diseño para la investigación.

Caracterización de los expertos que trabajaron en la investigación.

- Edad promedio: 30.7 años
- Experiencia como Ingeniero Informáticos: 10.8 años
- Calificación: Ingeniero Informáticos.

CONCLUSIONES.

Dada la necesidad de que los productos creados sean confiables y precisos, crece la exigencia por parte de clientes y usuarios en general, de que a los sistemas se les hayan aplicado las técnicas de Ingeniería de Software adecuadas y en su etapa de comprobación hayan sido probados con las técnicas necesarias para lograr el nivel de calidad requerido**.** Al aplicarse las pruebas al módulo Generador de Modelos, se cumplieron todos los objetivos planteados para esta investigación, llegándose a las siguientes conclusiones:

- Se verificó la funcionalidad de los requisitos planteados por los clientes de la Oficina Nacional de Estadísticas, mediante la aplicación de la estrategia seleccionada, basada en técnicas de prueba de caja negra, como son: partición equivalente, análisis de valores límites y además las pruebas a la documentación, aplicadas al módulo Generador de Modelos, perteneciente al Sistema Integrado de Gestión Estadística.

- Se detectaron errores significativos, que no fueron descubiertos por los programadores del producto, quedando probado el cumplimiento de los requisitos funcionales y el adecuado desempeño de los mismos, mediante la aplicación de casos de uso de prueba de caja negra, (al módulo Generador de Modelos).

- Se logró una mayor organización y planificación del proceso de prueba, efectuado al módulo Generador de Modelos, lo cual se obtuvo a través de la elaboración del Plan de Pruebas.

- La aplicación de la pruebas de caja negra al módulo Generador de Modelos, nos permitió ultimar como los errores encontrados representan el 72.2% del total de las mismas, siendo este aspecto importante, lográndose obtener un producto fiable, alcanzando cumplir con las expectativas de los clientes, entregándoseles un producto con un alto índice de correspondencia con lo planteado en sus requisitos funcionales.

RECOMENDACIONES.

Luego de ejecutar el proceso de prueba y conocer las posibilidades que ofrece a los desarrolladores de los productos de software, una adecuada detección de errores, en el instante adecuado, así como la significación y utilidad que representa la aplicación de nuestra propuesta, se recomienda:

- La integración de la herramienta CASE Enterprise Architect, con la herramienta de prueba NUnit, para la automatización de las pruebas unitarias, por sus numerosas ventajas entre las que tenemos: capacidad de generar clases de ***prueba*** NUnit e integrar los procesos de pruebas directamente en la interfaz de EA, para facilitar el trabajo de los desarrolladores y demás equipo de trabajo.

- Aplicar las pruebas al sistema en próximas iteraciones.

- Realizar pruebas de caja blanca, entre otras

BIBLIOGRAFÍA.

PERRY, D. E. A. G. E. K. *Adequate testing and object oriented programming*. , 1990.

CORTÉS, O. H. G. *Aplicación práctica del diseño de pruebas de software a nivel de programación*, 2004. Disponible en: http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf

BORIS, B. *Black box testing,* 1995.

BLACK, R. *Critical Testing Process,* Julio 29, 2003.

FLEITMAN, J. *Cómo Instalar Y Evaluar Un Modelo De Calidad.* 2004. Disponible en: http://redgestion.fundacionchile.cl/documents/files/Evaluaci%C3%B3n%20integral%20hacia%20modelos%20de%20calidad.doc

FERNANDEZ., G. *Del Test Driven Development (TDD) y utilización de NUnit*. Disponible en: http://www.willydev.net/descargas/TDD.pdf

CHEN, M. H., AND H.M. KAO. *Effect of class testing on the reliability of object oriented programs.*, 1997.

WU, K. L. A. M. *Effective GUI Test Automation: Developing an Automated Testing Tool*, 2005. Disponible en:

DUSTIN., E. *Effective Software Testing.*

CARVAJAL, L. G. A.-P. H. *Eficiencia de la caracterización de técnicas de prueba en un contexto real de aplicación:*

Parquesoft. Disponible en: http://www.greensqa.com/archivos/OptimizacionPruebasSoftware.pdf

IVAR JACOBSON , G. B., JAMESRUMBAUGH. *El Proceso Unificado de desarrollo De Software* p.

BADAL, M. *Elaboración de referencias y citas según las normas de la American Psychological Association (APA), 5ª Edición*. Disponible en: http://www.monografias.com/apa.shtml

Enterprise Architect - Herramienta de diseño UML. 2007. [Disponible en: http://www.sparxsystems.com.ar/products/ea.html

ARENAS, L. G. *Estrategia de Calidad - Parquesoft.* Disponible en: http://www.greensqa.com/archivos/GSQA-Estrategia%20Calidad%20ParqueSoft.pdf

ANNA. C GRIMÁN, M. P., LUIS. E MENDOZA. *Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales*. Disponible en: http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_09.pdf

SCALONE., L. F. *Estudio comparativo de los modelos y estándares de calidad del software"*. Facultad Regional Buenos Aires, Universidad Tecnológica Nacional, 2006. p.

QUESADA., J. A. L. *Fundamentos de Ingeniería del Software .Prueba de Software*, 2006-2007. [Disponible en: http://dis.um.es/~lopezquesada/documentos/FIS_0607/curso/Tema5.pdf

Herramientas para el entorno de desarrollo. 2007. Disponible en: http://www.als-es.com/home.php?location=herramientas/entorno-desarrollo

Herramientas para el entorno de pruebas. 2007. Disponible en:http://www.als-

es.com/home.php?location=herramientas/entorno-pruebas

YAMAURA, T. *How to design practical test cases*, 1998.

PRESSMAN, R. S. *Ingeniería del Software .Un enfoque práctico.* Quinta Edición. 2005. p.

S.HUMPHREY., W. *Introducción al Proceso Software Personal. PSP*. p.

LUZURIAGA, J. M. *Inspecciones de software*. Disponible en: http://www.monografias.com/trabajos6/isof/isof.shtml

IRINA BRITO REYES , I. N. T. *Las pruebas de software, su aplicación al Config. Case.*, 2003. p.

FORSYTH, P. D. *Lo Nuevo en Herramientas de Exchange 2007*2007. Disponible en: http://www.microsoft.com/latam/technet/prodtechnol/exchange/articles/toolsupdatese12.mspx

USAOLA, M. P. *Mantenimiento Avanzado de Sistemas de Información.* Disponible en: http://alarcos.inf-cr.uclm.es/doc/masi/doc/lec/parte5/polo-apuntesp5.pdf

PIQUERA., T. G. H. Y. L. A. V. *Manual de usuario del Microset NT.*, 1997.

ARENAS, L. D. S. G. *Metodología para evaluar la calidad de los sistemas de información.*, Cali, Octubre de 2004. Disponible en: http://www.greensqa.com/archivos/Metodologia%20BaQEM.pdf

HERNÁNDEZ, M. A. C. *Modelo para Pruebas de Software y auditoria en Entorno Microsoft.Net*. Disponible en: http://www.monografias.com/trabajos20/pruebas-de-software/pruebas-de-software.shtml

PATIÑO G., C. A. *Modelos de calidad en la formación profesional y en la educación. Análisis y complementariedad*, Montevideo: CINTERFOR/OIT, 2006

ALARCÓN, A. S. *Modelos de calidad. La industria del software en México*, Enero de 2004. Disponible en: http://www.enterate.unam.mx/Articulos/2004/Enero/modelos.htm

NUnit., 2006. Disponible en: http://www.nunit.org/index.php?p=home

VERMA, A. *Optimización de estrategias de prueba durante el diseño*. Disponible en: http://www.teradyne.com/atd/resource/docs/spectrum/Verma_electronica_Feb2003.pdf

MAÑAS, J. A. *Prueba de Programas*, 1994. Disponible en: http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s7

ACUÑA., C. J. *Pruebas del Software.* Disponible en: http://kybele.escet.urjc.es/documentos/ISG/%5BISG-2006-07%5DPruebasSoftware.pdf

MARIO L. GUERINI, E. F., ALEJANDRA OCHOA, HERNÁN MERLINO, EDUARDO DIEZ, PAOLA BRITOS Y RAMÓN GARCÍA-MARTÍNEZ. *Revisión de Resultados Experimentales sobre. Performance de Técnicas Pruebas de Software*. Disponible en: http://www.itba.edu.ar/capis/webcapis/RGMITBA/comunicacionesrgm/CACIC-2006-Articulo-692.pdf

AGUSTÍN CERNUDA DEL RÍO, P. L. O. D. T. D. D. E. I. *Sistema de verificación de componentes software*, Universidad de Oviedo. Departamento de Informática, Febrero de 2002. p.

Software SPSS., 2006. Disponible en: http://www.estadistico.com/soft.html?act=pak&id=mp

MARNIE L. HUTCHESON, J. W. *Software Testing Fundamentals: Methods and Metrics*, 2003

BEIZER. *Software Testing Techniques.*, 1990.

SPSS Base 15.0*: El software estadístico integral., 2006.* Disponible en: http://www.spss.com/la/productos/base/base.pdf

GERMÁN ARBERT, P. F., HÉCTOR I. MORÁN H. *Técnicas de Pruebas de software.* Disponible en: https://weblogs.udp.cl/hmoran/archivos/(526)Efectibidad_de_pruebas(hm).ppt#256,1,Técnicas

JERRY ZEYU GAO, H.-S. J. T. A. Y. W*. Testing and Quality Assurance for Component-Based Software,* 2003.

GOEL, B. I. A. L. *Testing object-oriented software".* 2000.

GLOSARIO DE TÉRMINOS.

CASE: Computer Aided Software Engineering. Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas informáticos, completamente o en alguna de sus fases.

CU: Caso de uso.

DESARROLLADOR: Toda aquella persona que se ocupa directamente en el desarrollo de un proyecto de software, por ejemplo : analista, programador, diseñador, arquitecto, etc.

EA: Herramienta CASE Enterprise Architect.

MED: Módulo de Entra de Datos.

MGM: Módulo Generador de Modelos.

MRC: Módulo de Registros y Clasificadores.

NUnit: Herramienta de prueba Nunit.

ONE: Oficina Nacional de Estadísticas.

RUP: Proceso Unificado del Software.

R1,R2,R3: Requisitos funcionales.

SIEN: Sistema Informativo de Estadística Nacional.

SIGE: Sistema Integrado de Gestión Estadísticas.

SPSS: Software estadístico integral.