UNIVERSIDAD CATOLICA ANDRES BELLO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA INFORMATICA

ESTRATEGIAS DE PRUEBA PARA PROPICIAR REQUERIMIENTOS NO FUNCIONALES QUE GARANTICEN LA CALIDAD DEL SOFTWARE OO

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: Diecinueve (19)

JURADO EXAMINADOR

Firma:		Firma:		Firma:	
Nombre:	Anna Grimán	Nombre:	SUSANA GARCIA M.	Nombre:	CARLO MAGUARO

REALIZADO POR: Clemente Tauil, Carol
Fernández Piñango, Norma Karina

PROFESOR GUIA: Grimán, Anna Cecilia

FECHA: Caracas, 17 de Julio del 2003

A Dios por ser mi eterno amigo, guiarme en el camino de mi vida y permitirme alcanzar este logro.

A mis padres por todo su amor, dedicación y ejemplo de constancia porque nunca dudaron que iba a lograr alcanzar mi meta de ser Ingeniero.

A mi abuela Lola por su gran amor y dedicación.

A mis hermanas Karla y Karen, quienes han sido mi orgullo y ejemplo a seguir por su fuerza y constancia para alcanzar sus logros. Por todo su amor, apoyo y por estar siempre presente en todos los actos de mi vida les dedico este logro.

A ti Karina por ser mi amiga incondicional con la que he compartido muchos momentos inolvidables de mi vida. Hemos culminado este camino juntas para seguir acompañándonos y compartiendo todos los que vienen.

A todos mis familiares y amigos que de una u otra forma han contribuido para alcanzar esta gran meta en mi vida.

Carol

A Dios, por guiarme en todos los caminos de mi vida.

A mi mamá y mi papá, por sus esfuerzos, su gran amor, por los valores que me han enseñando y por el incondicional apoyo que en todo momento me brindan y me permite seguir adelante. Este logro es gracias a ustedes.

A mi tía Zoraida, por ayudarme y aconsejarme en todo momento.

A mi abuelita por sus valiosas enseñanzas y por todo el amor que siempre me brinda.

A mi gran amiga Carol por brindarme siempre su amistad sincera y estar siempre a mi lado en las buenas y malas. Sólo juntas pudimos llegar hasta aquí.

A mis familiares y amigos, en especial a Eloy, que de una u otra forma siempre estuvieron conmigo y me ayudaron a alcanzar esta gran meta.

Karina

Agradecimientos

Queremos agradecer a Dios por estar siempre presente y ser nuestro guía en todos los momentos y acciones de nuestras vidas.

Queremos agradecer a muchas personas por su apoyo para que fuera posible la culminación exitosa de este trabajo.

A la profesora Anna Cecilia Griman, que por su labor como tutor académico, nos guió a encaminar este trabajo hacia una buena culminación.

A la profesora M Angélica Ovalles por su valiosa colaboración en la elaboración de este trabajo.

A todo el equipo de Sistemas de QUIMBIOTEC: Arilds Abollins, Andrea Mora, Armando Siems, Wilfredo Báez y Melesio Arias, por su colaboración y apoyo brindado durante nuestra permanencia en la Empresa.

A nuestros Padres, por el apoyo que nos brindaron, por la formación, por fomentar en nosotras el deseo de saber y abrirnos las puertas a un mundo lleno de triunfos.

A nuestros familiares por aconsejarnos y estar en los buenos y malos momentos que pasamos durante la carrera.

A nuestros amigos por estar presentes y contar con ellos siempre. En especial a todos los del CEI por esos gratos e inolvidables momentos. Y a cada una de las personas que de una u otra forma han colaborado en la culminación de este trabajo.

A todos Gracias.

Carol y Karina

Índice general

Índice de Apéndices y Anexos

Apéndices

Anexos

Índice de tablas

Índice de figuras

Resumen

Propiciar Calidad en el Software es una actividad que ha surgido como consecuencia de la fuerte demanda de software en todos los procesos que se desarrollan en la actualidad; desde programas elementales de contabilidad hasta programas complejos como los espaciales. De ahí el gran esfuerzo que se ha desplegado para obtener software de alta calidad.

Uno de los factores determinantes en un buen proceso de Control de Calidad en software es el relativo a la **Prueba.** Sin embargo, el proceso de prueba, generalmente, no se implementa de forma organizada y sistemática, y mucho menos cuando se trata de evaluar los **Requerimientos No Funcionales**.

El objetivo del presente trabajo de investigación es presentar una alternativa de **Prueba de Software Orientado a Objetos (OO)** evaluando los **Requerimientos No Funcionales** a través de una **Estrategia de Prueba**; para lo cual se utilizó la metodología de **Investigación Acción**, que permite realizarla de manera planificada. Esta Estrategia de Prueba incluye la generación y ejecución de un método de evaluación, representado por listas de verificación y casos de prueba, que permiten determinar el nivel de presencia en un Software, de cinco (5) **Requerimientos No Funcionales**: Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad.

El trabajo de investigación concluye con la aplicación de la Estrategia de Prueba a un caso de estudio real, representado por el **Sistema de Comercialización** de la Empresa **QUIMBIOTEC**. El resultado de la aplicación de la Estrategia de Prueba fue exitoso, en el sentido de que cumple con el objetivo principal de las Pruebas de encontrar fallas en el software, y demostró su potencial de aplicabilidad en Software OO.

Introducción

El proceso de globalización que se vive en la actualidad, aunado a la alta competencia, exige que las empresas desarrollen productos de **calidad** con el fin de mantenerse competitivas. Proyectando esto a la "industria del software", surge la necesidad de incorporar el enfoque de **calidad** como una condición vital al desarrollar sistemas de software. Los usuarios verán una mayor calidad en un producto de software en la medida que éste responda a los requerimientos, desarrolle un buen rendimiento, tenga facilidad de uso, presente una real ayuda y la documentación final para el usuario sea realmente útil.

Actualmente, los Sistemas de Información de la mayoría de las compañías están creciendo en complejidad. De hecho, los nuevos paradigmas en el desarrollo de software y las nuevas tecnologías son cada vez más poderosas y permiten alcanzar un mayor grado de flexibilidad, pero por otro lado, esto se traduce en soluciones más complejas para los sistemas de software finales y su mantenibilidad se convierte en una tarea muy cambiante. Existen Empresas donde los sistemas entregados a los clientes presentan 15% de defectos residuales (Reo, 2002). En este escenario, la realización de **Pruebas** durante todo el proceso de desarrollo ha llegado a ser una actividad muy exigida.

El software debe ser probado para asegurar su correcto funcionamiento. La buena planificación de una **Estrategia de Prueba** puede reducir significativamente el esfuerzo requerido para el desarrollo de pruebas adecuadas, reducir el tiempo de realización y ejecución de las mismas y, en gran parte, disminuir los altos costos que se generan.

De esta investigación resulta la propuesta de **Estrategias de Prueba** para evaluar la calidad en cualquier software OO, incluyendo las etapas del proceso de prueba, técnicas utilizadas y los requerimientos a evaluar.

El presente trabajo tiene como alcance la realización de una Estrategia de Prueba para software OO aplicada a un caso de estudio, evaluando cinco Requerimientos No Funcionales que propician la calidad del mismo. Este trabajo se llevará a cabo bajo la metodología Investigación Acción apoyado en el método DESMET.

Después de culminar la investigación se puede afirmar que la aplicación de una adecuada estrategia permite, además de planificar de forma eficiente y organizada el proceso de pruebas, identificar formalmente las fallas que se pueden presentar durante el desarrollo de un software, y de esta manera, garantizar la obtención de un producto de alta calidad.

El trabajo de investigación está organizado en siete capítulos: **Capítulo I**: Planteamiento del problema, describe el motivo de la investigación, los objetivos y su justificación; **Capítulo II**: Marco Teórico, presenta una revisión bibliográfica que proporciona al lector un marco teórico de referencia con respecto a la base fundamental de Calidad de Software y la influencia de esta en el proceso de Prueba; **Capítulo III**: Marco Metodológico, comprende un análisis de la metodología aplicada en la realización de la investigación, con el fin de llevar una planificación de los pasos a seguir; **Capítulo IV**: Desarrollo, describe las actividades realizadas en cada uno de los pasos expuestos en la metodología; **Capítulo V**: Resultados, resalta los resultados obtenidos en el desarrollo de la investigación; **Capítulo VI**: Conclusiones, presenta las conclusiones obtenidas después de desarrollar la investigación y analizar los resultados obtenidos en la misma; **Capítulo VII**: Recomendaciones, sugiere recomendaciones para trabajos futuros referentes al tema. Los Anexos incluyen información utilizada como complemento en la investigación y los Apéndices contienen material de elaboración propia que constituyen los detalles de todo lo el desarrollo de la investigación.

Capítulo I. Planteamiento del Problema

Cómo obtener un software con calidad y cómo evaluar esa calidad son el eje central del presente trabajo de investigación. Para entender las bases teóricas de esta problemática es necesario puntualizar los diferentes aspectos que lo relacionan, tales como: el planteamiento del problema, definir los objetivos, justificarlo y analizar los alcances y limitaciones.

I.1.-Planteamiento del Problema

Uno de los más relevantes problemas que enfrenta la Ingeniería del Software es el relacionado a la calidad. Las empresas entregan sus productos finales con un 15% de defectos. Además muchas compañías están perdiendo entre el 30% y el 44% de su tiempo y dinero en rescribir los productos software que ya han sido terminados Este tema ha sido motivo de investigación de ingenieros, especialistas y comercializadores de software, los cuales se han orientado a alcanzar dos objetivos fundamentales: **Cómo obtener un Software con Calidad** y **Cómo evaluar esa Calidad**.

La Ingeniería del Software es una disciplina relativamente reciente respecto a otras disciplinas productivas, mercantiles o industriales. La demanda y complejidad de la producción de software crecen a mayor velocidad que las metodologías, el personal capacitado y las herramientas para automatizar su producción, lo cual genera problemas. Aunque la incorporación de herramientas CASE (*Computer Aided Software Engineering*, por sus siglas en inglés) ayuda a resolver parte de estos problemas, la producción de

software continúa siendo una actividad de uso intensivo del recurso humano, por lo cual se considera cien por ciento intelectual (Estivill-Castro, 1999).

Se habla entonces de la **"crisis del software"**, donde los productos se entregan con demoras, los costos de sus desarrollos exceden lo inicialmente presupuestado y adicionalmente no cumplen con los requerimientos originales ni de calidad. Estos últimos requerimientos vienen dados por conceptos de confiabilidad, agilidad de respuesta, flexibilidad, consistencia, facilidad de mantenimiento, facilidad de comprensión, etc., lo que comúnmente es conocido como **Requerimientos No Funcionales (RNF)** del software, los cuales deben estar implícitamente incorporados al hablar de calidad. Lo que impone una congruencia entre los requerimientos y las características del producto para lograr plena satisfacción del usuario.

La problemática se intensifica cuando los Ingenieros del Software apresuran indebidamente las etapas de análisis, diseño y codificación obviando los requisitos de **Prueba**. Cuando las pruebas no se realizan, la calidad del producto final es, a menudo, muy pobre y es mucho mayor el tiempo y el costo que se van a emplear arreglando los defectos que los clientes detectarán una vez entregado el producto (Humprey, 1997).

De todo lo expuesto y en combinación con las nuevas tendencias en programación, que han venido sustituyendo a los enfoques clásicos en el mundo de la Ingeniería del Software, donde aparece una clara diferenciación en la aplicación de pruebas en software convencional y en software Orientado a Objetos (OO), se ha propuesto desarrollar una

investigación que conduzca a dar respuesta a la siguiente pregunta: ¿es posible diseñar una Estrategia de Prueba que garantice la calidad del software respecto a sus características no funcionales en software OO?. Para el desarrollo de esta investigación se plantean los siguientes objetivos

I.2.-Objetivo General

Definir Estrategias de Prueba que permitan garantizar la calidad del software OO en función del cumplimiento de los requerimientos no funcionales.

I.3.-Objetivos específicos

- Desarrollar un marco teórico que sirva como referencia para la propuesta de Estrategias de Prueba de Software OO tomando en cuenta los factores relacionados con la calidad, requerimientos no funcionales del software y pruebas.
- Elaborar la propuesta de Estrategias de Prueba de software OO para cinco (5) requerimientos no funcionales: Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad los cuales son esenciales para garantizar la Calidad del Software OO.
- Aplicar las Estrategias de Prueba a un caso de estudio: Sistema de Comercialización de la Empresa QUIMBIOTEC. El caso de estudio era inicialmente Proyectos DID-KMS, este software tuvo que ser reemplazado debido a que el mismo no era un software Orientado a Objetos, el diseño estaba realizado bajo el paradigma OO más no la implementación, siendo esta la condición indispensable que debia cumplir el caso de estudio seleccionado.

- Elaborar conclusiones y recomendaciones basadas en los resultados de la aplicación de las Estrategias de Prueba al caso de estudio.

I.4.-Justificación

En la "industria del software" el concepto de **calidad** ha sido abordado de una manera intensiva por diferentes autores como: Pressman (2002), Sommerville (2002), Wiegers (1999), Pleeger (1999) entre otros, y hoy por hoy constituye vanguardia dentro de los procesos de mejoramiento continuo referidos a los estándares de producción, en particular, cuando éstos se concentran en la producción de Sistemas de Información y software especializado

La **calidad** es sinónimo del cumplimiento de los **Requerimientos Funcionales (RF)** y de los **Requerimientos No Funcionales (RNF)** de un software. Estos últimos son a menudo más importantes en la determinación de éxito o fracaso de un sistema cuando se habla de calidad, ya que se refieren a cómo los usuarios esperan que el sistema funcione, proporcionándole asi a los diseñadores una comprensión clara de las expectativas del usuario (Wiegers, 1999).

Según Pressman (2002), la **Prueba de Software** surge como un elemento crítico para garantizar la calidad, generando revisiones de las especificaciones, del diseño y de la codificación que evalúan al producto antes de ser entregado al cliente. Por lo tanto no es raro que, una organización de desarrollo de software, emplee, en lo relativo a la prueba, hasta un 40% del esfuerzo total de un proyecto con un costo asociado de hasta cinco veces más que el resto de los pasos de la Ingeniería del Software juntos.

Estos tres aspectos (Calidad de Software, RNF, Pruebas) son la base fundamental del presente trabajo de investigación en software OO. Esta última delimitación fue precisa ya que el mismo es un enfoque novedoso, que ha venido sustituyendo a los enfoques clásicos en el mundo de la Ingeniería del Software, debido al potencial de este paradigma para fomentar software de calidad.

I.5.-Limitaciones y alcance

Las limitaciones y alcance del presente trabajo de investigación son las siguientes:

- La aplicación de la propuesta será probada a un solo caso de estudio, debido a las limitaciones de tiempo.
- Las Estrategias de Prueba será realizada sólo para Software OO, debido al potencial del mismo para obtener software de calidad.
- Las Estrategias de Prueba serán realizadas para evaluar cinco Requerimientos No Funcionales (Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad) que propician la calidad del software OO (ISO, 1996) las cuales se llevarán a cabo bajo la metodología Investigación Acción.

Una vez conocido el problema, establecido los objetivos y el alcance de la investigación, se procede a estructurar el marco teórico que ayudará a sentar las bases para el desarrollo del presente trabajo de investigación.

Capítulo II. Marco Teórico

En el presente capítulo se establece el marco referencial que permite tener una visión general de los aspectos teóricos en los que se fundamenta el trabajo de investigación. Los aspectos tratados están orientados hacia la **Calidad y Pruebas de Software**. La Calidad del software hace énfasis en los Requerimientos que la propician, mientras que las Pruebas hacen énfasis en los aspectos empleados en un proceso de prueba (Etapas, Técnicas, Estrategias y Casos de Prueba).

II.1.-Calidad

Según La Organización Internacional para la Estandarización (ISO por sus siglas en inglés), según su estándar ISO 8402, el término *Calidad* se define como (ISO, 1998):

"Conjunto de características de una organización, proceso o producto que le confieren la aptitud para satisfacer necesidades explícitas e implícitas. La calidad hace que un producto o servicio deba, en todos los aspectos, cumplir con el uso específico para el cual fue creado".

Según Gonzáles (2002), el aseguramiento de la calidad toma en cuenta todas aquellas acciones planificadas y sistemáticas necesarias para proporcionar la confianza en que un producto o servicio satisfaga los requisitos de calidad establecidos. Para que sea efectivo, se requiere una evaluación permanente de aquellos factores que influyen en la adecuación del diseño y de las especificaciones según las aplicaciones previstas.

Proyectando esto hacia la "industria del software", se incorpora el concepto de **calidad** ante el auge de las Tecnologías de la Información y la gran competitividad existente. El término **Calidad del Software** se ha convertido en uno de los mayores retos de muchos desarrolladores de software (Monsalve, 2000).

II.2.-Calidad del Software

La **Calidad del Software** está definida como la concordancia entre los requerimientos funcionales, el rendimiento explícitamente establecido, los estándares de desarrollo explícitamente documentados y las características implícitas que se espera de todo software desarrollado profesionalmente (Pressman, 2002). Esta definición queda sustentada por los siguientes puntos:

1. Los requerimientos explícitos del software son la base de la medida de calidad. La falta de concordancia con los mismos es una falta de calidad.
2. Los estándares específicos definen un conjunto de criterios de desarrollo que guían la manera en que se hace la Ingeniería del software. Si no se siguen los criterios, habrá seguramente poca calidad.
3. Existe un conjunto de **requerimientos implícitos** que ha menudo no se nombran (Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad). Si el software cumple con sus requerimientos explícitos pero falla en los implícitos, la calidad del software no se logra.

Estivill (1994) define la **Calidad del Software** como un conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad, además de cumplir con la funcionalidad, debe ser sinónimo de eficiencia, flexibilidad, corrección, confiabilidad,

mantenibilidad, portabilidad, usabilidad, seguridad e integridad, los cuales corresponden a requerimientos implícitos o no funcionales del software.

Muchos autores afirman que para lograr el control de calidad éste debe ser aplicado a lo largo de todo el ciclo de vida del proyecto. Pressman (2002) asegura que la garantía de calidad del software es una "actividad de protección" que se aplica a lo largo de todo el proceso de Ingeniería del software, la cual engloba: métodos y herramientas de análisis, diseño, codificación y prueba; revisión de técnicas formales que se aplican durante cada paso; estrategia de prueba multiescalada; control de la documentación del software y de los cambios realizados; procedimientos que aseguren un ajuste a los estándares de desarrollo del software; y mecanismos de medida y de información.

Wiegers (1999), establece que la calidad, en sus diferentes etapas, debe ser definida tanto por el usuario como por los analistas, diseñadores y los que mantienen el software, ya que los **requerimientos** dados por el usuario pueden aclarar las expectativas implícitas de calidad y el criterio que los diseñadores tomarán en cuenta para crear un producto totalmente satisfactorio.

II.3.-Requerimientos del Software.

Como ha sido resaltado, uno de los aspectos que se debe definir es el de los **requerimientos del software**. En esta sección se describen los conceptos y clasificaciones referidas a los mismos.

Los **requerimientos del software** son una descripción abstracta de los servicios que el sistema debería proporcionar al cliente, y las limitaciones bajo las cuales éste debería operar. Los requerimientos del software deben especificar la conducta externa del sistema y no deben estar comprometidos con las características de diseño del mismo (Sommerville, 2002).

Estos presentan tres importantes propósitos: 1) permiten a los desarrolladores entender cómo el cliente quiere que trabaje el sistema, 2) especifican a los diseñadores la funcionalidad y las características que el sistema debe tener, 3) especifican al grupo de prueba lo que deben demostrar para convencer al cliente que el sistema satisface sus necesidades (Pleeger, 1997).

Considerando lo antes expuesto se puede decir que los requerimientos del software son una descripción de lo que el sistema debería hacer y cómo lo debería hacer, tomando siempre en cuenta las necesidades del cliente. De ahí su importancia y la necesidad de que deban ser definidos y manejados de la forma más adecuada posible.

Según los especialistas en el tema: Sommerville (2002), Bass (1998), Bosch (2000), Dromey (1996), Pressman (2002), entre otros, los requerimientos del software se clasifican en: **Requerimientos Funcionales (RF)** y **Requerimientos No Funcionales (RNF)**. A continuación será explicada esta clasificación tomando en cuenta cada uno de los conceptos dados por los autores referidos.

Requerimientos Funcionales (RF): Representan lo que el sistema debe proveer, como el sistema debe reaccionar ante una entrada específica, y como debe comportarse en una situación particular. Se puede decir que los RF son los que describen lo que el sistema debe hacer. Es importante que describan el ¿Qué? y no el ¿Cómo?. Estos requerimientos, al tiempo que avanza el proyecto, se convierten en los algoritmos, la lógica y gran parte del código del sistema.

Requerimientos No Funcionales (RNF): Son aquellos requerimientos que aparecen junto con las necesidades del usuario y definen las restricciones y propiedades de un sistema. Los RNF tienen que ver más con los requerimientos del proceso que con los requerimientos del producto por dos razones:

1. **Calidad del sistema:** un buen proceso conlleva a un buen producto.
2. **Mantenibilidad del sistema:** los requerimientos pueden ser impuestos de tal forma que los métodos de desarrollo usados para diseñar e implementar el sistema sean compatibles con aquellos que van a ser usados para el mantenimiento del sistema.

Bosch (2000) denomina a los RNF como los **Requerimientos de Calidad**. Por lo tanto, para hacer referencia a los RNF en las siguientes explicaciones, se utilizará solo el término **Requerimientos de Calidad**.

Sommerville (2002), afirma que la mejor forma de asegurar que los **Requerimientos de Calidad** sean verificables, es expresándolos cuantitativamente. Esto puede lograrse midiendo algunas **características del software** que están relacionadas, de alguna manera,

con el sistema deseado; Partiendo de la definición de **características del software,** de acuerdo al estándar ISO (ISO 9126, 1991), estas son aquellas propiedades que identifican un producto software.

Wiegers (1999) afirma que las características del software incluyen, entre otras: con qué facilidad se usará el producto; con qué rapidez se ejecutara; cuán bien se comportará cuando se presenten situaciones inesperadas. Estas características del producto, pueden ser llamadas **Características de Calidad**, si los diseñadores saben cuáles de ellas son más cruciales para el desarrollo con éxito del software. Estas son parte de los Requerimientos de Calidad del sistema y a la vez son determinantes para su logro. Finalmente los **Requerimientos de Calidad** marcan la diferencia entre lo que el producto hace y lo que se supone debería hacer.

El presente trabajo de investigación utilizará el **Modelo de Calidad Sistémico MOSCA** (Martínez, 2001) el cual permite identificar las **Características de Calidad** que deben ser evaluadas en un software. De los elementos que conforman el Modelo de MOSCA, sólo serán tomados en cuenta las categorías asociadas a la Calidad del Producto, estas son: Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad, a excepción de la categoría de Funcionalidad, ya que, para efectos del alcance de este trabajo sólo serán consideradas las **Características de Calidad** determinantes para el cumplimiento de los **Requerimientos de Calidad** en el software.

Para alcanzar el objetivo de esta investigación se establecerán Estrategias de Prueba en función de estas Características, tomando en cuenta los siguientes basamentos teóricos referentes a la **Prueba de Software**.

II.4.-Prueba de software

De acuerdo con la IEEE (1990) la **Prueba** se define como:

"Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente".

Cuando se habla de condiciones especificas, se supone la presencia de un ambiente de operación de prueba, para el cual deben existir determinados valores de entradas, de salidas y condiciones que delimitan al ambiente de operación. Formalmente, esto se conoce como **Caso de prueba**, herramienta de gran importancia para el logro de los objetivos de la **Prueba** (IEEE, 1990).

Pressman (2002) afirma que la **Prueba** no puede asegurar la ausencia de defectos en el software, sólo puede demostrar que éstos existen. Entonces, una prueba tiene éxito si encuentra un error no detectado anteriormente. Por lo tanto el objetivo fundamental de la **Prueba** es encontrar un error en el sistema.

Muchos autores, especialistas en el tema, como Krutchen (2000), Pressman (2002), Pfleger (1998), Cardoso (2001) y Sommerville (2002) entre otros, afirman que el proceso de ejecución de **Prueba** debe ser considerado durante todo el ciclo de vida de un proyecto,

para así poder lograr un producto de alta calidad. Su éxito dependerá si se sigue una **Estrategia de Prueba** adecuada.

II.4.1.-Estrategia de Prueba

La **Estrategia de Prueba** de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuanto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software (Pressman, 2002).

Al desarrollar una Estrategia de Prueba es importante conocer las necesidades de la organización y cuáles son críticas para un desarrollo de software exitoso, para así lograr como resultado datos relevantes que le den información a la Empresa cuanto nivel de calidad ha alcanzado su producto (Jacobson, 1992).

II.4.2.-Pruebas en Software OO

Todo software diseñado a partir del modelo OO requiere ser sometido a pruebas con el propósito de garantizar su calidad. Las características propias del enfoque OO como: la encapsulación, la herencia, el polimorfismo, pase de mensajes y el comportamiento dinámico, proporcionan ventajas visibles en el diseño y programación de software. Sin embargo, estas nuevas características crean problemas desafiantes en las fases de aplicación de pruebas (Binder, 1996). Por lo tanto es importante definir las diferentes etapas que

permitan aumentar progresivamente el alcance de las pruebas y así cubrir el objetivo final de las mismas.

II.4.3.-Etapas de pruebas para software OO

Las etapas del proceso de pruebas progresan desde pruebas para los elementos más pequeños del sistema (componentes), hasta pruebas para sistemas completos (Kruchten, 2000).

II.4.3.1.-Prueba de Análisis y Prueba de Diseño en Software OO: Las pruebas que se realizan tanto al Análisis OO (AOO) como al Diseño OO (DOO), verifican si se cometieron errores o fallas en estas etapas. Para la realización de estas pruebas, aunque no se realizan de manera simultánea, se aplican las mismas técnicas que permiten descubrir errores dentro del contexto de la sintaxis, semántica y pragmática de los modelos generados en éstas etapas. Los modelos de análisis y diseño no se pueden probar en el sentido convencional, ya que no pueden ejecutarse. Sin embargo, se pueden utilizar revisiones técnicas formales para examinarlos (Pressman, 2002).

Las características a ser evaluadas en los modelos AOO y DOO son la **exactitud** y la **consistencia**, las cuales influyen en los RNF del software. Las técnicas que permiten evaluar cada una de estas características son nombradas a continuación y explicadas detalladamente en el *Apéndice A*:

- Prueba de Exactitud
- Prueba de Consistencia

II.4.3.2.-Pruebas de Unidad en Software OO: Cuando se considera el software OO, el concepto de unidad cambia, ya que se habla de clases y objetos. En vez de probar un módulo individual, la unidad más pequeña comprobable es la clase u objeto encapsulado, permitiendo detectar errores de datos, lógica y algoritmos (Pressman, 2002).

La prueba de unidad trata de verificar que esta proporcione los servicios que promete, que responda correctamente a las condiciones esperadas y, más aún, ante las inesperadas. Aspectos adicionales pueden verificar si la clase contiene y permite disponer de todas las funciones asociadas a ella o que cada método de la clase ejecute su responsabilidad especificada (Bashir & Goel, 1999).

Autores como MacGregor (1994), Murphy (1994) y Binder (1996), han propuesto técnicas para llevar a cabo la pruebas de unidad considerando aspectos como el orden en que los métodos son sometidos a la prueba, el orden en que una jerarquía de clases puede ser probada, ejercitar el flujo de datos o bien el análisis del estado del objeto. Estas técnicas son nombradas a continuación y explicadas en el *Apéndice A*:

- Pruebas Estructurales
- Prueba de Valores Limites
- Prueba Incremental

II.4.3.3.-Pruebas de Integración en Software OO: La prueba de integración es aquella que se realiza para identificar cualquier falla cuando dos o más elementos desarrollados independientemente se combinan para ejecutar sus funcionalidades (Bashir &

Goel, 2000). Entre las fallas típicas se tienen las debidas a defectos internos y externos en las interfaces, fallas de sincronización y de rendimiento.

Cuando se desarrollan sistemas OO claramente las operaciones de los datos se integran para formar objetos y clases de objetos. No existe una equivalencia directa para la prueba de módulos en los sistemas OO. Sin embargo, se sugiere que se prueben juntos los grupos y clases que actúan en combinación para proveer un conjunto de servicios. A esto se le llama **pruebas de clústers** (Sommerville, 2002).

Es importante destacar que, a diferencia de las otras etapas, en ésta existen algunas **estrategias** que permiten realizar el proceso de prueba a nivel de integración de diferentes formas según sea conveniente.

Debido a que el software OO no tiene una estructura de control jerárquico, las estrategias convencionales de integración descendente (top-down) y ascendente (bottom-up) tienen muy poco significado. Existen dos estrategias de integración (Pressman, 2002):

a.-Pruebas basadas en hilos: En este tipo de estrategia se integran el conjunto de clases requeridas, para responder una entrada o suceso al sistema. La prueba de hilos puede ser usada después de que los componentes han sido probados individualmente e integrados en sub-sistemas, para utilizar esta estrategia se tiene que identificar cómo se lleva a cabo el procesamiento de cadenas de eventos en el sistema.

b.-Pruebas basadas en uso: En esta estrategia se comienza la construcción del sistema probando aquellas clases (llamadas clases independientes), que utilizan muy pocas (o ninguna) clases servidoras. Después de que las clases independientes se prueban, esta secuencia de pruebas por capas de clases dependientes continúa hasta que se construye el sistema completo.

A continuación se nombran las técnicas correspondientes a esta etapa, las cuales están explicadas en el *Apéndice A*:

- Técnica de Camino de Mensajes
- Técnica de Overbeck

II.4.3.4.-Pruebas de Sistema en Software OO: Al nivel de sistema, los detalles de conexiones de clases desaparecen. La validación del software OO se centra en las acciones visibles al usuario y salidas reconocibles desde el sistema (Pressman, 2002).

La Prueba de Sistema permite detectar errores en comportamiento con respecto a las especificaciones de los requerimientos (funcionales y de calidad) lo cual genera fallas de funcionalidad del sistema (corrección, completitud) y fallas de robustez, detectando errores en el comportamiento esperado por el cliente.

Para realizar la evaluación en la etapa de Sistema son consideradas las siguientes técnicas, las cuales están explicadas en el *Apéndice A*:

- Prueba de Ejecución
- Pruebas de Aceptación

- Prueba Bajo Stress
- Prueba Negativa
- Prueba de Volumen
- Prueba de Carga
- Prueba Basada en Requerimientos
- Prueba Ergonómica
- Prueba de Documentación
- Prueba de Recuperación
- Prueba de Rendimiento
- Prueba de Instalación
- Pruebas de Configuración

Para llevar a cabo las pruebas en cada una de las etapas, es necesario realizar casos de prueba diseñados en base a las técnicas establecidas en cada una de las etapas. A continuación se explica todo lo referente a este concepto.

II.4.4.-Diseño de Casos de Prueba para Software OO

Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse (Jacobson, 1992).

El diseño de los casos de prueba es una clave para el éxito de las pruebas, ya que si los casos de prueba no son representativos y no ejercen las funciones que demuestren los

defectos y las validaciones del sistema, el proceso de prueba no es satisfactorio (Pfleeger, 1998). Se deben considerar tres aspectos a la hora de diseñar casos de prueba, estos son: la estructura, el enfoque y el método empleado para llevar a cabo el caso de prueba. Estos aspectos serán explicados en el *Apéndice B*. En la figura N° 1 se muestran de manera esquemática:

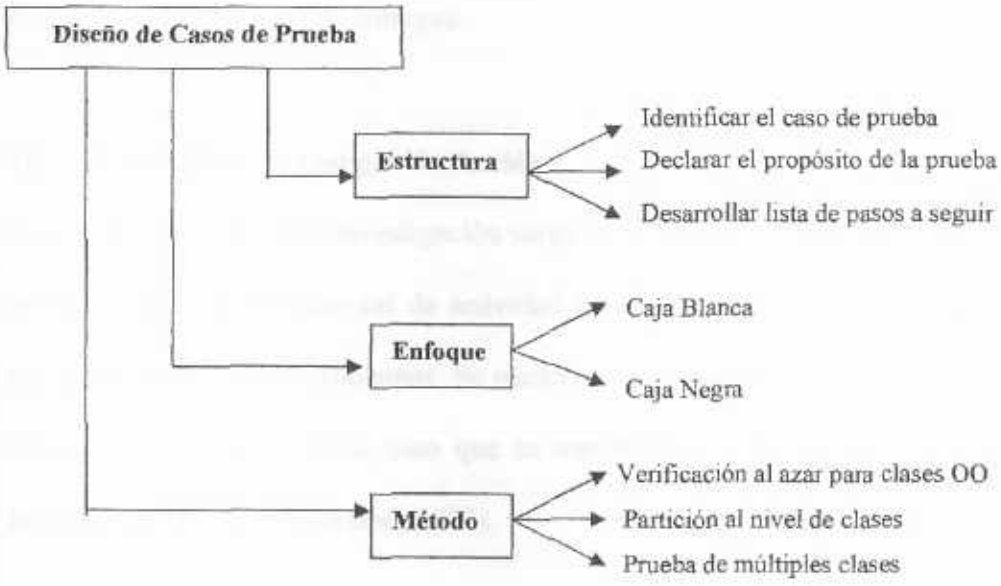

Figura N° 1. Diseño de Casos de Prueba. Fuente: Elaboración propia.

En este capítulo se trataron con detalle la Calidad del Software, los Requerimientos de Calidad en los que se basará la investigación de las pruebas a realizar y por último las Pruebas de Software OO, donde se precisaron los puntos más importantes para el logro de un proceso de pruebas exitoso, estos son: establecer una Estrategia que lleve a cabo este proceso de forma planificada y que esta contemple Casos de Prueba que permitan detectar las fallas del sistema. Definidos estos elementos básicos, es necesario conocer cuál será la metodología a seguir para el desarrollo del presente trabajo de investigación.

Capítulo III. Marco Metodológico

El presente capítulo tiene como objetivo presentar la metodología de Investigación Acción y el método DESMET aplicados en este trabajo de investigación. Se comienza con una explicación de la metodología Investigación Acción; después se tratan las bases del método DESMET, y luego se establece el ciclo metodológico a utilizar, determinando la secuencia de etapas y sus principales entregas.

III.1.-Metodología Investigación Acción

El concepto de acción de investigación surge de las ciencias del comportamiento y se aplica en la observación de sistemas de actividad humana llevados a cabo durante el proceso en que se intenta resolver problemas. Su núcleo es que el investigador no sigue siendo ajeno a la materia de investigación, sino que se transforma en un participante dentro del grupo humano pertinente (Checkland, 1993).

La metodología **Investigación Acción** utiliza esquemas de investigación menos rígidos y cíclicos, que convergen progresivamente en el resultado esperado, por lo que está orientada a la resolución de problemas suaves que según Checkland (1993) son problemas relacionados con las manifestaciones del mundo real de los sistemas de actividad humana, caracterizado por un sentido de desajuste, que elude la definición precisa, entre lo que se percibe como la realidad y lo que se percibe que podría ser la realidad.

El fundamento de la Investigación Acción es que los procesos de los Sistemas de Actividad Humana pueden ser estudiados mejor si se introducen cambios dentro de estos procesos y

se observan los efectos que producen estos cambios (Baskerville, 1999). En base a este fundamento, fue considerada la selección de ésta metodología, ya que la mejor forma de comprobar la validez de la propuesta del presente trabajo es aplicar ésta a un determinado sistema, para observar los efectos que se producen y de allí derivar las respectivas conclusiones.

Por tanto se puede decir que la razón fundamental por la que fue seleccionada esta metodología para llevar a cabo el presente trabajo, es que la misma permite interacción entre los investigadores y el contexto en donde es desarrollada la investigación. La investigación documental es sólo una fase que contempla la metodología, esta permite llevar a campo la investigación, con lo cual se puede comprobar si la investigación llega a resultados satisfactorios.

Baskerville (1999), afirma que el tipo de aprendizaje creado por la investigación acción representa un entendimiento mejorado de un problema organizacional-social complejo. Además, el dominio de la investigación acción es más clara donde la organización humana interactúa con Sistemas de Información (SI).

Para Baskerville (1999), la Investigación Acción en Sistemas de Información tiene cuatro características principales:

1.-Apunta a un mejor entendimiento de una situación social inmediata, con énfasis en la naturaleza compleja y multivariada de esta configuración social en el dominio de los SI.

2.-Asiste simultáneamente en la solución de un problema práctico y expande el conocimiento científico.

3.-Es ejecutada en colaboración y aumenta las competencias de los actores respectivos.

4.-Es principalmente aplicable al entendimiento de procesos de cambio en sistemas sociales.

Una descripción de la Investigación Acción es propuesta por Baskerville (1999), detallando un proceso cíclico de cinco fases como se muestra en la figura N° 2.

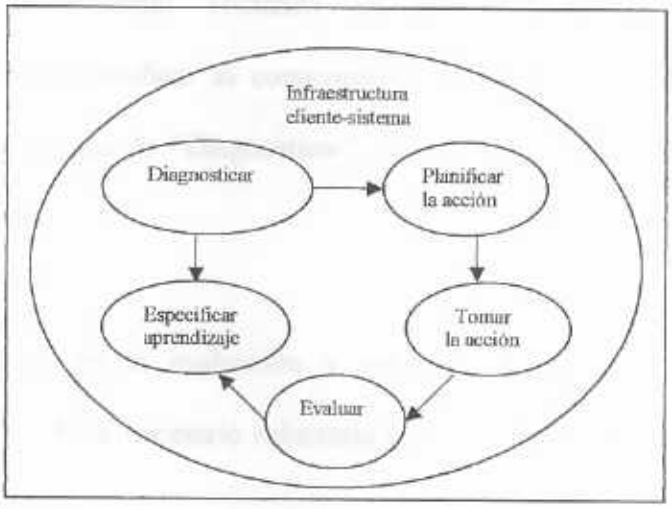

Figura N° 2. Ciclo de Investigación Acción

Fuente: (Baskerville, 1999)

Diagnosticar: Corresponde a la identificación de los problemas primarios que son las causas subyacentes por las cuales la organización desea cambiar. Este diagnostico desarrolla los supuestos teóricos (Hipótesis del trabajo) acerca de la naturaleza de la organización y el dominio del problema.

Planificar la acción: Especifica las acciones organizacionales que deberian revelar o mejorar los principales problemas. El plan establece el objetivo del cambio y su enfoque.

Tomar la acción: Implementa la acción planificada. Los participantes e investigadores colaboran en la intervención activa dentro de la organización, encausando ciertos cambios.

Evaluar: Se evalúan los resultados para determinar si los efectos teóricos de la acción fueron alcanzados y si estos efectos solventaron los problemas. Donde los cambios no fueron exitosos debe establecerse algún marco para la próxima iteración del ciclo de Investigación Acción.

Especificar el aprendizaje: Tomando los resultados de las evaluaciones, los investigadores deben especificar el conocimiento adquirido con el fin de proporcionar nuevas bases para la fase de "Diagnosticar", en preparación de una futura iteración de Investigación Acción.

El carácter empírico en la evaluación y selección de métodos de la metodología Investigación Acción hace necesario reforzarla con otros métodos de investigación como **DESMET** (Bass, 1996).

III.2.-Método DEMET

DESMET es un método para la evaluación de métodos y herramientas de Ingeniería del Software. Ayuda al evaluador de una organización a planear y ejecutar un ejercicio de evaluación que sea imparcial y fiable. A excepción de los experimentos formales, las evaluaciones de DESMET son dependientes del contexto, lo que revela su carácter sistémico, y significa que no se puede esperar que un método sea mejor para todas las circunstancias (Kit, 1996). En el *Apéndice C* se encuentra mayor detalle del método DESMET.

En el presente trabajo se aplicará el ciclo metodológico descrito en la figura Nº 3.

III.3 Ciclo Metodológico

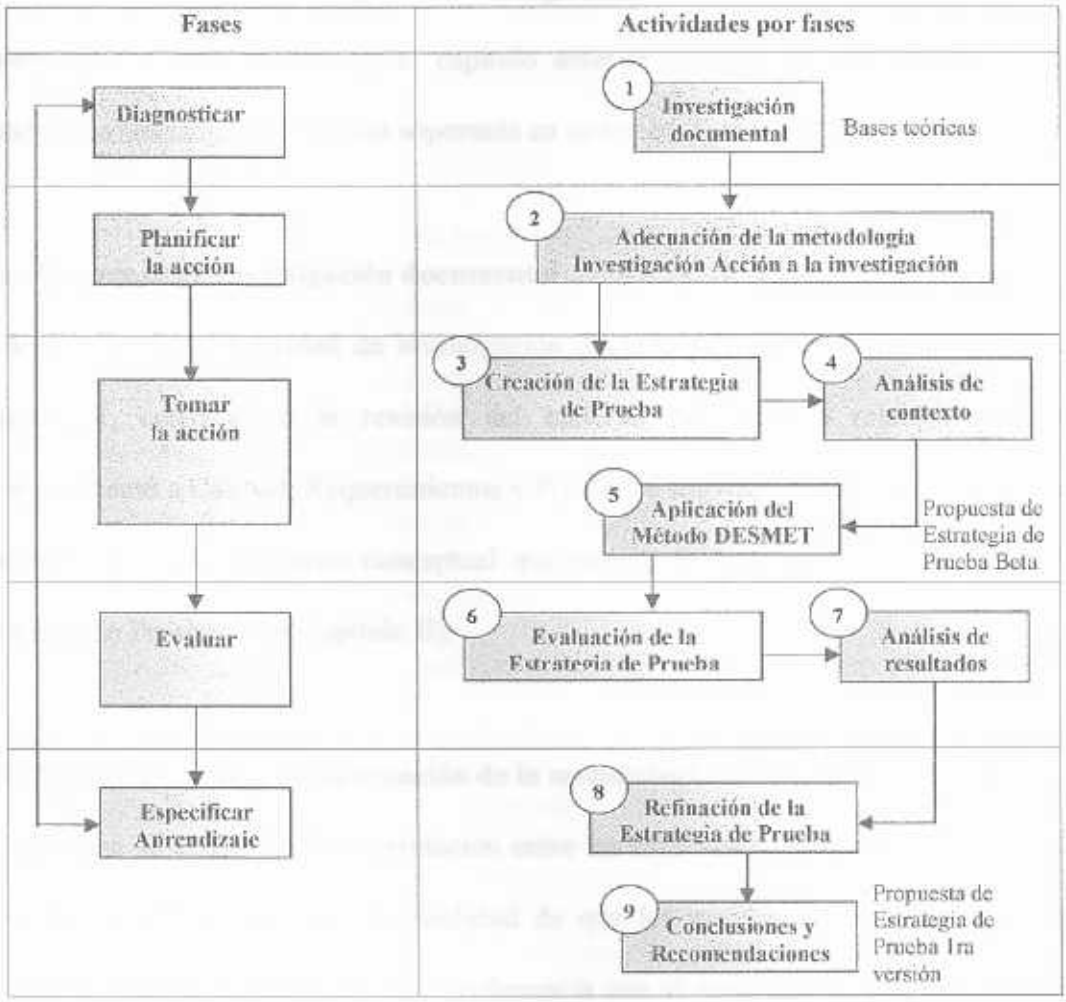

Figura Nº 3: Ciclo metodológico utilizado en el trabajo de investigación.
Fuente:(LISI, 2002), adaptado al presente trabajo de investigación.

El ciclo metodológico presentado es una adaptación de la metodología Investigación Acción al presente trabajo, el cual establece la secuencia y las principales salidas de las actividades a realizar en cada fase de la metodología. En el siguiente capítulo se describe el contenido y desarrollo de cada una de estas actividades.

Capítulo IV. Desarrollo

El presente capítulo explica cómo fueron desarrolladas las actividades especificadas en cada una de las fases del ciclo metodológico utilizado en el presente trabajo de investigación. Como se dijo en el capítulo anterior se trata de una adaptación de la metodología Investigación Acción soportada en su tercera fase por el método DESMET.

IV.1.-Diagnosticar-Investigación documental

El desarrollo de la actividad de investigación documental, correspondiente a la fase de diagnosticar, consiste en la revisión del material bibliográfico relacionado con lo correspondiente a Calidad, Requerimientos y Prueba de software. Esta actividad tiene como finalidad consolidar el marco conceptual que servirá de base para la generación de la Estrategia de Prueba. (Ver Capítulo II).

IV.2.-Planificar la Acción-Adecuación de la metodología Investigación Acción

En esta fase se construye la interrelación entre las diferentes actividades, propuestas por fases, de la metodología, con la finalidad de que los eventos se desarrollen de forma secuencial y que a su vez mantengan coherencia con el tema que se está tratando. En el caso particular del presente trabajo de investigación las cinco fases generaron nueve actividades (Ver Capítulo III). Las actividades que inciden directamente en la acción, en la evaluación y en el aprendizaje se desarrollan a continuación.

IV.3.-Tomar la acción-Diseño de la Estrategia de Prueba, análisis de contexto, aplicación de DESMET

Esta fase contempla la ejecución de tres actividades. El desarrollo de cada una de ellas será explicado a continuación.

IV.3.1.-Tomar la acción-Diseño de la Estrategia de Prueba: Para diseñar la Estrategia de Prueba se contempla una secuencia de pasos previos que conducen al establecimiento de la Estrategia propiamente dicha, quedando esta establecida en el último paso, con el fin de crear una estructura de trabajo que genere una evaluación de un software OO. Estos pasos y sus productos son el resultado del análisis y la síntesis del marco teórico y aportes propios. A continuación se presentan cada uno de ellos manteniendo la secuencia en su elaboración:

IV.3.1.1.-Definición de las Etapas del Proceso de Prueba: Se establecieron las etapas del proceso de prueba con sus respectivas técnicas. Estas etapas se basan en las etapas que debe seguir un proceso de desarrollo de software, ya que las pruebas deben contemplarse a lo largo de todo el ciclo de vida de elaboración del software. Las etapas identificadas en este paso especifican el orden que seguirá la Estrategia de Prueba.

IV.3.1.2.-Asociación de las Técnicas de Prueba a cada una de las Características de Calidad: Debido a que el objetivo principal es evaluar los RNF del software se asociaron las técnicas de prueba a cada Subcaracterística de las Características de Calidad, con el fin de saber específicamente cómo probar cada una de ellas.

IV.3.1.3.-Definición de las Listas de Verificación: Se establecieron Listas de Verificación como herramienta principal para llevar a cabo las pruebas. Estas fueron definidas para cada una de las Subcaracterísticas, permitiendo dar revisión formal de las

mismas. Una lista de verificación es un simple formulario de preguntas las cuales dependen del objetivo para el cual son usadas.

En este caso, el contenido de las preguntas se basa en las técnicas previamente asociadas a las Subcaracterísticas y en el objetivo planteado para evaluar las mismas. Es importante considerar, dado que el proceso de prueba se lleva cabo en cinco (5) etapas, que se definieron listas de chequeo para cada una de ellas, dividiendo las preguntas de cada etapa según las Subcaracterísticas.

La razón por la cual fueron utilizadas listas de verificación es porque pueden ser usadas para el análisis de cualquier atributo cualitativo y permiten transferir observaciones obtenidas a través de revisiones e inspecciones a medidas numéricas reales. Además no se ve limitada por factores externos y las preguntas pueden ser expresadas genéricamente, por lo cual podrán ser aplicadas a la evaluación de los Requerimientos No Funcionales de cualquier software OO.

IV.3.1.4.-Diseño de los Casos de Prueba: Al estudiar el concepto de casos de prueba, se estableció una relación entre éstos y las listas de verificación, ya que para dar respuesta a algunas preguntas de las mismas se deben realizar casos de prueba específicos. Los casos de prueba o también llamados escenarios, dependerán del tipo de software que se esta evaluando y de las tareas especificas que éste realice. Todos los casos de prueba deben ser registrados.

IV.3.1.5.-Obtención de resultados: En este paso se identificó la forma en que se obtendrán los resultados a partir de las respuestas dadas a las preguntas de las listas de

verificación, especificando los cálculos que serán realizados y la forma en que estos serán presentados.

IV.3.1.6.-Diseño Final de la Estrategia de Prueba: Tomando en cuenta el desarrollo de los pasos anteriormente explicados, se estableció el conjunto de actividades que conformarán la Estrategia de Prueba.

IV.3.2.-Tomar la acción-Análisis del contexto: El objetivo es estudiar el contexto donde será evaluada la propuesta. Para ello se analizan los criterios que el método DESMET utiliza para determinar las condiciones en las que se desenvuelve la Estrategia de Prueba. Estos criterios son: contexto de evaluación, naturaleza del impacto, naturaleza de la Estrategia de Prueba, alcance del impacto, madurez de la Estrategia de Prueba, tiempo de entrenamiento y madurez de la Organización que lleva a cabo la evaluación.

IV.3.3.-Tomar la acción-Aplicación del método DESMET: El objetivo de esta actividad es seleccionar el método de evaluación que mejor se adapte a la Estrategia de Prueba. Para esto se identifican las condiciones que favorecen a cada uno de los métodos de evaluación presentados por DESMET, con el fin de seleccionar aquel que cumpla con el mayor número de condiciones favorables frente al análisis del contexto.

IV.4.-Evaluar-Evaluación de la Estrategia de Prueba, análisis de resultados

En esta fase se realiza la evaluación de la Estrategia de Prueba utilizando el método arrojado por DESMET: **Análisis de Características-Estudio de Caso** y se desarrolla un

análisis de los resultados obtenidos de la evaluación dando retroalimentación a la Estrategia de Prueba.

IV.4.1.-Evaluar-Evaluación de la Estrategia de Prueba: Para realizar la evaluación de la Estrategia de Prueba se desarrollaron los pasos que contempla el método obtenido, denominado **Análisis de Características-Estudio de Caso**. Estos pasos son: alcance de la evaluación, base de la evaluación, roles y responsabilidades, procedimiento de la evaluación (creación de las características de evaluación, aplicación de la Estrategia de Prueba al caso de estudio y evaluación de las características), limitaciones, escala de tiempo y esfuerzo realizado.

IV.4.2.-Evaluar-Análisis de resultados: Se analizan los resultados obtenidos de las características establecidas para evaluar la Estrategia y en base a éstos se determina si ésta es exitosa o no, basándose en los puntos en donde la evaluación presente deficiencias.

IV.5.-Especificar Aprendizaje-Refinación de la propuesta, conclusiones y recomendaciones

Una vez analizados los resultados se identifican los puntos más importantes a los que se llegó a través de toda la investigación.

IV.5.1.-Especificar aprendizaje-Refinación de la propuesta: Se proponen modificaciones necesarias con el fin de aumentar la confiabilidad de la aplicación de la

Estrategia de Prueba y sus posibilidades de éxito. En esta actividad se obtiene una primera versión formal de la Estrategia de Prueba.

IV.5.2.-Especificar aprendizaje-Conclusiones y Recomendaciones: Se establecen las conclusiones relacionadas con la Estrategia y con los resultados obtenidos después de la evaluación de la misma. Además se sugieren algunas recomendaciones para futuros refinamientos e investigaciones relacionadas. (Ver Capítulo VI y VII).

Una vez explicadas todas las actividades desarrolladas en el presente trabajo de investigación, a continuación se muestran los resultados obtenidos en cada una de ellas, los cuales representan los aportes más significativos y resaltantes de la Investigación.

Capítulo V. Resultados

El presente Capítulo muestra los resultados obtenidos durante el desarrollo secuencial de las actividades definidas en la metodologia Investigación Acción utilizada en el presente trabajo de investigación. Los primeros resultados se obtuvieron durante la creación de la estrategia que contempla definir; las etapas, las listas de verificación, el registro de casos de prueba y las pautas para procesar los resultados. Todos estos resultados condujeron a la generación de la Estrategia de Prueba, la cual se considera el principal y más importante aporte de este trabajo de investigación. Igualmente se consideran resultados, aquellos obtenidos durante la evaluación de la Estrategia de Prueba a través de la aplicación del método arrojado por DESMET que consistió en una prueba de campo (Quimbiotec-Sistema de Comercialización) que permitió verificar la efectividad de la Estrategia de Prueba.

V.1.-Resultados obtenidos en el Diseño de la Estrategia

A continuación se presentan los resultados obtenidos en los pasos previos al diseño de la Estrategia de Prueba. El último paso corresponde al diseño final de la misma.

V.1.1.-Definición de las Etapas del Proceso de Prueba: Con el fin de mejorar la calidad y confiabilidad del producto, el proceso de prueba quedó estructurado de tal manera que se puede evaluar desde cada segmento de código hasta el software completo. En la Figura N° 4 se presenta esquemáticamente como quedaron establecidas las etapas del proceso de prueba.

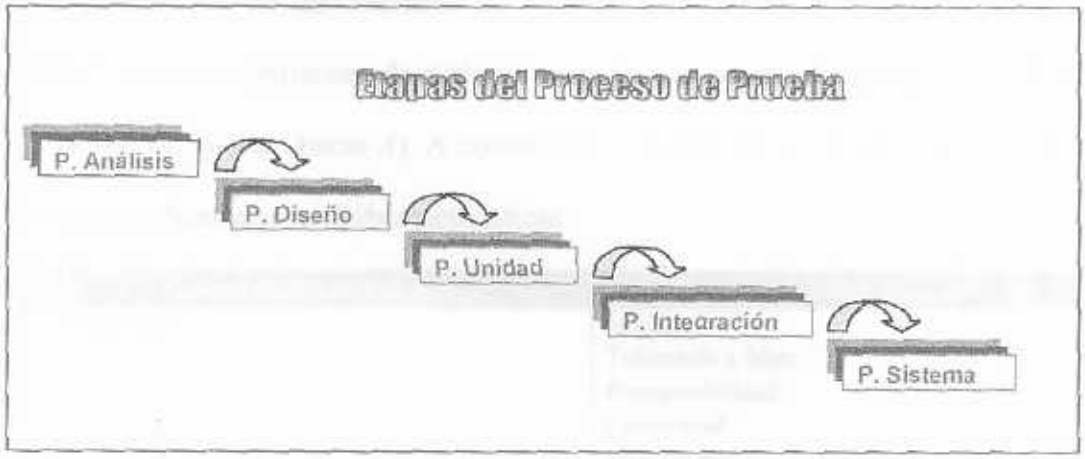

Figura N° 4: Etapas del proceso de prueba. Fuente: Elaboración propia

Una vez establecidas las etapas a seguir durante el proceso de prueba, se asociaron las técnicas a cada una de estas etapas con el fin de asegurar que se realicen las verificaciones correspondientes. Esto se muestra en la figura N° 5.

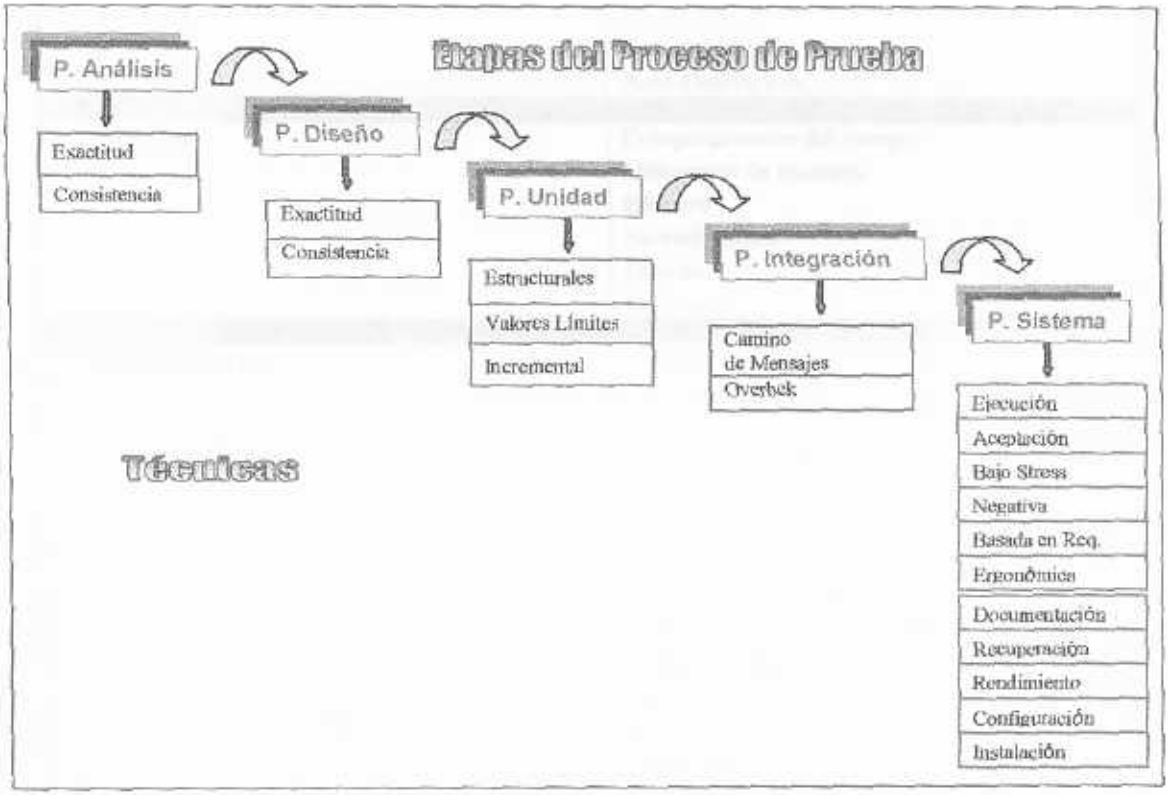

Figura N° 5. Técnicas asociadas a las etapas del Proceso de Prueba. Fuente: Elaboración propia

V.1.2.-Asociación de las Técnicas de Prueba a cada una de las Características de Calidad: Las Características de Calidad consideradas son las descritas en el Modelo de Calidad MOSCA (ver *Anexo A*). A continuación se presenta en la tabla Nº 1, cada una de las Características con sus Subcaracterísticas.

Característica de Calidad	Subcaracterísticas
1.-Fiablidad	Madurez Tolerancia a fallas Recuperabilidad Correctitud Encapsulado Estructurado
2.-Usabilidad	Facilidad de comprensión Capacidad de aprendizaje Interfaz gráfica Operabilidad Completo Consistente Efectivo Especificado Documentado Auto-Descriptivo
3.-Eficiencia	Comportamiento del tiempo Utilización de recursos Efectivo No-redundante Directo Utilizado
4.-Mantenibilidad	Capacidad de análisis Capacidad de cambio Estabilidad Capacidad de prueba Acoplamiento Cohesión Encapsulamiento Atributos de madurez Sistema de estructura de control Sistema de estructura de información Descriptivo Correctitud Estructural Parametrizado

Característica de Calidad	Subcaracteristicas
5.-Portabilidad	Adaptabilidad Capacidad de instalación Coexistencia Capacidad de reemplazo Consistente Parametrizado Encapsulado Cohesivo Especificado Documentación Auto-Descriptivo No-redundante

Tabla N° 1. Características de Calidad y Subcaracteristicas. Fuente: Elaboración propia

El resultado obtenido de la asociación de las técnicas de prueba y las Características de Calidad se muestra en las siguientes tablas:

1.-Técnicas de Prueba para la Fiabilidad

Subcaracteristica	Objetivo	Técnicas que aplican
Madurez	Evaluar la capacidad que tiene el software para evitar fallas.	**Prueba Negativa** ✓ Hacer que el sistema falle intencionalmente para medir su capacidad de respuesta frente a un error.
Tolerancia a Fallas	Verificar la capacidad del software para mantener un nivel de rendimiento especifico ante un error, es decir, la capacidad de continuar procesando en caso de falla.	**Prueba de Valores Límites** ✓ Evaluar valores frontera, es decir, los valores mínimo y máximo que la unidad puede aceptar. **Prueba Bajo Stress** ✓ Evaluar la habilidad del sistema para seguir operando apropiadamente ante bajos recursos o competencias para los recursos. ✓ Ejecutar el sistema de manera que demande recursos en cantidad, frecuencia o volúmenes anormales. **Prueba Negativa** ✓ Hacer que el sistema falle intencionalmente para medir su capacidad de continuar su

Subcaracteristica	Objetivo	Técnicas que aplican
		ejecución a pesar de la falla. **Prueba de Volumen** ✓ Someter al software a una gran cantidad de datos para determinar si los límites alcanzados hacen que este falle.
Recuperabilidad	Verificar que el proceso de recuperación del sistema restaura apropiadamente la base de datos, la aplicación y el sistema a un estado conocido o deseado.	**Prueba de Recuperación** ✓ Exponer al software a condiciones extremas y verificar que la recuperación se realiza correctamente.
Correctitud	1.- Evaluar la capacidad de cómputo. 2.- Comprobar la completitud de las formas estructurales y del software como un todo. 3.- Evaluar la consistencia.	**Prueba Estructural** ✓ Verificar que las formas estructurales de las clases sean completas. **Prueba de Ejecución** ✓ La capacidad de cómputo esperada se puede evaluar durante la ejecución del software en aquellos módulos en donde se apliquen cálculos. **Prueba de Carga** ✓ Probar diferentes cargas para evaluar la capacidad de cómputo.
Estructurado	En caso de que el software no sea desarrollado en lenguaje OO puro se debe verificar que sea estructurado, es decir, que siga las reglas de la programación estructurada.	**Prueba Estructurales** ✓ Esta técnica permite evaluar los valores de las variables, las constantes y los tipos de datos y si éstos son usados en el contexto en que se definieron.
Encapsulado	Si el software esta desarrollado en lenguaje OO puro no es necesario verificar que sea encapsulado, pero si no lo es, se debe aplicar las técnicas de la subcaracteristica: Estructurado.	**Prueba Estructurales** ✓ Esta técnica permite evaluar los valores de las variables, las constantes y los tipos de datos y si éstos son usados en el contexto en que se definieron.

Tabla N° 2 Técnicas de Prueba para la Fiabilidad. Fuente: Elaboración propia

2.-Técnicas de Prueba para la Usabilidad

Subcaracteristica	Objetivo	Técnicas que aplican
Facilidad de Compresión	Probar la capacidad que tiene un producto de software para facilitar al usuario el entendimiento del software y la forma en que puede ser utilizado, es importante para ello considerar lo siguiente:	**Prueba de aceptación** ✓ Evaluar el comportamiento del usuario frente al sistema. **Prueba Ergonómica**

Subcaracterística	Objetivo	Técnicas que aplican
	1.-Evaluar las interfaces de usuario (qué tan fáciles son de manejar y entender). 2.-Evaluar todo lo referente a la documentación. 3.-Evaluar la utilización del vocabulario.	✓ Crear pruebas para verificar que los elementos que componen las pantallas del sistema son fáciles de comprender. **Prueba de Documentación** ✓ Crear pruebas para la documentación a nivel de usuario, para evaluar si la misma es fácil de comprender.
Capacidad de Aprendizaje	Evaluar la capacidad del producto de software para habilitar al usuario el aprendizaje de la aplicación.	**Prueba de Documentación** ✓ Evaluar la ayuda del software y los manuales de usuarios para medir la capacidad del software para ser aprendido.
Interfaz gráfica	Evaluar todo lo relacionado con la interfaz: 1.-Uso del color y la naturaleza del diseño grafico. 2.-Navegación. 3.-Caracteristicas de las ventanas: menús, tamaño, posición, estado, y la conformidad con los estándares.	**Prueba de Interfaz** ✓ Verificar que la interfaz cumple con los estándares de diseño grafico (colores, tamaños, etc.). **Prueba Ergonómica** ✓ Crear pruebas para verificar que los elementos que componen las pantallas son fáciles de comprender.
Operabilidad	Evaluar la capacidad del producto de software para habilitar al usuario a operarlo y controlarlo, por lo cual es importante considerar: ✓ La documentación. ✓ La interfaz.	**Prueba de Documentación** ✓ Crear pruebas para la documentación a nivel de usuario para verificar que los manuales y el comportamiento del sistema sean consistentes. **Prueba Ergonómica** ✓ Crear pruebas para verificar que los elementos que componen la interfaz son fáciles de operar.
Completo	Verificar que se tienen todos los elementos necesarios para definir e implementar la forma estructural.	**Pruebas Estructurales** ✓ Verificar que las formas estructurales sean completas, es decir, que posean todos sus elementos. Por ejemplo, una instrucción condicional que puede abortar; funciones que no generan salidas, etc. ✓ Verificar que se definen e implementan todas las clases necesarias para

Subcaracterística	Objetivo	Técnicas que aplican
		satisfacer todos los requerimientos del sistema.
Consistente	Verificar que la forma estructural mantiene sus propiedades o funcionalidad y si todos sus elementos contribuyen a reforzar su intención o efecto.	**Prueba de consistencia** ✓ Verificar que las relaciones entre todas las entidades sean consistentes a nivel de análisis y diseño. **Prueba Estructural** ✓ Verificar que la forma estructural de las clases sea consistente con su funcionalidad.
Efectivo	Verificar que el software presenta sólo los elementos necesarios para definir e implementar la forma estructural.	**Prueba de consistencia** ✓ Verificar que los elementos representados en los modelos son los necesarios para cumplir con los requerimientos. **Pruebas Estructurales** ✓ Verificar si los algoritmos utilizados son efectivos. **Técnica de Overbeck** ✓ Probar que no exista un objeto que no se este usando. ✓ Comprobar que no exista duplicidad de objetos. ✓ Determinar la relación entre los objetos y definir cuáles son aquellos que son redundantes.
Especificado	Verificar que la funcionalidad del software es descrita con pre-condiciones y post-condiciones	**Prueba de Documentación** ✓ Evaluar que estén bien descritas las pre-condiciones y las post-condiciones de cada unos de los elementos del código.
Documentado	Verificar que tanto el propósito como las propiedades del sistema están explícitamente definidas en el contexto de la forma estructural. Aplicable a clases, métodos, ciclos, estructuras de datos, variables, constantes, tipos, etc.	**Prueba de Documentación** ✓ Verificar que el código (ciclos, estructura de datos, tipos, variables, constantes, etc.) este bien documentado.
Auto-Descriptivo	Verificar que los nombres de los módulos y los identificadores en el software tienen significado en el contexto de la aplicación. Aplicable a clases, métodos, ciclos estructuras de datos, variables,	**Prueba de Exactitud** ✓ Verificar si el modelo refleja con exactitud el dominio del problema del mundo real.

Subcaracterística	Objetivo	Técnicas que aplican
	constantes, tipos, etc.	**Prueba de Documentación** ✓ Evaluar que los nombres de las clases, objetos, métodos y variables correspondan con su propósito. ✓ Verificar que los nombres de los componentes de la interfaz sean auto-descriptivo.

Tabla N° 3 Técnicas de Prueba para la Usabilidad. Fuente: Elaboración propia

3.-Técnicas de Prueba para la Eficiencia

Subcaracterística	Objetivo	Técnicas que aplican
Comportamiento del tiempo	Verificar la capacidad que tiene el software para proveer respuestas y tiempos de procesamiento apropiados en tiempo de ejecución bajo condiciones específicas.	**Prueba Bajo Stress** ✓ Evaluar los tiempos de respuesta del sistema bajo condiciones específicas. ✓ Identificar la carga pico que el sistema puede manejar en un determinado tiempo. **Prueba de Rendimiento** ✓ Verificar el rendimiento del software en tiempo de ejecución. ✓ Verificar si el número de tareas completadas en un tiempo estimado es satisfactorio.
Utilización de Recursos	Verificar la capacidad que tiene el software para utilizar cantidades apropiadas de los recursos en tiempos de ejecución bajo condiciones específicas.	**Prueba Bajo Stress** ✓ Probar al software utilizando bajos recursos a nivel de CPU. ✓ Verificar que el sistema funcione correctamente y sin error bajo determinadas condiciones de stress: Poca memoria en el servidor, máximo número de clientes conectados, etc. **Prueba de Rendimiento** ✓ Probar el rendimiento utilizando recursos en tiempo de ejecución.
Efectivo	Ya fue explicada anteriormente en la Usabilidad.	Son las mismas técnicas aplicadas en la Usabilidad.
No-redundante	Evaluar que el software presente sólo los elementos lógicos necesarios para definir una forma estructural. El objetivo de la No redundancia se diferencia al de la Efectividad en que la No-	**Pruebas Estructurales** ✓ Chequear la no redundancia a nivel de clases, es decir, que las condiciones y elementos que se utilizan en los métodos no sean redundantes, etc.

Subcaracterística	Objetivo	Técnicas que aplican
	redundancia involucra redundancia lógica, en vez de redundancia computacional.	
Directo	Verificar si una expresión es directa, es decir, que la abstracción, representación, y la estructura del cálculo es congruente con el problema original. Aplicado a instrucciones, expresiones, variables, constantes, tipos.	**Pruebas Estructurales** ✓ Evaluar las instrucciones del código para comprobar que exista una congruencia con lo planteado originalmente. Además esta técnica permite evaluar los valores de las variables, constantes y condiciones con el propósito de comprobar que estas cumplan realmente con su función original. **Técnica de Overbeck** ✓ Evaluar si el pase de mensajes (a través de parámetros) entre las clases es directo.
Utilizado	Verificar que una forma estructural ha sido definida y luego utilizada en su alcance.	**Pruebas Estructurales** ✓ Verificar que todos los componentes de una clase son utilizados. **Técnica de camino de mensajes** ✓ Verificar que todas las clases u objetos son utilizados, verificando las relaciones entre los mismos.

Tabla N° 4 Técnicas de Prueba para la Eficiencia. Fuente: Elaboración propia

4.-Técnicas de Prueba para la Mantenibilidad

Subcaracterística	Objetivo	Técnicas que aplican
Capacidad de análisis	Verificar la capacidad que tiene el software para ser diagnosticado ante fallas o deficiencias.	**Prueba de Documentación** ✓ Verificar si la documentación a nivel de código está lo suficientemente explicativa para ser entendida y poder analizar de donde proviene la falla.
Capacidad de Cambio	Verificar la capacidad que tiene el software para ser modificado, entendiendo por modificación cualquier corrección, mejora o adaptación del software.	**Prueba de Consistencia** ✓ Verificar que existe consistencia entre los requerimientos, diseño del sistema, e implementación. **Pruebas Estructurales** ✓ Verificar que las estructuras de las funciones o métodos están basadas en el uso de parámetros. **Técnica de Overbeck**

Subcaracterística	Objetivo	Técnicas que aplican
		✓ Verificar que las comunicaciones entre objetos y/o módulos se realizan a través de invocaciones de parámetros. **Prueba de Documentación** ✓ Verificar que los cambios son documentados.
Estabilidad	Verificar la capacidad que tiene el software para evitar efectos inesperados después de realizar alguna modificación en el mismo.	**Prueba Overbeck** ✓ Realizar cambios en los objetos y módulos para evaluar si éstos conservan la correcta interacción con los componentes que corresponden. **Prueba de Ejecución** ✓ Evaluar la ejecución del software luego de realizar cambios en los objetos y módulos.
Capacidad de Prueba	Verificar la capacidad que tiene el software para que una vez que sea modificado este siga siendo válido.	**Prueba de Documentación** ✓ Evaluar si la documentación es especifica y adecuada ya que el fácil entendimiento de lo que inicialmente se programó, permite realizar cambios satisfactoriamente y que los casos de prueba sean fáciles de establecer. **Prueba de Ejecución** ✓ Evaluar el impacto de los cambios. **Prueba de Consistencia** ✓ Verificar que el sistema este bien definido.
Acoplamiento	Medir la interconexión entre los componentes del software. Evaluar que el software presente un acoplamiento débil. El acoplamiento depende de la complejidad de la interfaz entre los módulos, el punto en el que se entra o se hace referencia al módulo y que datos pasan a través de la interfaz.	**Prueba de consistencia** ✓ Verificar que las colaboraciones entre las clases estén representadas adecuadamente en los modelos de análisis y diseño. **Técnica de Overbeck** ✓ Evaluar la interconexión entre los componentes que conforman el sistema para comprobar el nivel de acoplamiento. **Técnica de camino de mensajes** ✓ Verificar que todas las colaboraciones entre clases u objetos son a través de mensajes.
Cohesión	Evaluar si todos los elementos de una clase están enlazados unos con otros y contribuyen a llevar a cabo un objetivo.	**Técnica de Consistencia** ✓ Verificar que las responsabilidades se han repartido adecuadamente entre las clases.

Subcaracterística	Objetivo	Técnicas que aplican
		Pruebas Estructurales ✓ Verificar que las operaciones de las clases sean lo mas especificas posibles.
Encapsulamiento	Ya fue explicada en la Fiabilidad	Las técnicas ya fueron explicada en la Fiabilidad
Atributos de madurez	Evaluar las características físicas y las medidas asociadas a la edad y uso del sistema de software.	**Pruebas de Documentación** ✓ Obtener, a través de la documentación, las características que corresponden con los atributos de madurez del software, tales como :edad del software, horas de mantenimiento, etc.
Sistema de Estructura de Control	Evaluar las propiedades asociadas a la Modularidad, los atributos de control Inter-modular, la selección y uso de construcciones de flujo de control, la manera en la cual el programa o sistema es descompuesto en algoritmos, y el método con el cual los algoritmos son implementados.	**Prueba de Consistencia** ✓ Verificar en los modelos de diseño cómo esta estructurado el software en relación a la división de las clases. **Pruebas Estructurales** ✓ Verificar las sentencias de flujo de control de las clases. **Técnica de Overbek** ✓ Verificar los atributos de control entre las clases colaboradoras.
Sistema de Estructura de Información	Evaluar aquellas propiedades asociadas al almacenamiento y flujo de información en un programa o sistema, incluyendo definiciones de datos y las características de las entradas y salidas del sistema.	**Pruebas de estructura de información** ✓ Verificar que se tenga una estructura de almacenamiento de información. ✓ Verificar, si se utiliza una base de datos, que este en tercera forma normal. ✓ Chequear la conectividad con la base de datos durante la ejecución del sistema.
Descriptivo	Evaluar si el software es fácil de entender y usar, en cuanto a tipografía, nombres y comentarios.	**Prueba de Exactitud** ✓ Verificar si el modelo refleja con exactitud el dominio del problema del mundo real. **Prueba de documentación** ✓ Verificar que las etiquetas nombres y comentarios sean descriptivas.
Correctitud	Es el mismo objetivo de la Correctitud de la Fiabilidad	Son las mismas técnicas de la Correctitud de la Fiabilidad
Estructural	Que cumpla con ser: Estructurado (igual que la Fiabilidad), Efectivo (igual a la Eficiencia), No-redundante (igual a la Eficiencia), Directo (igual a la Eficiencia)	Aplicar las mismas técnicas de: estructurado, efectivo, no-redundante y directo.

Subcaracterística	Objetivo	Técnicas que aplican
Parametrizado	Comprobar que el código contiene solo los parámetros necesarios para caracterizar una función, clase o método.	**Pruebas estructurales** ✓ Evaluar los parámetros internos de las clases. **Técnica de overbeck** ✓ Evaluar que los parámetros son correctos al interactuar dos clases colaboradoras

Tabla N° 5 Técnicas de Prueba para la Mantenibilidad. Fuente: Elaboración propia

5.-Técnicas de Prueba para la Portabilidad

Subcaracterística	Objetivo	Técnicas que aplican
Adaptabilidad	Verificar que el software puede ser adaptado a diferentes ambientes sin aplicar acciones diferentes a las previstas.	**Prueba de Ejecución** ✓ Ejecutar el sistema en los diferentes ambientes donde este pueda ser ejecutado dentro de la organización para verificar si el sistema es adaptable a los mismos. **Prueba de Configuración** ✓ Comprobar que el software se adapta a diferentes configuraciones de PC.
Capacidad de Instalación	Evaluar la capacidad que tiene el producto software para ser instalado en un ambiente especificado.	**Prueba de Instalación**: ✓ Comprobar la capacidad que tiene el software para realizar el proceso de instalación correctamente. ✓ Evaluar el proceso de instalación en distintos ambientes.
Coexistencia	Evaluar la capacidad que tiene el producto software para co-existir con otro software independiente en un ambiente común compartiendo recursos comunes.	**Basada en Requerimientos** ✓ Verificar en los requerimientos con cuales software debe coexistir. **Prueba de Ejecución** ✓ Ejecutar el sistema junto con los software especificados en los requerimientos en un ambiente común, compartiendo recursos.
Capacidad de Reemplazo	Evaluar la capacidad del producto de software para ser usado en lugar de otro producto de software especificado para el mismo propósito en un mismo ambiente.	**Prueba Basada en Requerimientos** ✓ Según lo especificado en los requerimientos, probar al ejecutar el software, que este cumple con los mismos y por ende, sustituir el software por otro que cumpla con el mismo propósito en un mismo ambiente.

Subcaracterística	Objetivo	Técnicas que aplican
Consistente	Ya fue explicada en la característica de Usabilidad.	Ya fue explicada en la característica de Usabilidad.
Parametrizado	Ya fue explicada en la Mantenibilidad.	Las técnicas se explicaron anteriormente la Mantenibilidad.
Encapsulado	Ya fue explicada en la Fiabilidad.	Las técnicas ya se explicaron en la Fiabilidad.
Cohesivo	Ya fue explicada en la Mantenibilidad.	Las técnicas ya se explicaron en la Mantenibilidad.
Especificado	Ya fue explicada en la Usabilidad.	Las técnicas son las mismas que se explicaron anteriormente en la Usabilidad.
Documentación	Ya fue explicada en la Usabilidad	Ya se explicaron las técnicas en la Usabilidad.
Auto-Descriptivo	Ya fue explicada en la Usabilidad.	Ya fue explicada anteriormente en la Usabilidad.
No-Redundante	Ya fue explicada en la Eficiencia.	Las técnicas son las mismas que las de Eficiencia.

Tabla N° 6 Técnicas de Prueba para la Portabilidad. Fuente: Elaboración propia

V.1.3.-Definición de las Listas de Verificación: Las Listas de Verificación están basadas en la identificación de las técnicas de Prueba para evaluar cada Subcaracterística de las Características de Calidad. Estas listas están clasificadas según cada una de las etapas del Proceso de Prueba. Para dar respuesta a cada pregunta se considera una escala del 1 al 5, en donde el uno (1) siempre es la respuesta menos significativa y cinco (5) la más significativa. Esta listas son mostradas en el *Apéndice D*; sin embargo, a los fines ilustrativos se presenta la Lista de Verificación de la etapa de Análisis.

Es importante señalar que, como valor agregado a la investigación, las Listas de Verificación fueron automatizadas en el Programa **EPS-OO ChekList** utilizando como herramienta de desarrollo Visual Basic. Las especificaciones de dicho programa se encuentran detallas en el *Apéndice E*.

Lista de Verificación para la Prueba de Análisis

Característica	Pregunta	Casos de prueba sugeridos	Evaluación
1.-Fiabilidad.			
1.1 Madurez	1.-¿Se realizó un proceso de levantamiento de información apropiado?	No aplica un caso de prueba específico, es un simple revisión formal (N/A).	1= Inaceptable 2= Debajo del Promedio 3= Promedio 4= Bueno 5= Excelente
	2.-¿El levantamiento de información fue establecido formalmente?	N/A	1= No 5= Si
1.2 Correctitud	1.-¿ Están siendo considerados todos los procesos necesarios para dar solución al problema?	N/A	1= No 5= Si
2.-Usabilidad			
2.1 Facilidad de Comprensión	1.-¿Cuál es el grado de facilidad de entender los modelos de análisis?	N/A	1= Muy difícil 2= Difícil 3=Dificultad Media 4= Fácilmente 5=Muy fácil
2.2 Operabilidad	1.-¿Se analizaron los papeles o funciones de las personas que trabajan en los procesos?(Ejemplo: clientes, representantes de ventas, etc).	N/A	1= Inaceptables 2= Debajo del Promedio 3= Promedio 4= Buenas 5= Excelentes
	2.-¿Se tiene un modelo de casos de usos?	N/A	1= No 5= Si
2.3 Completo	1.-¿El modelo de casos de uso contempla todas las especificaciones del sistema?	N/A	1= Ninguna 2= Muy pocas 3= Algunas 4= Casi todas 5= Todas
2.4 Consistente	1.-¿Se mantiene la notación estándar en todos los modelos?	N/A	1= No 5= Si
2.5 Efectivo	1.-¿Se identificó cómo interactúan las entidades del negocio?	N/A	1= No 5= Si
	2.-¿Fueron optimizados procesos realizados en la empresa durante el análisis?	N/A	1= No 5= Si
2.6 Auto-Descriptivo	1.-¿Se tiene una notación estándar para la construcción de los modelos?	N/A	1= No 5= Si
	2.-¿Todas las relaciones implicadas en el análisis están representadas adecuadamente?	N/A	1= Inaceptable 2= Debajo del Promedio 3= Promedio 4= Bueno 5= Excelente
	3.-¿Los expertos en el dominio del problema aceptan las definiciones dadas en el análisis?	N/A	1= No 5= Si
	4.-¿El modelo representa con exactitud el dominio del problema?	N/A	1= No 5= Si

Característica	Pregunta	Casos de prueba sugeridos	Evaluación
3.-Eficiencia			
3.1 Efectivo	Aplican las mismas preguntas que para Efectivo de Usabilidad	N/A	
4.-Mantenibilidad			
4.1 Capacidad de Análisis	1.-¿Se identificaron todos los procesos de la empresa relacionados con el sistema?	N/A	1= Ninguno 2= Muy pocos 3= Algunos 4= Casi todos 5= Todos
4.2 Descriptivo	1.-¿Se tiene un modelo conceptual del negocio?	N/A	1= Inaceptable 2= Debajo del Promedio 3= Promedio 4= Bueno 5= Excelente
	2.-¿Se tienen identificados y representados los conceptos en el dominio del problema?	N/A	1= Inaceptable 2= Debajo del Promedio 3= Promedio 4= Bueno 5= Excelente
	3.-¿El modelo del negocio describe el trabajo de un área específica del negocio?	N/A	1= No 5= Si
4.3 Correctitud	Aplican las mismas preguntas que en la Fiabilidad	N/A	
5.-Portabilidad			
5.1 Consistente	Aplican las mismas preguntas que Consistente de Usabilidad	N/A	
5.2 Auto Descriptivo	Aplican las mismas preguntas que Auto-Descriptivo de Usabilidad	N/A	

Tabla N° 7 Lista de Chequeo para la etapa de Análisis. Fuente: Elaboración propia

V.1.4.-Diseño de Casos de Prueba: Para registrar cada uno de los Casos de Prueba que se realicen para dar respuesta a las Listas de Chequeo, se diseñó, como forma de almacenamiento, una planilla cuyos campos son los datos requeridos para estructurar un Caso de Prueba. Cada pregunta de las Listas de Verificación puede generar uno (1) o más Casos de Prueba. Esta planilla, se muestra a continuación. El instructivo para llenarla es mostrado en el *Apéndice F*.

Planilla de Caso de Prueba

Datos iniciales			
Fecha:		**Nº de Caso de Prueba**	
Característica:	**Subcaracterística:**		
Pregunta de la lista de chequeo			
Información del Caso de Prueba			
Descripción:		**Enfoque:**	
Datos de entrada:		**Resultados Esperados:**	
Procedimiento del caso de prueba			
1.- Pasos a seguir:			
2.- Condiciones externas:			
Resultados			
Resultados obtenidos:	**Completación**		
	Aprobado		
	No aprobado		
	En caso de no ser aprobado especificar:		
	Severidad de la falla		
	Grave		
	Menor		
Observaciones			

Figura Nº 6. Planilla de Caso de Prueba. Fuente: Elaboración propia.

V.1.5.-Obtención de Resultados: Se establecieron los pasos a seguir para obtener los resultados cuantitativos en base a las respuestas dadas en las Listas de Verificación. Los resultados están determinados por las respuestas y por la importancia que el cliente asigna a cada Subcaracterística.

V.1.5.1.-Ponderación de las Subcaracterísticas: Para determinar la importancia de cada Subcaracterística en la evaluación, el cliente da un peso (en una escala del 1 al 5) a cada

una de ellas a través de la Planilla de Ponderación, en donde el uno (1) siempre es el peso menos significativo y cinco (5) el más significativo. A continuación se muestra la planilla de ponderación. Ver instructivo en el *Apéndice G*.

PLANILLA DE PONDERACION					
Característica de calidad: Fiabilidad					
Subcaracterística	**Peso Asignado**				
	1	**2**	**3**	**4**	**5**
Madurez					
Tolerancia a Fallas					
Recuperabilidad					
Correctitud					
Estructurado					
Encapsulado					
Característica de Calidad: Usabilidad					
Subcaracterística	**Peso Asignado**				
	1	**2**	**3**	**4**	**5**
Facilidad de Comprensión					
Capacidad de Aprendizaje					
Interfaz grafica					
Operabilidad					
Completo					
Consistente					
Efectivo					
Especificado					
Documentado					
Auto-Descriptivo					
Característica de Calidad: Eficiencia					
Subcaracterística	**Peso Asignado**				
	1	2	3	4	5
Comportamiento del Tiempo					
Utilización de Recursos					
Efectivo					
No-Redundante					
Directo					
Utilizado					
Característica de Calidad: Mantenibilidad					
Subcaracterística	**Peso Asignado**				
	1	2	3	4	5
Capacidad de Análisis					
Capacidad de Cambio					
Estabilidad					
Capacidad de Prueba					

Acoplamiento					
Cohesión					
Encapsulamiento					
Atributos de Madurez					
Sist. de Estructura de Control					
Sist. de Estructura de Inf.					
Descriptivo					
Correctitud					
Estructural					
Parametrizado					
Característica de Calidad: Potabilidad					
Subcaracterística	**Pesos Asignados**				
	1	2	3	4	5
Adaptabilidad					
Capacidad de Instalación					
Coexistencia					
Capacidad de Reemplazo					
Consistente					
Parametrizado					
Encapsulado					
Cohesivo					
Especificado					
Documentación					
Auto-Descriptivo					
No-redundante					

Figura N° 7. Planilla de Ponderación. Fuente: Elaboración propia.

A continuación se detallan cada uno de los pasos establecidos para obtener los resultados en base a los dos criterios: respuestas a las preguntas y la ponderación.

V.1.5.2.- Dar respuesta a las preguntas de las Listas de Verificación: Las respuestas a las preguntas de las Listas de Verificación se pueden dar de forma directa o mediante la realización de **Casos de Prueba**. En caso de que sea a través de los Casos de Prueba las respuestas se conseguirán dependiendo de los resultados obtenidos en los mismos.

Los **Caso de Prueba** serán evaluados por medio de la siguiente escala:

Aprobado = 5

No Aprobado = { **Falla Menor** = 3 ; **Falla Grave** = 1

Una pregunta puede ser contestada por más de un **Caso de Prueba.** En este caso la respuesta a la pregunta siempre será el del Caso de Prueba que obtenga menor valor. Esto implica que de haber un caso de prueba no aprobado éste afectara al resto en cuanto a la calificación de la calidad. A continuación se muestra un ejemplo de esta explicación.

Ejemplo: Se tiene la pregunta: ¿El software responde satisfactoriamente a una falla?

Casos de Prueba para responder a la pregunta

Casos de Prueba	Valor obtenido
Caso de Prueba 1	5
Caso de Prueba 2	3
Caso de Prueba 3	5

Respuesta a la pregunta = 3

V.1.5.3.- Generación de resultados: A partir del procesamiento de las respuestas dadas en las Listas de Verificación se generan tres (3) tipos de resultados:

a.-Resultados de la presencia de Subcaracterísticas en cada etapa del proceso de prueba según la Característica de Calidad a la que corresponden.

- Se halla el promedio aritmético de las respuestas de cada pregunta de la Subcaracterística que se está evaluando.
- Los resultados del promedio aritmético se obtendrán en una escala del 1 al 5, considerando los siguientes valores para cada puntuación:

  1= La Subcaracterística no está presente en esta etapa.

2= La Subcaracterística se presenta de manera muy deficiente.

3= La Subcaracterística se presenta medianamente.

4= La Subcaracterística se encuentra presente.

5= La Subcaracterística se encuentra altamente presente.

b.-Resultados de la presencia de las Características de Calidad en cada una de las etapas, considerando la importancia dada por el cliente: Una vez obtenidos los resultados de todas las Subcaracterísticas se procederá a realizar los cálculos para obtener la evaluación de la Característica de Calidad. Este cálculo se realizará de la siguiente manera:

- Se halla el promedio ponderado de las Subcaracterísticas tomando en cuenta los pesos que le han sido asignados a cada una de ellas.
- Para hallar este promedio ponderado se multiplican los valores obtenidos de cada Subcaracterística (SC) por su peso correspondiente (P).
- Se suman los valores obtenidos de la multiplicación y se divide este valor entre la suma de todos los pesos. Este cálculo se representa a través de la siguiente fórmula:

$$\frac{\sum SC * P}{\sum P}$$

Este resultado representa la evaluación de presencia de la Característica de Calidad dentro de una etapa. Esta evaluación será representada en un rango del 1 al 5, al igual que los resultados dados para las Subcaracterísticas. El valor obtenido es llevado a porcentaje. A continuación se presenta un ejemplo de la obtención de estos resultados.

Ejemplo: Evaluación de la Fiabilidad en la etapa de Sistema

- Resultados por Subcaracterísticas

Subcaracterística 1: Madurez **Peso:** 5

Pregunta	Resultado
Pregunta 1	5
Pregunta 2	5
Pregunta 3	5
	Total= 15

Promedio aritmético = 15/3 = 5

Subcaracterística 2: Tolerancia a fallas **Peso:** 3

Pregunta	Resultado
Pregunta 1	5
Pregunta 2	5
Pregunta 3	3
	Total= 13

Promedio aritmético = 13/3 = 4.3

Resultados de la Fiabilidad

Resultado Subcaracterística	Peso de la Subcaracterística	Resultado×Peso
5	5	25
4.3	3	12.9
	Suma **Total** = 8	Suma **Total** = 37.9

Promedio Ponderado = 37.9/ 8 = 4.73

Este promedio se lleva a porcentaje respecto al máximo valor, es decir, cinco (5).

La Fiabilidad, en este caso, está presente en la etapa de Sistema en un 94,6 %.

c.-Resultados de la presencia de las Características de Calidad en todo el Sistema:

Después de realizar las evaluaciones de las Características de Calidad en cada una de las

etapas del proceso de prueba, se obtiene el porcentaje de presencia de cada una de las Características de Calidad en todo el sistema (**PPCS**) el cual es el promedio de los porcentajes de presencia de cada una de las Características en cada etapa del proceso de prueba (**PPCE**).

Ejemplo:

% de presencia de la Fiabilidad en la Etapa de Análisis: 80 %

% de presencia de la Fiabilidad en la Etapa de Diseño: 95 %

% de presencia de la Fiabilidad en la Etapa de Unidad: 90 %

% de presencia de la Fiabilidad en la Etapa de Integración: 98 %

% de presencia de la Fiabilidad en la Etapa de Sistema: 100 %

$$PPCS = \frac{\sum PPCE}{5}$$

El promedio de la presencia de la característica Fiabilidad = 92,6 %

Es importante señalar que se tomó como **valor mínimo de presencia** de la Característica de Calidad, **75 %.** Una presencia con un valor por debajo del **75%** se la considera deficiente en el Software que se está evaluando. En este caso se recomienda que sea revisada la Característica en todo el proceso de desarrollo del software.

V.1.6-Diseño Final de la Estrategia de Prueba: Todo lo anterior soporta el diseño de la Estrategia de Prueba. Esta consiste en un conjunto de acciones organizadas y secuénciales que se sintetizan en la Tabla N° 8. Esta Estrategia de Prueba, producto de la investigación, se ha nombrado, a los fines de su presentación y posterior referencia, como **EPS-OO**.

EPS-OO				
Actividad	**Descripción**	**Responsable**	**Producto Obtenido**	**Documento/ Entregable**
1-Solicitud de Prueba	Se debe procesar la planilla de solicitud de prueba para dar comienzo al análisis de la prueba solicitada.	- Solicitante	- Datos del Solicitante. - Tiempo estimado. - Información del equipo. - Definición de riesgos. - Plan de prueba.	Planilla de Solicitud de Prueba (ver planilla en el *Apéndice H*).
2-Recepción de los requisitos de la prueba	El evaluador debe verificar toda la información suministrada por el solicitante sobre el módulo o sistema a probar. Esta información servirá de apoyo a la ejecución de las pruebas.	- Líder del Proyecto. - Evaluador.	-Requerimientos del sistema (funcionales, no funcionales, HW, SW). - Manuales. - Documentos de Análisis y Diseño.	N/A
3-Presentación del Sistema o Módulo	El evaluador recibe una presentación del módulo o sistema. Debe revisar con el solicitante cómo opera el sistema, validaciones, cálculos, etc.	- Líder de Proyecto. - Evaluador	- Certificación del ambiente de prueba. - Certificación de la operabilidad del sistema o modulo. - Certificación de los cálculos y procesos.	N/A
4-Identificar los recursos necesarios para ejecutar las pruebas	Se identifican, en base a los recursos utilizados por el sistema, los recursos necesarios para ejecutar las pruebas, considerando de antemano el programa EPS-OO.	- Evaluador. - Líder de Proyecto	N/A	Planilla de Solicitud de Prueba.
5-Identificar riesgos y limitaciones	En esta actividad se deben detallar los riesgos y limitaciones que se pueden presentar durante la ejecución de las pruebas.	- Líder de Proyecto. - Evaluador	Riesgo y Limitaciones Certificadas.	Planilla de Solicitud de Prueba
6-Definir el alcance de la prueba	Una vez analizados los requerimientos se debe definir cual será el alcance real de la prueba, esto depende de los requerimientos iniciales, tiempo y limitaciones.	- Líder de Proyecto - Evaluador	Propuesta del alcance de la prueba	Planilla de solicitud de prueba.
7-Definir el plan de pruebas	En esta actividad se definen y planifican todas las actividades	- Evaluador - Líder de Proyecto.	Plan detallado de la prueba	Diagrama de Gant del Cronograma de actividades.

EPS-OO				
Actividad	**Descripción**	**Responsable**	**Producto Obtenido**	**Documento/ Entregable**
	que se deben realizar, estimando los tiempos y prioridad de la ejecución de las mismas, a través de un diagrama de Gant.			
8-Ponderación de las características de calidad	Se realiza una ponderación a nivel de Subcaracterísticas de calidad, para considerar la importancia dada por el solicitante a las mismas, y así establecer prioridad de una sobre otras al momento de evaluarlas.	- Líder de Proyecto	Ponderación de las subcaracterísticas.	Planilla de ponderación (ver *Apéndice G*).
9-Establecimiento del ambiente de prueba	Se establece el ambiente de prueba para que el evaluador pueda dar inicio al proceso.	- Líder de Proyecto.	N/A	N/A
10-Ejecución de las pruebas	Las pruebas se realizan por etapas a través de la aplicación de listas de verificación implementadas en el programa EPS-OO-Checklist el cual genera de forma automática los resultados de las preguntas. Para dar respuesta a algunas preguntas se deben procesar y registrar casos de prueba. Otras preguntas necesitan ser respondidas por los usuarios finales para lo cual se aplica un cuestionario a los mismos.	-Evaluador - Usuarios Finales. - Desarrollador	N/A	- Planillas de Casos de Prueba (ver *Apéndice F*). - Cuestionario realizado (ver *Apéndice I*). - Listado de fallas encontradas (ver *Apéndice J*). - Reporte de resultados del EPS-OO CheckList. - Análisis de resultados.
11-Culminación de las pruebas	El criterio de culminación es completar las listas de verificación para cada etapa del proceso de prueba, tomando en cuenta el tiempo previsto para las pruebas.	- Líder del Proyecto. - Evaluador.	N/A	Acta de culminación de pruebas (ver *Apéndice K*).

EPS-OO				
Actividad	Descripción	Responsable	Producto Obtenido	Documento/ Entregable
	Al culminar el proceso de prueba se debe concretar la finalización de las mismas a través del Acta de Culminación.			

Tabla N° 8: Estrategia de Prueba EPS-OO. Fuente: Elaboración propia

V.2.-Análisis del contexto

Al analizar el contexto se pudo determinar la influencia que tiene la Estrategia de Prueba en el ambiente donde se está desarrollando y donde será aplicada. Este análisis permitió identificar las condiciones favorables y no favorables que tiene la Estrategia de Prueba en los diferentes métodos de evaluación presentados por DESMET, con el fin de seleccionar el más idóneo para la evaluación de la misma. El detalle de estos resultados se muestran en el *Apéndice L.*

V.3.- Aplicación del Método DESMET

Una vez realizado el análisis del contexto se procedió a la aplicación del Método DESMET, el cual arrojó como resultado que el mejor método para realizar la evaluación de la Estrategia de Prueba es **Análisis de Características-Estudio de Caso.** El *Apéndice M* muestra la aplicación del método DESMET y cómo se llegó a este resultado.

V.4.- Evaluación de la Estrategia de Prueba

La evaluación, a través del método **Análisis de Características-Estudio de Caso**, incluye los siguientes pasos: Alcance de la Evaluación, Bases de la evaluación, Roles y responsabilidades, Procedimientos de la evaluación, Limitaciones, Tiempo requerido y Esfuerzo. Los resultados obtenidos en estos pasos son:

V.4.1.- Alcance y Bases de la Evaluación: Como alcance de la evaluación se tiene la EPS-OO, tomando cómo bases para la evaluación las pautas impuestas por la Empresa donde opera el software que la Estrategia evaluará. Para el caso del presente trabajo la Empresa es QUIMBIOTEC C.A.

V.4.2.- Roles y Responsabilidades: Los roles identificados fueron los siguientes:

- **El responsable**: El Gerente de Sistemas de QUIMBIOTEC, quien provee todos los recursos necesarios para llevar a cabo la evaluación.

- **El evaluador:** Se trata del grupo responsable que llevó adelante esta investigación es decir las autoras del presente trabajo.

- **Usuarios de la tecnología:** Se conformó un grupo mixto compuesto por los diferentes usuarios del software sometido a Prueba (Quimbiotec) y los evaluadores, es decir las autoras del presente trabajo.

V.4.3.- Procedimiento de la Evaluación: Para la evaluación de la Estrategia de Prueba, se tomo como proyecto piloto el **Sistema de Comercialización** de la empresa **QUIMBIOTEC**. Este sistema es el encargado del control de la comercialización de los

productos que en ella se elaboran. La descripción de la Empresa y del Sistema se encuentra detallada en el *Anexo B*.

El procedimiento de la evaluación se basa en los siguientes pasos: creación de las características de evaluación; aplicación de la Estrategia de Prueba al Sistema de Comercialización de QUIMBIOTEC y evaluación de las características. A continuación se explican los resultados obtenidos en cada uno de estos pasos.

V.4.3.1.- Creación de las Características de Evaluación: A continuación se presentan las Características y Subcaracterísticas que se evaluarán en la Estrategia de Prueba. Estas son:

Estructura: Alcance, Documentación, Enfoque y Resultados

Confort: Comprensivo y Fácil.

Efectividad: Flexibilidad, Adecuación y Utilidad

La forma de evaluar cada una de las Características es respondiendo las preguntas asociadas a cada Subcaracterística, utilizando una escala del uno (1) al cinco (5), donde el 1 representa el valor menos significativo y el 5 el más significativo, dependiendo del cumplimiento de la Característica en la EPS-OO. En el *Apéndice N*, se muestra detalladamente la tabla que especifica todas estas consideraciones de evaluación.

V.4.3.2.-Aplicación de la Estrategia de Prueba al Sistema de Comercialización: Una vez definidas las características, que servirán como pauta para la evaluación de la Estrategia de Prueba, se procede a la aplicación de la Estrategia de Prueba en el sistema seleccionado como proyecto piloto: **Sistema de Comercialización de QUIMBIOTEC**. Esto conlleva a determinar en que nivel de aceptación se encuentra la Estrategia de Prueba respecto a las Características planteadas.

La figura Nº 8 muestra los resultados obtenidos al aplicar la EPS-OO al sistema de QUIMBIOTEC, donde las barras representan los porcentajes de presencia de las Características de Calidad en cada una de las etapas y los sectores de distribución representan el porcentaje promedio de la presencia de las Características en el sistema. Se puede decir que el sistema presenta un alto nivel de calidad. Los resultados de la aplicación de la EPS-OO al sistema evaluado se muestran en el *Apéndice O*.

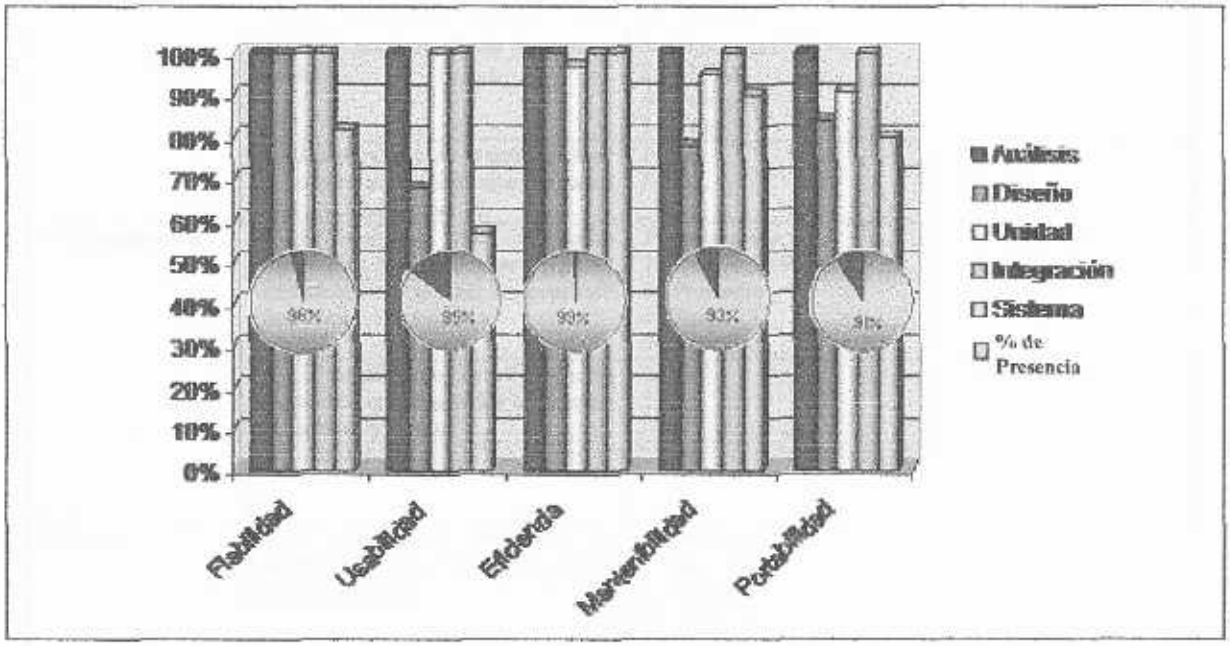

Figura Nº 8. Resultados del Sistema de Comercialización de QUIMBIOTEC.

V.4.3.3.-Evaluación de las Características: Una vez aplicada la Estrategia de Prueba al caso de estudio, se cuantificaron las características de evaluación, por medio de la selección de uno de los cinco niveles de evaluación por cada respuesta a la pregunta planteada. Las siguientes tablas muestran los resultados obtenidos en la evaluación.

Característica: Estructura

Característica	Preguntas	Evaluación	Resultado
Estructura: La estrategia debe estar creada con una buena estructura. Esta característica se divide en las siguientes subcaracterísticas:			
Alcance	1.-¿Está especificado el alcance de la prueba?.	1= No está especificado 2 = No está claramente especificado 3 = medianamente especificado 4 = Está especificado 5= Está claramente especificado	5
	2.-¿Se tienen claramente identificadas las condiciones a probar?	1= Ninguna 2= Muy pocas 3= Algunas 4= Casi todas 5= Todas	5
	3.-¿Se encuentran explícitos los tipos de pruebas que se llevarán a cabo?	1= Ninguna 2= Muy pocas 3= Algunas 4= Casi todas 5= Todas	5
	4.-¿La estrategia de prueba incluye revisiones desde bajo nivel hasta un alto nivel del software?	1= No 5= Si	5
Documentación	1.-¿El diseño de casos de prueba incluye los datos del software que se utilizarán para realizar las pruebas? Son los adecuados?	1= Inaceptable 2= Debajo del Promedio 3= Promedio 4= Bueno 5= Excelente	5
	2.-¿El diseño de los casos de prueba especifica qué resultados se esperan al realizar la prueba?	1= No 5= Si	5
Enfoque	1.-¿El enfoque de los casos de prueba fue el más adecuado para el tipo de software?	1= Ninguno 2= Muy pocos 3= Algunos 4= Casi todos 5= Todos	5
	2.-¿El diseño de los casos de prueba es lo suficientemente específico?	1= No está especificado 2 = No está claramente especificado 3 = medianamente especificado 4 = Está especificado 5= Está claramente especificado	5
Resultados	1.-¿Los resultados de la evaluación identifican claramente donde fueron encontrados los errores?	1= No 5= Si	5
	2.-¿La estrategia de prueba considera la generación de reportes de la evaluación?	1= No 5= Si	5

Tabla N° 9. Evaluación de la Estructura en la EPS-OO. Fuente: Elaboración propia

Característica: Confort

Características	Objetivo	Preguntas	Evaluación
Confort: La estrategia de prueba debe ser lo más confortable posible para el evaluador, es decir, su aplicación debe ser fácil de comprender con el fin de obtener los resultados más óptimos.			
Comprensivo	1.-¿La estrategia de prueba es comprensible para la persona que la lleva a cabo?	1= Incomprensible 2= No es claramente comprensible 3= Medio comprensible 4= Comprensible 5= Muy comprensible	4
	2.-¿Se identifica claramente lo que se debe llevar a cabo en cada paso de la Estrategia?	1= No es identificable 2 = No es claramente identificable 3 = Medianamente identificable 4 = Identificable 5= Claramente identificable	5
Fácil	a.- La estrategia de prueba debe ser fácil de llevar a cabo por quien la ejecute.	1.-¿La estrategia es fácil de llevar a cabo?	4
		2.-¿Los pasos a seguir están claramente identificados?	5

Tabla Nº 10. Evaluación del Confort en la EPS-OO. Fuente: Elaboración propia

Característica: Efectividad

Características	Objetivo	Preguntas	Evaluación
Efectividad: La estrategia de prueba debe cumplir con ciertas características de efectividad, es decir, su aplicación debe ser lo más efectivamente posible para la Empresa a la cual se le está evaluando el software.			
Flexibilidad	1.-¿La aplicación de la estrategia se adaptó fácilmente a las características del software?	1= Inaceptable 2= Debajo del Promedio. 3= Promedio 4= Bueno 5= Excelente	5
	2.-¿La estrategia es flexible para adecuarse a cualquier arquitectura de software OO?	1= Ninguna 2= Muy pocas 3= Algunas 4= Casi todas 5= Todas	5
Adecuación	1.-¿Se tienen definidas las situaciones del sistema que se desean probar?	1= Ninguna 2= Muy pocas 3= Algunas 4= Casi todas 5= Todas	5
	2.-¿La estrategia de prueba determina las prioridades que tiene el cliente de lo que se está probando?	1= No 5= Si	5
	1.¿La estrategia de prueba incluye la aplicación de casos de prueba?.	1= No 5= Si	5
	2.-¿Los casos de prueba son específicos para cada condición que se quiere probar?	1= No 5= Si	4

Características	Objetivo	Preguntas	Evaluación
	3.-¿El diseño de los casos de prueba se adecuan para probar cualquier condición en un software OO?	1= Inaceptable 2= Debajo del Promedio. 3= Promedio 4= Bueno 5= Excelente	5
Utilidad	1.-¿Los resultados obtenidos proporcionan información útil para la empresa?	1= Inaceptable 2= Debajo del Promedio. 3= Promedio 4= Bueno 5= Excelente	5
	2.-¿La empresa está conforme con los resultados obtenidos?	1= Inaceptable 2= Debajo del Promedio. 3= Promedio 4= Bueno 5= Excelente	5
	3.-¿La aplicación de la estrategia es útil para alcanzar un software de alta calidad?	1= No 5= Si	5

Tabla N° 11. Evaluación de la Efectividad en la EPS-OO. Fuente: Elaboración propia

V.4.4.- Limitaciones, Tiempo requerido y Esfuerzo: Es importante resaltar que al realizar la aplicación de la Estrategia de Prueba se comprobó que ésta no presenta mayores limitaciones, puesto que los recursos necesarios para aplicarla son de fácil obtención. Asimismo, el tiempo requerido para aplicar la Estrategia de Prueba es adecuado, ya que sólo se necesitó de 2 meses para su aplicación siguiendo un cronograma sencillo que contempló todas las actividades.

V.5.-Análisis de resultados de la evaluación de las Características

Para analizar los resultados de la evaluación de la Estrategia de Prueba se calculó el promedio aritmético de los valores obtenidos por cada Característica (Estructura, Confort y Efectividad) a partir del promedio de los porcentajes obtenidos por cada Subcaracterística. Se obtuvieron los siguientes resultados:

Estructura

Subcaracterística	Porcentaje
Alcance	100%
Documentación	100%

Enfoque	100%
Resultados	100%
Porcentaje promedio	**100%**

Confort

Subcaracterística	**Porcentaje**
Comprensivo	90%
Fácil	90%
Porcentaje promedio	**90%**

Efectividad

Subcaracterística	**Porcentaje**
Flexible	100%
Adecuación	95%
Utilidad	100%
Porcentaje promedio	**98%**

Porcentaje Promedio de la evaluación = 96%

De acuerdo a los resultados el **Confort** es la Característica que presentó menor porcentaje, por lo cual fue importante analizar este aspecto en la **Estrategia de Prueba**, pudiéndose observar que esta Característica es altamente dependiente de la capacitación del evaluador en cuanto a la programación OO, por lo tanto no tiene un impacto que se traduzca en una modificación de la Estrategia. Sin embargo, al observar los resultados de las otras dos características (Efectividad y Estructura), se puede afirmar que; al utilizar la **EPS-OO** en el proyecto piloto (Sistema de comercialización de QUIMBIOTEC), ésta resulto efectiva y aplicable.

V.6- Refinación de la Estrategia de Prueba

En cuanto a la Refinación, ésta se aplicó en la medida en que la Estrategia de Prueba se aplicaba al caso de estudio incorporando modificaciones, tanto en las Listas de Verificación como en la secuencia de los pasos de la Estrategia, que llevaron al mejoramiento de la misma, en cuanto a la precisión y efectividad de su aplicación.

Capítulo VI. Conclusiones

Como resultado de la investigación, surge una serie de conclusiones que se han subdividido en los tres aspectos más resaltantes: Las Pruebas y la Calidad, la Estrategia de Prueba como producto de la investigación y por último el Caso de Estudio.

En base a las Pruebas y la Calidad:

- Destaca la gran diversidad de criterios que existe entre los diferentes autores que dominan el tema sobre la **Prueba OO** y la **Calidad del Software**. En el presente trabajo de investigación se unificaron los términos para tener una base sobre la cual establecer los conceptos manejados durante la investigación.
- Para probar adecuadamente un **Software OO**, la definición de la **Prueba** debe ampliarse para incluir las técnicas de detección de errores aplicados a los modelos de análisis y diseño OO. La estrategia para la prueba de unidad e integración deben cambiar significativamente. El diseño de **Casos de Prueba** debe tener en cuenta las características propias del **Software OO**.
- La aplicación de una estrategia de prueba debe realizarse antes de que el Software sea entregado al usuario final, es decir, durante su desarrollo, con el fin de ir corrigiendo errores en cada etapa. Esta acción reducirá el costo de detección de un error y el trabajo en las últimas etapas y, como consecuencia, se tendrá un producto de Calidad.

En base a la Estrategia de Prueba:

- La EPS-OO, resultó ser una herramienta aplicable y de gran utilidad, en el sentido de que cumple con el objetivo principal de un proceso de pruebas de encontrar fallas

en el Software. Esta herramienta proporciona retroalimentación útil al programador durante el desarrollo del software lo que le permite irlo depurando. Esto conduce a un Software de alta Calidad.

- La EPS-OO permite detectar áreas débiles dentro del sistema en cuanto a los RNF (Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad). Con lo cual se puede llegar a un equilibrio entre lo que el cliente quiere y los resultados obtenidos de la **Estrategia de Prueba**.
- La aplicación de la EPS-OO ayuda a llevar un seguimiento de las Características de Calidad en las diferentes etapas del proceso de desarrollo de software.
- Para aplicar la EPS-OO el evaluador debe tener conocimientos previos en Software OO debido a que la EPS-OO maneja conceptos y terminologías propias de éste enfoque de programación.
- La característica Confort no se tradujo en una modificación de la Estrategia, a pesar de haber obtenido los valores mas bajos en la evaluación, ya que depende de los conocimientos y dominio de la programación OO por parte del evaluador.

En base al caso de estudio:

- Con la aplicación de la **Estrategia de Prueba** al Sistema **de Comercialización de QUIMBIOTEC**, se pudo detectar la existencia de fallas, mayormente en la Usabilidad, identificándolas claramente y pudiendo comunicárselo al desarrollador.
- Las Características de Calidad en el Sistema de Comercialización de QUIMBIOTEC obtuvieron valores promedios por encima del 75%, por lo cual se puede concluir que, en general, el mismo presenta las Características de Calidad tratadas en el presente trabajo.

Capítulo VII. Recomendaciones

Como resultado de la investigación y en base a las conclusiones obtenidas, se sugieren una serie de recomendaciones que ayudarán en investigaciones futuras referentes al tema del presente trabajo de investigación.

- Se recomienda realizar mayor número de iteraciones en la aplicación de la EPS-OO, con la finalidad de obtener una evaluación más precisa de la Estrategia en cuanto a las tres características evaluadas en la misma: Estructura, Confort y Efectividad. Con lo cual refinar la Estrategia de Prueba.
- La Estrategia de Prueba definida debe ser automatizada completamente, para lo cual se puede considerar ampliar el programa EPS-OO checklist.
- Se recomienda aplicar la estrategia de prueba en cada etapa del proceso de desarrollo del software para evitar la falta de seguimiento de alguno de los RNF planteados en el presente trabajo de investigación: Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad.
- Se recomienda que el evaluador tenga conocimientos previos del Sistema a evaluar y del paradigma OO para obtener resultados más precisos y confiables.
- Se recomienda evaluar sólo aquellas Características de Calidad más importantes para el buen desempeño del Sistema, ya que siempre prevalecen unas sobre otras dependiendo del tipo de software que se esta probando y de los intereses del cliente.

Bibliografía

(Bashir & Goel, 1999). Bashir, I. y Goel A. *Testing object-oriented software. Life cycle solutions.* Springer-Verlag New York , Inc.1999.

(Baskerville, 1999). Baskerville, R. *Investigating Information Systems with action research. Comunications of the AIS* . Vol 2. Art. 19. Octubre 1999.

(Bass et al., 1998). Bass, L., Clements, P. & Kazman, R. (1998). *Software Architecture in Practice*. Third Edition. Addison Wesley.

(Binder, 1996). Binder, Roger. *Testing object-oriented software: a survey*. Software testing,Verification and Reliability.

(Bosch, 2000). *Design et Use of Software Architectures*. 2000.

(Cardoso, 2001). Cardoso, Rodrigo. *Pruebas del Software.* Mérida, Venezuela 2001.

(Checkland, 1993).Ckeckland, P. *Pensamiento de Sistema, prácticos de Sistema*. Grupo Noriega. Editores. 1993.

(Dromey, 1996). Dromey Geoff. *Cornering the Chimera*.1996. IEEE. 1996 Pag 33-43.

(Gonzales, 2002). Gonzales, J. *Las normas de la calidad del software*. Addison-Wesley. Iberoamericana. España 2002.

(Esrivill, 1994). Estivill Castro V. *Calidad total en informática.* Laboratorio Nacional de Informática Avanzada A.C. México 1994.

http://www.lania.mx/spanish/actividades/newsletters/1994-otono/art2.html

(ISO, 1998). ISO/IEC FCD 9126. *Information Technology - Software Product Quality - Part 1: Quality Model*. Junio, 1998.

(ISO 9126, 1991). *Information Technology - Software Product Evaluation*. 1991.

(IEEE,1990). IEEE Standard Glossary of Software Engineering Terminology. 1990.

(Kitchenman, 1996). Kitchenman, B. *Evaluation Software Engineering Methods and tools*. Part1: The evaluation context and Evaluation Methods. SIGSOFT. Notes. 1996.

(Kruchten, 2000). Kruchten, P. (2000). *The Rational Unifed Process As Introduction*. 2nd Edition. Addison – Wesley.

(Mc Gregor & Korson, 1994). McGregror, J.D. y Korson, T. *Integrated object-oriented testing and development processes*. CACM 37:9, pp 59-77. 1994.

(Martínez, 2001). Martínez, J. (2001). *MOSCA Modelo Sistémico de Calidad (Integración del Modelo de Calidad de Software y el Modelo de Calidad del Proceso de Desarrollo con un Enfoque Sistémico)*.

(Monsalve, 2000). Monsalve, Luis. *Calidad de los productos del software*. 2000.

(Murphy 1994). Murphy, G.C., Townsend P., y Wong P.S. *Experiences with cluster and class testing*. CACM 37:9, pp 39-47. 1994.

(Overbeck, 1994). Overbeck, J. 1994: *Integration testing for object-oriented software*, Ph.D. Dissertation, Vienna University of technology, Viena, Austria. 1994.

(Pfleger, 1997) Pfleeger Lawrence Shari, Norman E. Fenton. *PWS Publishing Company*. 1997.

(Pfleeger, 1998). Shari Lawrence Pfleeger. *Software Engineering*. 1998.

(Pressman, 2002). Pressman, Roger. *Ingeniería del Software: Un enfoque Práctico*. McGraw Hill, 2002.

(Sommerville, 2002). Ian Sommerville. *Software Engineering*. 2002.

(Reo, 2002). Reo, David, *CMM for the Right Reasons*. ESI 2002.

(Wiegers,1999). Karl E.Wiegers. *Software Requirements*. 1999.

Glosario de términos

- **Calidad**: Es el conjunto de características de una organización, proceso o producto que le confieren la aptitud para satisfacer necesidades explícitas e implícitas. La calidad hace que un producto o servicio deba, en todos los aspectos, cumplir con el uso específico para el cual fue creado.

- **Calidad de Software**: Es la concordancia entre los requerimientos funcionales, el rendimiento explícitamente establecido, los estándares de desarrollo explícitamente documentados y las características implícitas que se espera de todo software desarrollado profesionalmente.

- **Características del software**: Son aquellas propiedades que identifican un producto software.

- **Características de Calidad**: Son aquellas características del software cruciales para el desarrollo con éxito del mismo, en cuanto a calidad. Estas características pueden ser medibles.

- **Caso de Prueba**: Un Caso de Prueba es aquel que identifica ambiente de operación de prueba, determinando los valores de entradas, de salidas y condiciones que delimitan dicho ambiente.

- **DESMET**: Es un método para la evaluación de métodos y/o herramientas de Ingeniería del Software.

- **Eficiencia**: Es la capacidad del producto de software para proveer un rendimiento apropiado, relativo a la cantidad de recursos utilizados, bajo condiciones específicas.

- **EPS-OO:** Nombre dado a la Estrategia de Prueba para Software OO propuesta en el presente Trabajo de Investigación.

- **EPS-OO CheckList**: Programa utilizado para la ejecución de las pruebas de la EPS-OO, cuya función principal es la automatización de las listas de chequeo.

- **Estrategia de Prueba**: Es aquella que integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba.

- **Fiabilidad**: Es la capacidad del producto de software para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones especificas.

- **Listas de Verificación**: Son simples formulario de preguntas las cuales dependen del objetivo para el cual son usadas.

- **Mantenibilidad**: Es la capacidad del software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software ante cambios del ambiente, en requerimientos y especificaciones funcionales.

- **Metodología Investigación Acción**: Es una Metodología que utiliza esquemas de investigación menos rígidos y ciclicos, que convergen progresivamente en un resultado esperado.

- **MOSCA**: Modelo de Calidad Sistémico que integra las ideas más significativas del estándar ISO 9126, del Modelo Dromey y del enfoque Sistémico de la Calidad propuesto por Callaos.

- **Orientado a Objetos (OO)**: Es un enfoque de programación basado en el comportamiento del objeto. El software que implanta el objeto contiene estructuras de datos y operaciones que expresan dicho comportamiento. La representación en software OO del objeto es entonces una colección de tipos de datos y objetos.

- **Portabilidad**: La portabilidad es la capacidad del producto de software para ser transferido de un ambiente a otro.

- **Pruebas**: Es una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente con la finalidad de encontrar fallas en el mismo.

- **Pruebas de Clústers**: Son pruebas de los grupos de clases que actúan en combinación para proveer un conjunto de servicios.

- **QUIMBIOTEC**: Empresa productora de derivados sanguíneos.

- **Requerimientos del software:** Son una descripción de los servicios que el sistema debería proporcionar al cliente y las limitaciones bajo las cuales éste debería operar.

- **Requerimientos Funcionales**: Son los requerimientos que describen lo que el sistema debe hacer.

- **Requerimientos No Funcionales (RNF)**: Son aquellos requerimientos que definen las restricciones y propiedades de un sistema.

- **Software**: Conjunto de programas que les son instalados a los equipos de computación para que funcionen y puedan proveer determinados servicios.

- **Subcaracterísticas de Calidad**: Son las características relacionadas con cada una de las Características de Calidad.

- **Técnicas de prueba**: Son un conjunto de procedimientos utilizados para dirigir las pruebas.

- **Usabilidad**: Es la capacidad del producto software para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones especificas.