# Propuesta de manual de procedimiento para pruebas de sistema

***Propose for a manual of procedures for testing system***

**Yuniet del Carmen Toll Palma, Yohandri Ril Gil**

Universidad de las Ciencias Informáticas

{ytoll@uci.cu, rilltt}@uci.cu

**Resumen**

La Gestión de la Calidad de Software es un pilar fundamental a lo largo de la vida de un producto, teniendo como propósito entender las expectativas del cliente en términos de calidad, y poner en práctica un plan proactivo para satisfacer esas expectativas. EL proceso de pruebas, y específicamente, la aplicación de pruebas de sistema son las encargadas de comprobar el buen funcionamiento y la calidad del software, tanto de los requisitos funcionales como de los requisitos no funcionales. Existen varios tipos de pruebas de sistema que se aplican a un software, tales como: pruebas de seguridad, pruebas de rendimiento, pruebas de resistencia, pruebas de vulnerabilidad, pruebas de compatibilidad, etc. Para controlar este proceso se hace necesaria la utilización de guías, que apoyadas en estas pruebas es posible definir una estrategia tanto para pruebas funcionales con los casos de pruebas correspondientes según las especificaciones de los requisitos funcionales y para pruebas no funcionales, que contiene los requisitos no funcionales descritos en la aplicación y en caso de la existencia de un requisito particular se confecciona un caso de prueba para ese tipo de prueba. Por lo que en la investigación se propone un manual de procedimiento o estrategia para realizar pruebas de sistema, logrando con el mismo ganar en organización, experiencia, madurez, rapidez, habilidad y control durante el proceso de pruebas de sistema.

**Palabras clave:** Calidad, casos de prueba, estrategia para pruebas funcionales, estrategia para pruebas no funcionales, manual de procedimiento, prueba, prueba de sistema, requisitos funcionales, requisitos no funcionales.

***Abstract***

*The Quality Management Software is a key pillar along the life of a product, with the purpose to understand customer expectations in terms of quality, and implement a proactive plan to meet those expectations. The testing process and specifically, the implementation of system tests are responsible for checking the proper functioning and quality of the software, both functional requirements and non-functional requirements. There are several types of system tests that apply to software, such as safety tests, performance tests, tests of strength, vulnerability testing, compatibility tests, etc. To control this process requires the use of guides, which supported this evidence it is possible to define a strategy for both functional tests with cases of evidence according to the specifications of the functional requirements and tests for non-functional, which contains non-functional requirements described in the application and if the existence of a particular requirement is up a test case for such proof. As far as the investigation suggests a procedural manual or strategy for testing system, achieving a win with the same organization, experience, maturity, speed, skill and control during the process of testing system.*

***Key words****: Functional requirements, manual of procedure, non-functional requirements, quality test, strategy for non-functional tests, strategy for tests, test cases, test system.*

**Introducción**

Para garantizar y controlar la calidad durante el proceso de las pruebas de sistema, es necesario contar con una serie de pasos que puedan servir de guía en el momento de llevar a cabo dichas pruebas de forma organizada. Debido, al volumen de las pruebas como a la complejidad de las mismas, que van desde entrar grandes cantidades de datos, verificar la funcionalidad del sistema, hasta verificar la rapidez o el tiempo de respuestas de las aplicaciones, surge la necesidad de tener un manual de procedimiento para planificar, diseñar, ejecutar y evaluar eficientemente estas pruebas, permitiendo el control y seguimiento durante esta etapa. El objetivo fundamental es definir un manual de procedimiento para realizar Pruebas de Sistema en un software.

Las Pruebas de Sistema, se realizan después de la construcción del sistema. Las mismas prueban a fondo el sistema, comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción.

**Estrategia de prueba de software**

El proceso de ejecución de Pruebas de sistema debe ser considerado durante todo el ciclo de vida de un Proyecto, para así obtener un producto de alta calidad. Su éxito dependerá del seguimiento de una Estrategia de Prueba adecuada [1].

Una estrategia de prueba integra técnicas de casos de prueba en una serie de pasos planeados y bien definidos que llevan a la construcción exitosa del software. Se describen los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar estos pasos, y cuántos recursos se requieren [2].

**Descripción de la estrategia de prueba de software**

Cualquier estrategia de prueba debe incorporar la planeación de las pruebas, el diseño de casos de prueba, la ejecución de pruebas y evaluación de las pruebas, es decir, los datos resultados finales [2].

Una estrategia de prueba debe ser lo suficientemente flexible para promover la creatividad y la adecuación necesaria a los sistemas de software, y lo suficientemente rígida para promover una planeación razonable y un seguimiento administrativo a medida que el proyecto progresa[2].

La estrategia de pruebas debe estar dirigida a encontrar los errores más importantes con la mayor rapidez y el menor costo posible. Al involucrar más pruebas y medidas de detección, se puede definir mejor la estrategia. Las duplicidades u omisiones involuntarias que tengan lugar entre diferentes pruebas pueden ser evitadas coordinando a los probadores y las actividades de pruebas, así como fijando el alcance de la prueba [3].

Existen una serie de deficiencias al realizar las Pruebas de sistema sin estrategia, tales como: la falta de motivación, lo cual hace que las pruebas se tornen incómodas, aburridas; también está el inconveniente que se produce al probar todo al mismo tiempo. Todo lo anterior, acompañado de los fallos que puedan ocurrir por otras partes, hacen que se vuelva más costoso corregir un error. Es más difícil diagnosticar las causas de fallos y por tanto el resultado no sería otro que un producto defectuoso [3].

**Estrategia de Prueba Funcional (requisitos funcionales)**

**Técnicas de diseño de pruebas**

Las técnicas de Diseño de Pruebas están dadas por la imposibilidad de realizar la mayor cantidad de Prueba, pues se deben seguir determinados criterios para seleccionar los casos de prueba. Es por eso que el objetivo de las técnicas para el Diseño de Pruebas es garantizar con el mayor grado de confianza posible la detección de los defectos del software [4].

Los Casos de Prueba especifican una forma de probar el sistema, incluyendo los datos de entrada y resultados esperados. Un Caso de Prueba puede derivarse de un Caso de Uso del Modelo de Casos de Uso o de una realización de un Caso de Uso del Modelo de Diseño [5].

En la práctica lo que se prueba puede estar dado por un requisito o varios requisitos del sistema cuya implementación justifica una prueba que es posible realizar y resulta poco costosa.

Las pruebas funcionales requieren el uso y la aplicación de una técnica de caja negra, siguiendo esta técnica se confeccionan los casos de pruebas para probar las condiciones de entrada tanto válidas como inválidas. Existen tres enfoques principales para el diseño de casos de prueba, ellos son [6]:

1. El enfoque estructural o de caja blanca: este tipo de enfoque es para realizar las pruebas de unidad las cuales se le realizan al código del software.

2. El enfoque funcional o de caja negra: en este enfoque funcional las pruebas están centradas en los requisitos funcionales del software y para elegir bien un Caso de Prueba se debe tener en cuenta los siguientes requisitos:

- ✓ Que cubra un conjunto extenso de otros casos posibles.
- ✓ Indique algo acerca de la ausencia o la presencia de defectos en el conjunto específico de entradas que prueba, así como de otros conjuntos similares.
- ✓ Reduzca el número de otros casos necesarios para que la prueba sea razonable.
- ✓ Implique que el caso ejecute el máximo número de posibilidades de entrada diferentes para así reducir el total de casos.

3. El enfoque aleatorio consiste en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.

**Estrategia de Prueba no Funcional (requisitos no funcionales)**

Una vez establecidos los pasos a seguir durante el proceso de prueba, se procede a asociar las técnicas de pruebas de software que garantizarán los Requisitos No Funcionales (RNF). Para identificar los RNF se partió del Modelo de Calidad del Sistema. Este modelo permite identificar las características de calidad que deben ser evaluadas en un software. Estas características tienen a su vez subcaracterísticas asociadas. Las características que se tomaron en cuenta fueron: Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad. Las mismas tienen una serie de atributos que responden a cada una de las características [1].

**Desarrollo**

**Estrategia de Pruebas de Sistema**

La Estrategia de Pruebas de Software propuesta está compuesta por las siguientes actividades:

- ✓ Planificación de las pruebas.
- ✓ Diseño de las pruebas.
- ✓ Ejecución de las pruebas.
- ✓ Evaluación de las pruebas.

**Planificación de las pruebas**

La planificación de las pruebas se realiza con antelación para lograr organización durante su ejecución, para evitar pérdida de tiempo, confusiones, que los probadores se aburran, y no le den la seriedad que lleva la actividad, para poner en práctica todos los pasos, es preciso realizar las pruebas sin perder un solo detalle del proceso.

**Definir los tipos de pruebas que se le aplicarán al software, explicando bien sus objetivos y metas**

Es aconsejable que en la primera interacción del proceso de pruebas de sistema se realicen este grupo de pruebas: Funcionalidad, Seguridad, Disponibilidad y Red, Rendimiento o Carga, Compatibilidad, Resistencia o Stress, Usabilidad y Fiabilidad.

**1. Prueba de Funcionalidad**

**Objetivo**

Verificar la función del sistema al fijar la tensión en la validación de las funciones, métodos, servicios y casos de usos.

**Metas**

Validar que la aplicación:

- ✓ Cumpla con los requisitos funcionales especificados en el diseño de la solución.
- ✓ Cumpla con los requisitos No funcionales especificados en el diseño de la solución.
- ✓ Cumpla con las restricciones de entrada y salida de la información especificada en el diccionario de Datos, de cada caso de uso.
- ✓ Cumpla íntegramente con la estructura referencial especificada en el Mapa de Navegación.

**2. Prueba de Seguridad**

**Objetivo**

Verificar que los mecanismos de protección incorporados en el sistema realmente lo protegerán de accesos impropios.

**Metas**

Validar que en la aplicación:

- ✓ Los datos y funciones del sistema solo pueden ser accesibles por los autores debidamente autorizados.
- ✓ Las funciones que atenten contra la integridad de los datos de negocios sean debidamente impedidas.

**3. Prueba de Disponibilidad y Red**

**Objetivo**

Verificar el comportamiento de la aplicación cambiando la infraestructura de red al aplicar diferentes configuraciones, o retardos.

**Metas**

Validar que en la aplicación:

- ✓ No se reduzca la disponibilidad de los sistemas, debido a la actividad de alguna persona o sistema, ya sea, accidental o malintencionado.

**4. Prueba de Rendimiento o Carga**

**Objetivo**

Verificar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado y así como la verificación de la capacidad del sistema para manejar volúmenes de datos extremos de acuerdo a la velocidad que se especifique para el sistema.

**Metas**

Validar en la aplicación:

- ✓ Comprobar los tiempos de respuesta del sistema en una cantidad limitada de escenarios de trabajo (a nivel de número de usuarios y número de transacciones), bajo una configuración de hardware y software constante.
- ✓ Comprobar el tiempo de respuesta al realizar una función.
- ✓ Comprobar el tiempo de respuesta al realizar accesos concurrentes a una determinada información.
- ✓ Atender múltiples solicitudes de parte de los actores que acceden a un mismo recurso.

**5. Prueba de Compatibilidad**

**Objetivo**

Verificar el funcionamiento del sistema sobre diferentes componentes de software.

**Metas**

Validar en la aplicación:

- ✓ Sistemas Operativos.
- ✓ Navegadores.

**6. Prueba de Resistencia o Stress**

**Objetivo**

Verificar cómo se comporta el sistema bajo condiciones anormales:

**Metas**

Validar en la aplicación:

- ✓ Carencia de sistemas externos con los que interactúa el sistema.
- ✓ Aplicar carga excesiva de trabajo al sistema (extremas sobrecarga).
- ✓ Hardware no disponible.
- ✓ Recursos compartidos no disponibles.
- ✓ Además verifica qué hace el sistema cuando el usuario no hace lo que supuestamente debe hacer.

**7. Prueba de Usabilidad**

**Objetivo**

Determinar si la organización de los contenidos y las funcionalidades que se ofrecen desde el Sitio Web son entendidas y utilizadas por los usuarios de manera simple y directa.

**Metas**

Validar en la aplicación:

- ✓ Para poder ver las páginas adecuadamente necesita utilizar un navegador compatible con estándares Web.
- ✓ Proporcionar al usuario información relacionada con el estado actual del sistema.

**8. Prueba de Fiabilidad**

**Objetivo**

Verificar la probabilidad de que ese sistema funcione o desarrolle una cierta función, bajo condiciones fijadas y durante un período de tiempo determinado.

**Meta**

Validar en la aplicación:

- ✓ El sistema deberá estar disponible todo el tiempo.
- ✓ El tiempo permitido para que el sistema puede estar fuera de operación después de un fallo no debe exceder 2 días.

**Diseño de las pruebas de Sistema**

Las pruebas de sistema se dividen en dos tipos: las pruebas funcionales y las pruebas no funcionales.

Las pruebas funcionales se encargan de verificar la funcionalidad del sistema basado en la especificación de casos de usos o requisitos funcionales.

Las pruebas no funcionales son las encargadas de verificar el rendimiento, el stress, configuración, seguridad, etc. del sistema, los cuales conforman los requisitos no funcionales específicos que debe cumplir una aplicación determinada.

Las pruebas funcionales requieren el uso y la aplicación de una técnica de caja negra, siguiendo esta técnica se confeccionan los casos de pruebas para probar las condiciones de entrada tanto válidas como inválidas.

Las pruebas no funcionales no es de obligatorio cumplimiento el uso del diseño de casos de prueba que contenga condiciones de entrada para clases válidas y clases inválidas, pues esto depende de la aparición o no de la descripción de uno o varios requisitos no funcionales y a la hora de efectuar las pruebas de este tipo sea preciso adicionarlas para verificar la calidad del sistema.

El diseño se realizará siguiendo dos estrategias o manuales, una Estrategia de Prueba Funcional y una Estrategia de Prueba No Funcional, las cuales se detallan en las siguientes secciones.

**Estrategia de Prueba Funcional (requisitos funcionales)**

El diseño de las pruebas se debe describir para cada uno de los tipos de pruebas de sistema con los siguientes datos:

**Nombre de la Prueba:** *[Nombre de la prueba que se aplicará ejemplo (stress, etc.)]*

Todo manual o estrategia de prueba debe llevar el nombre de la prueba que se le aplicó para poder identificar el tipo de error encontrado a la hora de recopilar en la tabla de no conformidades y así saber hacia donde se debe enfocar el trabajo para remediar el error.

**Tipo de Prueba:** *[Pruebas de Sistema]*

El tipo de prueba se refiere en este caso al nivel de prueba, es decir, si es prueba de unidad, si es prueba de integración, prueba de sistema o si es prueba de aceptación.

**Entrada:** *[Poner las condiciones de entradas según los casos de usos del sistema]*

En la entrada se explica el cómo se introducirán los datos, que aparecen en el diccionario de datos, los cuales son necesarios solamente para las pruebas de funcionalidad del sistema, es decir, para confección de los casos de pruebas que ayudarán a verificar la funcionalidad de la aplicación en cuestión, aunque para las pruebas no funcionales pueden hacerse casos de pruebas. Todo depende de la especificación de los requisitos no funcionales del sistema.

**Tabla 1. Descripción de las clases válidas y clases inválidas.**

| Condición de Entrada | Clases válidas | Clases inválidas | Resultado Obtenido | Resultado de la prueba | Observaciones |
|---|---|---|---|---|---|
| *[Es la condición especificada en el diccionario de datos del CU]* | [*Será la que cumpla con la especificación de un rango de valores para los datos de entrada*] [*Datos válidos*] | [*Será la que no cumpla con la especificación de un rango de valores para los datos de entrada*] [*Datos no válidos o erróneos*] | *[El resultado que devolvió el sistema como respuesta según la entrada]* | *[Satisfactorio o insatisfactorio]* | *[Cualquier sugerencia]* |

**Técnica:** *[Explicación de cómo obtener el caso de prueba]*

Del método de caja negra la técnica de particiones de equivalencia es la utilizada para diseñar los casos de prueba que se describen con el formato siguiente:

**Tabla 2. Descripción del Caso de Prueba.**

| | |
|---|---|
| **Tipo de Prueba** | *[Nombre del tipo de prueba de sistema: funcional]* |
| **Nombre del Caso de Uso (1…n)** | *[Nombre del CU que se probará]* |
| **Descripción:** | *[Describir brevemente los pasos a seguir para llegar al CU sometido a prueba ]* |
| **Nombre del Caso de Prueba (1….n)** | *[Nombre del CP y especificar el escenario donde se encuentra]*<br>*Ejemplo: El nombre del caso de prueba está dado según la cantidad de flujos tanto básicos como alternos.* ***Ejemplo CPR 1. Registrar consulta médica.*** |
| **Condición de Entrada:** | *[Descripción textual de lo que ocurra en el mundo real que hace necesario ejecutar el caso de prueba, precisando los datos de entrada y los comandos a dar por el actor. Descripción textual del estado de la información almacenada.]* |
| **Salida Esperada:** | *[Descripción textual del estado en que queda la información y las alertas que pueden generarse, una vez ejecutado el CU con los valores y el estado especificado en la entrada.]* |

Ejemplo de un Caso de Prueba del CU Administrar Actividades del Caso de Estudio Ficha Técnica.

**Tabla 3. Ejemplo de caso de prueba.**

| | |
|---|---|
| **Tipo de Prueba** | Funcional |
| **Nombre del Caso de Uso (1…n)** | Administrar Actividades |
| **Descripción:** | 1. El caso de uso inicia cuando el administrador desea administrar una actividad.<br>2. Muestra las siguientes opciones:<br>- Registrar datos de una actividad (sección 2a)<br>- Modificar datos de una actividad (sección 2b)<br>- Eliminar una actividad (sección 2c)<br>3. Selecciona la opción de su interés<br>4. Brinda el servicio solicitado.<br>5. El caso de uso termina. |
| **Nombre del Caso de Prueba (1….n)** | CPR 1. Administrar Actividades |
| **Condición de Entrada:** | Las actividades contienen:<br>– Nombre de la actividad: cadena de carácteres, no permite carácteres extraños: ( ) *, - +´´ ¨¨ ç *^; [ ] {¿? º ¡ @ |
| **Salida Esperada:** | Las actividades contienen:<br>– Nombre de la actividad: Reunión Proyecto |

**Procedimientos:** *[Cómo proceder en el momento de la prueba]*

**Paso 1.** *Montar el entorno del laboratorio.*

**Paso 2**. *Montar las herramientas necesarias para la aplicación de la prueba.*

**Paso 3.** *Distribuir cantidad de probadores por tipo de Prueba de Sistema a realizar al software.*

**Paso 4.** *Una vez montado el entorno de prueba con todos los probadores listos y preparados para realizar la prueba correspondiente en cualquier navegador y plataforma, se procede a la ejecución de la prueba en cuestión.*

**Paso 5.** *Recopilar en la tabla de no conformidades todos los errores encontrados, para luego corregirlos.*

**Salida o Resultado:** *[Plantilla para recoger los resultados]*

**Tabla 4. Descripción de la tabla de No Conformidades para pruebas funcionales.**

| | |
|---|---|
| **Tipo de Prueba** | *[Nombre de la prueba]* |
| **Nombre del Probador** | *[Nombre del probador]* |
| **Nº** | *[Número de la no conformidad]* |
| **Elementos** | *[Tipo de elemento, ejemplo: interfaz]* |
| **CU** | *[Nombre del CU correspondiente a la no conformidad encontrada]* |
| **No conformidad** | *[Describir la no conformidad encontrada]* |
| **Localización de la no Conformidad** | *[El escenario, sección o pantalla del CU asociado, donde se encontró]* |
| **Sugerencias** | *[Comentario que desee poner sobre la no conformidad encontrada]* |
| **Atributos asociados** | *[Especificar la subcaracterística asociada tanto para las no conformidades, como para las sugerencias, por separado. Ejemplo: Completitud (no conformidad), Corrección (sugerencia)]* |

**Estrategia de Prueba no Funcional (requisitos no funcionales)**

No es de obligatorio cumplimiento la elaboración de casos de pruebas para requisitos no funcionales, ya que los requisitos no funcionales de una aplicación determinada, valida el rendimiento, el stress, la seguridad, la disponibilidad y red, la fiabilidad, compatibilidad, entre otras que no requieren de una condición de entrada específica sino de explicar el procedimiento del cómo se va a llevar a cabo en la aplicación [7].

Al igual que puede suceder que en la especificación de los requisitos no funcionales para la aplicación no aparezca la descripción de uno o varios requisitos no funcionales y a la hora de efectuar las pruebas de este tipo sea preciso adicionarlas para verificar la calidad del sistema, aunque en ocasiones si el cliente desea medir alguna condición de entrada específica comprendido entre estos requisitos no funcionales se pueden elaborar casos de pruebas por ejemplo para la prueba de seguridad, que se miden una serie de parámetros.

El diseño de las pruebas se debe describir para cada uno de los tipos de pruebas de sistema con los siguientes datos:

**Nombre de la Prueba:** *[Nombre de la prueba que se aplicará ejemplo (stress, etc.)]*

Toda estrategia de prueba debe llevar el nombre de la prueba que se le aplicó para poder identificar el tipo de error encontrado a la hora de recopilar en la tabla de no conformidades y así saber hacia donde se debe enfocar el trabajo para remediar el error.

**Tipo de Prueba:** *[Pruebas de Sistema]*

El tipo de prueba se refiere en este caso al nivel de prueba, es decir, si es prueba de unidad, si es prueba de integración, prueba de sistema o si es prueba de aceptación.

**Entrada:** *[“Deber ser” para hacer cada prueba] Poner diccionario de datos*

En la entrada se explica el cómo se introducirán los datos, que aparecen en el diccionario de datos, los cuales son necesarios solamente para la realización de las pruebas de funcionalidad del sistema, es decir, para confección de los casos de pruebas que ayudarán a verificar la funcionalidad de la aplicación en cuestión, aunque para las pruebas no funcionales pueden hacerse casos de pruebas todo depende de la especificación de los requisitos no funcionales del sistema.

**Procedimientos:** *[Cómo proceder en el momento de la prueba]*

**Paso 1.** *Montar el entorno del laboratorio*

**Paso 2**. *Montar las herramientas necesarias para la aplicación de la prueba.*

**Paso 3.** *Distribuir cantidad de probadores por tipo de Prueba de Sistema a realizar al software.* **Ejemplo:** *Tres probadores para realizar pruebas funcionales, cuatro probadores para realizar pruebas de carga, etc.*

**Paso 4.** *Una vez montado el entorno de prueba con todos los probadores listos y preparados para realizar la prueba correspondiente en cualquier navegador y plataforma, se procede a la ejecución de la prueba en cuestión.*

**Paso 5.** *Recopilar en la tabla de no conformidades todos los errores encontrados, para luego corregirlos.*

**Salida o Resultado:** *[Plantillas para recoger los resultados]*

**Tabla 5. Descripción de de la tabla de No Conformidades para Pruebas no Funcionales.**

| | |
|---|---|
| **Tipo de Prueba** | *[Nombre de la prueba]* |
| **Nombre del Probador** | *[Nombre del probador]* |
| **Nº** | *[Número de la no conformidad]* |
| **Elementos** | *[Tipo de elemento, ejemplo: interfaz]* |
| **Nombre del Requisito No Funcional** | *[Nombre del requisito no funcional correspondiente a la no conformidad encontrada]* |
| **No conformidad** | *[Describir la no conformidad encontrada]* |
| **Localización de la no Conformidad** | *[El escenario, sección o pantalla del CU asociado, donde se encontró]* |
| **Sugerencias** | *[Comentario que desee poner sobre la no conformidad encontrada]* |
| **Atributos asociados** | *[Especificar la subcaracterística asociada tanto para las no conformidades, como para las sugerencias, por separado. Ejemplo: Completitud (no conformidad), Corrección (sugerencia)]* |

**Conclusiones**

Finalizando este trabajo se llegaron a las siguientes conclusiones:

- ✓ Se demostró la importancia de contar con un manual de procedimiento para asegurar la máxima organización en la planificación, diseño, ejecución y evaluación de los resultados de las pruebas de sistema, evitando que el software sea rechazado al final del ciclo de desarrollo.

- ✓ Se concluyó que la evaluación de las pruebas es un aspecto determinante para la producción, pues de esto depende medir hasta qué punto el software cumple con las especificaciones del cliente.
- ✓ Se estableció un manual que contiene un conjunto de pasos secuenciales que orientan cómo proceder durante la realización de las pruebas de sistema en un software.

**Glosario de términos**

**Calidad de Software:** conjunto de propiedades inherentes a un objeto que le confieren capacidad para satisfacer necesidades implícitas o explícitas.

**Caso de Prueba:** conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

**Caso de Uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

**CP:** Casos de Prueba.

**CU:** Casos de Uso

**Estrategia de Prueba de Sistema:** es el conjunto de pasos que indican cómo desarrollar de forma organizada las pruebas de sistema a un software o aplicación determinada.

**Manual de Procedimiento:** es el conjunto de pasos lógicos ordenados cronológicamente que indican cómo proceder en determinado proceso.

**Pruebas de Sistema:** son las que se le realizan al software una vez que se termina la fase de construcción.

**Ralentiza:** sinónimo de lentitud.

**Sistema de Software:** es el conjunto de programas que administra los recursos de hardware.

**Software:** es el conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

**Referencias Bibliográficas**

[1] M.P. Anna. C Grimán, Luis. E Mendoza. (2007), Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales.

[2] Pruebas. (2007), Desarrollo de Sistemas.

[3] M. Collado. (2003), Pruebas de software. Técnicas de prueba del software. Estrategias de prueba del software.

[4] F.Ó.G. Rubio, and C.B. Santos. (2007), Tema 7. Pruebas del Software Tema 9. Pruebas del Software.

[5] M. DeSoft. (2005), Conceptos de Disciplina de Pruebas.

[6] Pressman. (2006), Ingeniería de Software, un enfoque practico. DISEÑO DE CASOS DE PRUEBA.

[7] L.L. Castillo. (2007), Casos de Pruebas para Pruebas No Funcionales, DeSoft.