Universidad de Cienfuegos “Carlos Rafael Rodríguez”
Facultad de Informática
Carrera de Ingeniería Informática

Manual de Prueba para Ingenieros de software en la Facultad de Informática en la Universidad de Cienfuegos “Carlos Rafael Rodríguez”

Trabajo de Diploma para optar por el título de Ingeniería en Informática

Autora:
Andrisbel Ruiz Romero.

Tutoras:
Ing. Anabel Martín Aparicio.
MSc. Daimarelys Acevedo Cardoso.

Cienfuegos, Cuba
Curso 2009 - 2010

El testing puede probar la presencia de errores pero no la ausencia de ellos.

Edsger Dijkstra

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Informática de la Facultad de Informática en la Universidad de Cienfuegos “Carlos Rafael Rodríguez”, para que hagan el uso que estimen pertinente con el trabajo de diploma.

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

_____

Andrisbel Ruiz Romero

Los abajo firmantes certificamos que el presente trabajo ha sido revisado según acuerdo de la dirección de nuestro centro y el mismo cumple los requisitos que debe tener un trabajo de esta envergadura referente a la temática señalada.

---------- ----------

Firma Tutor Firma Tutor

---------- ----------

Firma ICT Firma Vicedecano

Agradecimientos.

A mis dos grandes tesoros, mis padres, que han depositado en mí todo su amor, dedicación y atención durante toda la etapa de mi vida, ¡Los quiero mucho!. Que el Señor los acompañe y los ilumine siempre.

A mis abuelos, por acompañarme siempre y estar a mi lado.

A mi novio, por depositar en mi toda su confianza y por apoyarme siempre, ¡te quiero mucho!

A mis tíos, Maritza, Israel y Gloria, por su preocupación, interés y además por ser mi brazo de apoyo en todos los momentos que los necesite.

A mi familia y en especial a mi prima Gladys, por su gran ayuda cuando la necesite.

A mis Tutoras Anabel y Daimarelys, por tener tanta paciencia y tratar de que sea una gran profesional.

A mis amigas en la UCI, Rismary y Aliuska, por los años compartidos y su gran interés en ayudarme cuando las necesite.

A mi gran amiga Annalié, por sus críticas constructivas que me ayudaron hacer mejor cada día, nunca te olvidaré.

A mis compañeros en la UCf, por su gran preocupación.

Esencialmente a todas aquellas personas que aportaron su granito de arena para que este proyecto se hiciera realidad.

Gracias por todo… Andrisbel

Dedicatoria.

A ti dios por regalarme una familia tan maravillosa y por darme la oportunidad de vivir.

A mis dos ángeles de la guarda, mis padres, por darme además de todo su amor y cariño, por depositar en mí toda su confianza, dedicación y sacrificio. ¡Los amo mucho!

A mi novio José Raúl por ser tan especial, Te quiero.

A mis abuelos, Gladys, Andrea e Israel por apoyarme siempre.

A mis tíos Maritza e Israel por haber estado conmigo toda mi vida.

A mi familia por confiar en mí en todo momento y brindarme siempre un gesto de cariño.

Andrisbel…

Resumen

El presente trabajo de investigación lleva por título: Manual de prueba para Ingeniero de software en la Facultad de Informática de la Universidad de Cienfuegos “Carlos Rafael Rodríguez”, donde centra su objeto de estudio en el proceso personal del software, pero más específicamente en el proceso personal de prueba.

Se realizó con la necesidad de evaluar el proceso de prueba durante todo el ciclo de vida del software para detectar en tiempo todos los fallos ocurridos en toda su etapa de ejecución, quedando como objetivo esencial establecer una guía para la aplicación de pruebas a cada uno de los proyectos que son elaborados en la Facultad de Informática y además lograr que el cliente quede complacido.

Como resultado más relevante se llegó a la confección de un manual que tiene como estrategia el estándar IEEE (1012-2004) del Proceso de Verificación y Validación, la cual será aplicada de forma paralela al ciclo de vida del software que sigue la metodología RUP, donde se definieron tareas y tipos de pruebas a realizar en cada una de las actividades y así poder tener una mejor organización y control sobre las mismas. Además, se propone un modelo de prueba para aplicar la estrategia propuesta, que no es más que una modificación del Modelo W, el cual centra su atención en el ciclo de vida del software antes de su puesta en producción.

Abstract

The title of this investigation paper is “Test for engenier is software in the Computing Faculty of Cienfuegos is University Carlos Rafael Rodriguez", it focus on the study object in the elaboration process of software and particularly in the test process of the software.

It was carried for the necessity of evaluating the test process during the whole cycle of life of the software to detect all the shortcomings happened in all their execution stage on time, being as essential objective to establish a guide for the application of tests to each one of the projects that have been elaborated in the Computing Faculty and also to solve the client necessities.

The most relevant result achieved was the elaboration of a manual that has as strategy the standard IEEE (1012-2004) of the Process of Verification and Validation, which will be applied from a parallel way to the cycle of life of the software that follows the methodology RUP, where were defined tasks and types of tests to carry out in each one of the activities and to have a better organization and control over them, this model also intends to apply the strategy that is a modification of the Pattern W, which focus its attention on the cycle of life of the software before its setting in production.

Índice

Índice de Tablas

Índice de Figuras

Introducción

En el mundo actual existe una tendencia que consiste en aplicar las pruebas a lo largo de todo el proceso de desarrollo de software, a pesar de esto, se puede decir que esta rama de la ingeniería de software comienza ahora. Muchas empresas renombradas y de gran experiencia como: Microsoft[1] e Intel[2], desarrollan este proceso de forma eficiente y eficaz, producto a que realizan sistemas de información de forma confiable y seguros para la toma de decisiones. Además, crean sus propias herramientas a la hora de evaluar sus productos. Las herramientas que utilizan son: modelos de pruebas automáticos, elaboran auditorías a sus aplicaciones buscando con ello, mejorar la calidad y la confiabilidad de los sistemas de información, disminuyendo así, su costo operativo y los errores en el sistema. Estas empresas han sido muy influyentes en el desarrollo de la industria informática a través de la construcción de software con un alto nivel de calidad.

La empresa especializada en aseguramiento de calidad de software GreenSQA[3], como proveedor del servicio de pruebas de software, establece las prácticas más comunes del servicio, así como las consideraciones a tener en cuenta para subcontratar el servicio de pruebas de software. Este proceso analiza aspectos relevantes en el momento de subcontratar el servicio de pruebas de software. Los requisitos del proveedor del servicio de pruebas de software y la capacidad tecnológica del proveedor dependen del conocimiento y uso de estándares internacionales de pruebas. Los tipos de pruebas que podrían ejecutarse son: (funcionales, no funcionales, de carga y de estrés), la infraestructura tecnológica con que cuenta, la disponibilidad de un laboratorio de pruebas y en algunos casos la automatización de su proceso de pruebas de software. Todos los aspectos

[1] Microsoft: Empresa dedicada al sector de la informática, esta desarrolla, fábrica, licencia y produce software y equipos electrónicos.

[2] Intel: Es una de las empresas más influyentes en el desarrollo de la industria informática, líder de productos para computadoras, redes y comunicación.

[3] Software Quality Assurance: Empresa para hacer pruebas funcionales a productos de software.

mencionados anteriormente son los que tienen presente esta organización para prestar servicios y para la realización de pruebas de software. Adicionalmente ofrece el servicio de acompañamiento en la implementación de sistemas de gestión de calidad bajo referentes como la norma nacional ISO 9001[4] y el modelo CMMI[5]. Junto a esta empresa se ha unido Parquesoft[6] con el objetivo de liderar en la industria de software con empresas y productos competitivos, para trabajar conjuntamente en una estrategia de calidad que cubra todos los aspectos de calidad en la producción de software, para convertir los productos desarrollados en ParqueSoft y testeados por GreenSQA en los mejores productos de software a nivel internacional (1), (2), (3).

Igualmente existen metodologías que validan el proceso de desarrollo de software como BaQEM[7]. Este proceso está originalmente basado en tres etapas, las cuales elaboran un modelo jerárquico de requerimiento de prueba, partiendo de los procesos funcionales que soportan el producto a evaluar (4).

Las pruebas que se le realizan al software son un elemento crítico para la calidad del mismo, puesto que la captura de sus errores promueve la definición y aplicación de un proceso de prueba minuciosa y bien planificada. Estas pruebas permiten validar (proceso que reconoce si el software satisface sus requisitos) y verificar (proceso que determina si los productos de una fase satisface sus condiciones) el software. Una de las prioridades en el desarrollo de software en la actualidad, es obtener sistemas que cumplan con lo requisitos especificados por los clientes y un alto grado de calidad, por lo que se hace importante tener presente que la calidad del software es

[4] Organización Internacional de Normalización 9001: Sistema de Gestión de la calidad - Requisitos.

[5] Capability Maturity Model Integration: Modelo de la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software.

[6] Parque Tecnológico de Software: Empresa de base tecnológica especializada en la industria del conocimiento, a través de modelos de procesos de producción de productos y servicios basados en las mejores prácticas.

[7] Business Application Quality Evaluation Method: es una metodologías de calidad en el desarrollo de software, elabora un modelo jerárquico de requerimientos de prueba partiendo de los procesos funcionales que soporta el producto a evaluar, además se estudiaran los modelos de evaluación de software de ParqueSoft para tener marco de referencia claro.

el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia, siendo uno de los elementos fundamentales para el desarrollo de un software con calidad en el proceso de pruebas.

Las pruebas de software son una parte del proceso de aseguramiento de calidad; realizar pruebas a un sistema de información no significa necesariamente que el proceso de desarrollo esté asegurado y tampoco que de manera directa esté mejorando. Pero implementar un proceso de pruebas de software y más aún sostenerlo en el tiempo, es un buen inicio para más adelante aumentar el alcance y con base en las reflexiones al interior del equipo y de los hallazgos registrados en su producto, lo que realiza un mejoramiento del proceso de desarrollo basado en los lineamientos del aseguramiento de calidad de software.

¿Cómo obtener un software con calidad?, se tiene que controlar que estén bien definidos los parámetros, indicadores o los criterios de medición para la utilización de metodologías o procedimientos para el análisis, diseño, programación y pruebas del software que permitan lograr una mayor confiabilidad y facilidad de prueba (5), (6).

En Cuba muchas de las empresas trabajan para lograr informatizarse de manera acelerada, brindan soluciones integrales eficaces en Tecnologías de Información a las organizaciones, para contribuir eficientemente a la sociedad cubana, prestando servicios de desarrollo que les permitan organizar y gestionar los productos y su comercialización mayorista, desplegando e implantando software. Las mismas tienen presente la calidad de su software, para que esté libre de errores en su etapa de ejecución y que su interacción con el cliente sea fácil y cumpla con los requisitos propuestos, lo cual hace posible que conlleve a lograr un amplio sector de competitividad en el mercado.

Como ejemplo se encuentran Softel[8], CITMATEL[9], Segurmática[10], la UCI (Universidad de las Ciencias Informáticas) y Desoft[11], esta última es líder en el país

[8] Soluciones Informática: Empresa que proporciona soluciones informáticas que eleven la eficiencia de los servicios de salud con personal y tecnologías de avanzada.

[9] Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados: se distingue en el desarrollo científico y tecnológico de la sociedad cubana.

en soluciones integrales con reconocimiento en el mercado internacional, que cuentan con poderosas redes para brindar servicios de soporte, con un elevado nivel de profesionales y con tecnologías avanzadas para la creación de sus servicios (7).

Hoy en día, en la Universidad de Cienfuegos "Carlos Rafael Rodríguez" (UCF) específicamente en la facultad de informática, no se conoce de un grupo capacitado que analice detenidamente todo el proceso de desarrollo de software, este proceso será realizado por el programador en un período de tiempo a corto plazo; luego de haberse culminado el ciclo de vida del software. Este período de prueba se realiza empíricamente, lo cual no queda plasmado en ningún documento, lo que provoca que no se tenga conocimiento acerca del tipo y el método de prueba utilizado, dando la medida de que todo este proceso debe hacerse en un tiempo más largo para que se pueda tener un amplio conocimiento acerca de qué prueba será las más adecuada a dicho software y no hacerlo de forma empírica, además esta institución carece de una guía bien definida para verificar y validar los errores mientras el software está en su etapa de ejecución.

Teniendo en cuenta la situación problémica anteriormente especificada queda definido como ***problema científico***: ¿Cómo plasmar el comportamiento deseado de los ingenieros de prueba, con el fin de facilitar el proceso personal de prueba de software llevadas a cabo en la Facultad de Informática de la UCF?

El ***objeto de estudio*** es: El proceso de prueba de software.

El ***campo de acción*** queda definido como: El proceso personal de pruebas de software en la Facultad de Informática.

Quedando definido así el siguiente ***Objetivo general:*** Elaborar un Manual de Prueba para Ingenieros de software en la Facultad de Informática en la Universidad de Cienfuegos "Carlos Rafael Rodríguez".

[10] Consultoría y Seguridad Informática: Empresa de Consultaría y Seguridad informática en Cuba.

[11] Empresa Nacional de Software: Empresa encargada de la automatización de la sociedad cubana en la elaboración de software.

Para dar cumplimiento al objetivo general trazado quedan definidas las siguientes ***tareas de investigación***:

- Realización de un estudio de las normas y estándares relacionados con el proceso de pruebas de software, ejemplo ISO 9000-1, 9126[12], ISO/IEC 12207[13].
- Realización de un estudio sobre las Metodologías de Desarrollo de Software, de forma tal que permita fundamentar la elección de la metodología a utilizar.
- Análisis de las herramientas que soportan el proceso de pruebas.
- Realización de un estudio de Equipo Procesamiento de Software (TSP) y Proceso Personal de Software (PSP), de forma tal que permita fundamentar la elección del proceso a utilizar.
- Realización de un estudio de los Modelos V y W.

Teniendo en cuenta la información anterior queda definida la siguiente ***idea a defender:*** la realización de un manual de prueba para ingenieros de software, facilitará el proceso personal de prueba llevadas a cabo en la Facultad de Informática de la UCF.

Este trabajo ***aporta teóricamente*** la propuesta de un manual de prueba que ayudará al ingeniero de prueba a evaluar que el software se desarrolle con calidad y cumpla con las expectativas del cliente.

El trabajo se estructuró en tres capítulos: En el ***primer capítulo***, "Fundamentación Teórica", se da a conocer el concepto de prueba de software, su clasificación en cuanto a métodos, tipos y niveles, además de las características generales que debe tener una prueba para determinar que es eficiente. Se analizaron métodos y estándares referentes al proceso del ciclo de vida del software. Se tuvo en cuenta qué es Verificación y Validación y cómo se ve en una organización. Se identificaron herramientas de soporte para el proceso de prueba. ***El segundo capítulo***, lleva por

[12] International Standards Organization 9126: Estándar internacional para evaluar la calidad del software.

[13] Information Tecnnology - Software life cycle pocesses 12207: Estándar para gestionar el ciclo de vida del software.

nombre “Manual de prueba para el Ingeniero de software”, en el cual se brinda el manual a utilizar a la hora de evaluar un software, basado en el proceso de Verificación y Validación propuesto en la IEEE 1012-2004, además se analizan las actividades que lo sustentan, los roles vinculados al proceso de prueba y los tipos de pruebas, revisión e inspección que se deben utilizar para realizar dicho proceso. ***El tercer capítulo***, “Validación del Manual de prueba para el Ingeniero de software”, en este capítulo se muestra la validación del manual propuesto mediante el paquete estadístico SPSS 15.0.

Capítulo 1 – Fundamentación teórica.

1.1. Introducción

En este capítulo se hará referencia al proceso de pruebas de software; el cual se ejecuta de manera controlada, se usará en conjunto con los términos de Verificación y Validación. La estrategia de prueba de software permite enfocar el plan de prueba, que tiene presente los roles involucrados en cada una de ellas, dándose a conocer las actividades y su visión global. Estas pruebas tienen un objetivo fundamental: encontrar defectos en los proyectos. Mientras más fácil y menos engorrosas sean estas pruebas el resultado será mejor. Es preciso aclarar que la calidad del software no se certifica, lo que se certifica son los procedimientos para su elaboración, estos deben estar en función de la normalización (ISO 9001, CMMI). El proceso de pruebas es parte del ciclo de desarrollo de software, como uno de los tantos procedimientos que intervienen en la creación de un producto.

1.2. ¿Qué son las pruebas?

1.2.1. Conceptos de Pruebas.

Es importante destacar que para realizar una buena evaluación a un proceso de desarrollo del software, hay que tener presente qué es una prueba y, además, conocer otros conceptos definidos por varios autores:

En la siguiente cita (8) define el concepto de prueba*: "*Una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto".

Define **Javier Gutiérrez** como Prueba: "La verificación dinámica del comportamiento del software a partir de un conjunto finito de casos de prueba" (9).

Según Pressman, las Pruebas son: "El proceso de ejecución de un programa con la intención de descubrir un error" (10).

La (IEEE 1012, 2004) define que prueba 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" (11).

Se puede decir que prueba es ahorrar tiempo y recursos, garantiza la satisfacción del cliente y la calidad del software; además permite verificar y revelar, de un producto de software, cualquier fallo que se haya detectado y, algo fundamental, se subsanan todos los errores encontrados.

1.2.2. Clasificación

1.2.2.1 Métodos

Uno de los métodos de pruebas utilizados es el de Caja Blanca o también conocido, como Caja Clara o Transparente (12); su criterio es basado en el contenido de los módulos. El criterio de selección de casos de prueba buscará cierta cobertura de caminos independientes; valores de las condiciones bucles dentro y fuera de sus límites operacionales, estructuras de datos, los errores se esconden en los rincones y se acumulan en las fronteras (13).

Este método presenta algunas clases de pruebas (14) como:

- Pruebas de cubrimiento: esta ejecuta al menos una vez cada sentencia y necesita varios casos de prueba.
- Pruebas de condiciones: deben cumplir o no cada parte de cada condición, se necesitan varios casos de prueba entre los que se encuentran: determinar expresiones simples en las condiciones, una por cada operando lógico o comparación y cada expresión simple debe cumplir en un caso y en otro no, siendo decisiva en el resultado. Además sus expresiones simples no independientes que son imposibles cubrir al 100 %.
- Pruebas de bucles: consiguen el número de repeticiones especiales, bucles simples (repetir cero, una y dos veces, repetir un número medio (típico) de veces, repetir el máximo-1, máximo y máximo +1) además presenta bucles anidados (repetir un número medio (típico) los bucles internos, el mínimo los externos, y variar las repeticiones del bucle intermedio ensayado y ensayarlo con cada nivel de anidamiento).

El método de Caja Negra o Caja Opaca procura ejercitar cada elemento de la interfaz y las especificaciones de los módulos. Este método detecta los errores de la interfaz, los errores de accesos de estructura de datos externos, el problema de rendimiento y los errores de inicio y terminación. Utiliza como cobertura los valores representativos de conjuntos de datos, fronteras, combinaciones de valores conflictivos y la capacidad de proceso.

Este método presenta algunas clases de pruebas (14) como:

- Cubrimiento: invocar todas las funciones (100%).
- Clases de equivalencia de datos: donde los datos se clasifican según las distinciones visibles en la interfaz del programa.
- Casos de ensayo con datos de cada clase.
- Pruebas de valores de límite.

1.2.2.2. Tipos

Las pruebas además de evaluarse a través de métodos, también se clasifican por tipos; donde cada uno de ellos tiene su funcionalidad correspondiente de un producto específico, ejemplos:

- Pruebas Performance.
- Pruebas Funcionales.
- Pruebas de Integridad.
- Pruebas de Volumen.
- Pruebas de Estructuras.
- Pruebas de Estrés.
- Pruebas de Benchmark.
- Pruebas de Carga.
- Pruebas de Configuración.
- Pruebas de Contención.
- Pruebas de Seguridad.
- Pruebas de Instalación.

1.2.2.3. Niveles

Las pruebas además de agruparse por métodos y tipos también poseen niveles, por ejemplo:

- Prueba de Unidad.
- Prueba de Integración.
- Prueba de Desarrollador.
- Prueba de Sistema.
- Prueba de Independiente.
- Prueba de Aceptación.

1.2.3. ¿Qué evalúa una prueba?

Como se explicaba con anterioridad las pruebas no son más que un medio o instrumento utilizado para evaluar, detectar y resolver los posibles errores de un producto en desarrollo, permitiendo elevar las cualidades y funcionalidades que garanticen la calidad del producto final.

Existen varios modelos de calidad de software que pueden ser aplicados en diferentes empresas o proyectos, tratando de poner en práctica el concepto de calidad con el objetivo de mejorar su producto, tales como McCall, Boehm e ISO 9126 (15).

1.2.4. Características

Para poder clasificar una prueba como idónea las mismas deben cumplir con ciertas características que son vitales, dentro de las cuales no pueden faltar las siguientes.

- Ha de tener una falta de probabilidad de encontrar un fallo. Cuanto más, mejor.
- No debe ser redundante. Si ya funciona, no se prueba más.
- No debería ser ni demasiado sencilla ni demasiado compleja. Si es muy sencilla no aporta nada, si es muy compleja a lo mejor no se sabe lo que ha fallado.

1.2.5. Cuándo frenar la etapa de prueba.

Según el estándar IEEE 1012, 2004 las pruebas al software deben ser realizadas desde que se inicia el desarrollo del producto y debe culminarse cuando se termina el funcionamiento del producto. Existiendo, a pesar de esto, diversidades de criterios de cuándo se debe parar este proceso. Debido a un conjunto de factores de riesgo que obligan a reducir el tiempo dedicado a las pruebas y en muchas ocasiones hasta provocan su fin; uno de estos factores está dado por un desfasaje de retardo del cronograma de desarrollo del producto, se agota el dinero disponible para tal efecto o cuando los casos de pruebas no detectan errores. Según (16), cuando se han detectado 4-8 errores por cada 100 líneas de código.

Existe un modelo llamado logarítmico de Poisson de tiempo de ejecución, que permite determinar el número máximo de errores que presenta el software cuando está siendo probado y comprobar si van disminuyendo a medida que se avanza con las pruebas.

El modelo adquiere la siguiente forma:

(1)f(t) = (1/ *p) ln* (l_o *pt* +1)

f(t): es el número de fallos que se espera que se produzca en una x cantidad de tiempo una vez probado el software .

l_o: es la intensidad de fallo inicial de software (fallos por unidad de tiempo) al principio de la prueba.

p: es la reducción exponencial de la intensidad de fallo a medida que se encuentren los errores y se van haciendo las correcciones.

L a intensidad de fallo instantáneo, l (t) se obtiene mediante la derivada de f(t).

(2)(t) = l_o / (l_o *pt* + 1)

Mediante la ecuación (2) los ingenieros de pruebas pueden predecir la disminución de errores a medida que avanzan en las mismas.

La intensidad del error real se puede trazar junto a la curva predicha (Figura 1. Curva predicha). Si los datos reales recopilados durante las pruebas y el modelo logarítmico de Poisson de tiempo de ejecución, están razonablemente cerca uno de otro. Sobre un número de punto de datos, el modelo se puede usar para predecir el tiempo total requerido de prueba, para alcanzar una intensidad de fallos aceptablemente baja y poder determinar una fecha posible de culminación de las etapas de pruebas.

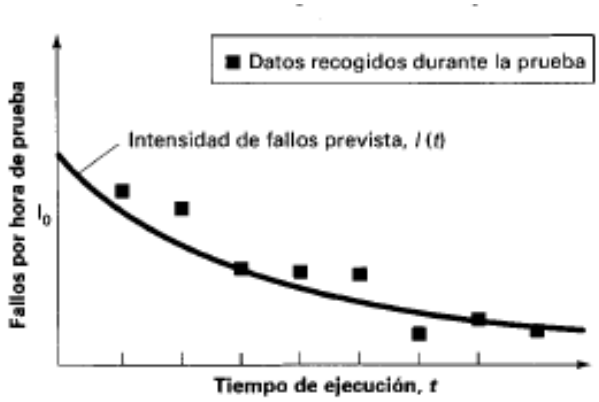

Figura 1. Curva predicha

No es recomendable que se dejen de realizar pruebas al producto cuando se culminen sus fases de desarrollo, sino que se continúen aplicando hasta asegurar un determinado nivel de calidad y aceptación por el cliente del producto (10).

1.3. Desarrollo del software y su relación con las pruebas.

1.3.1. V & V

El proceso de Verificación y Validación (V&V) es el conjunto de procesos de comprobación y análisis que aseguran que el software desarrollado está acorde con su especificación y cumple las necesidades de los clientes (17). Existen una serie de actividades de V&V en cada una de las etapas del proceso de desarrollo de software.

La V&V es un proceso de vida completo, inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño y las inspecciones del código hasta la prueba del producto.

Es importante destacar que Validación y Verificación no son la misma cosa, para poder darle una explicación más detallada de estos dos términos del proceso de

prueba del software, podemos decir, que Verificación es la revisión o las pruebas de los elementos, incluyendo el software para obtener conformidad y consistencia con una especificación asociada. Las pruebas de software son sólo un tipo de Verificación. La Validación asegura que el software cumple las expectativas del cliente. Va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado. Dentro del proceso de Verificación y Validación se utilizan dos técnicas de comprobación y análisis de sistemas. La primera técnica es la *inspección del software* que analizan y comprueban las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y el código fuente del programa. Se aplica a todas las etapas del proceso de desarrollo. Las inspecciones se complementan con algún tipo de análisis automático del texto fuente o de los documentos asociados. Las inspecciones del software y los análisis automatizados son técnicas de Verificación y Validación estáticas, puesto que no requieren que el sistema se ejecute (17). Y la segunda técnica de este proceso *son las pruebas del software,* esta consiste en contrastar las respuestas de una implementación del software a series de datos de prueba y examinar las respuestas del software y su comportamiento operacional, para comprobar que se desempeñe conforme con lo requerido. Llevar a cabo las pruebas es una técnica dinámica de la Verificación y Validación, porque requiere disponer de un prototipo ejecutable del sistema (17).

La técnica de prueba sólo se puede utilizar cuando existe prototipo o código ejecutable. El proceso que localiza y corrige los errores descubiertos durante la Verificación y Validación, es el proceso de depuración. Este proceso está integrado por:

- La Verificación y Validación establece la existencia de defectos en el programa.
- La depuración es el proceso que localiza el origen y corrige estos defectos.

No existe un proceso sencillo para la depuración de programas. Los mejores depuradores buscan patrones en los resultados de las pruebas donde el defecto se detecta y para localizarlo se utiliza el conocimiento que tienen sobre el tipo de

defecto, el patrón de salida, así como del lenguaje y proceso de programación. El conocimiento del proceso es importante. Los depuradores conocen los errores de los programadores comunes (olvidar, incrementar un contador y errores de direccionamiento de punteros en lenguaje C.) y los comparan contra los patrones observados.

Dentro de la técnica de prueba de software existen dos tipos diferentes de prueba, que se utilizan en distintas etapas de desarrollo del software:

1. *Las pruebas de defectos*: pretenden encontrar las inconsistencias entre un programa y su especificación. Estas inconsistencias se deben habitualmente a los fallos o defectos en el código del programa. Las pruebas se diseñan para revelar la presencia de defectos en el sistema, más que para evaluar su capacidad operacional.

2. *Las pruebas estadísticas:* se utilizan para probar el desempeño y la fiabilidad del programa y comprobar cómo trabaja bajo condiciones operacionales. Las pruebas se diseñan para reflejar las entradas de los usuarios y su frecuencia. Después de llevar a cabo las pruebas, se puede hacer una estimación de la fiabilidad operacional del sistema contando el número de caídas observadas en el mismo. La capacidad del programa se valora midiendo el tiempo de ejecución y el tiempo de respuesta del sistema cuando procesa los datos estadísticos de la prueba.

Localizar los fallos es un proceso complejo, porque no necesariamente se localizan cerca del punto en que se detectan. Para localizar un fallo de un programa, el programador responsable de la depuración tiene que diseñar programas de prueba adicionales que repitan el fallo original y que ayuden a descubrir el origen del fallo. Habitualmente, permiten controlar la ejecución paso a paso sobre el código del programa.

Después de que se descubre el origen del fallo, este debe corregirse y entonces reevaluar el sistema. Esto implica repetir de nuevo las pruebas anteriores (pruebas de regresión). Estas pruebas se hacen para comprobar que los cambios introducidos resuelven definitivamente el fallo y no introducen nuevos fallos.

1.3.2. Metodologías RUP y XP

Existen diversas metodologías que las empresas utilizan como herramienta a la hora de realizar pruebas de software, las cuales permiten llevar el nivel de calidad del software, para poder realizar un proceso de prueba más efectivo con la detención temprana de sus errores.

Citado de (18) la metodología RUP (Rational Unified Process) es un proceso de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistema orientado a objeto. Además, es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, está centrado en la arquitectura y guiado por casos de uso. Incluye además artefactos (los productos tangibles del proceso) y roles (papel que desempeña una persona en un determinado momento). El desarrollo del software se divide en 4 fases: Inicio, Elaboración, Construcción y Transición. El flujo de trabajo de las pruebas verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias. Los objetivos de la prueba de RUP son: planificar las pruebas necesarias de cada iteración incluyendo las de integración y las de sistemas, diseñar e implementar, realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Durante la fase de Transición el centro se desplaza hacia la corrección de defectos durante los primeros usos y a las pruebas de regresión. El modelo de prueba de RUP cambia constantemente debido a la eliminación de casos de pruebas obsoletos. Esta metodología es más adaptable para proyectos de largo plazo.

La estructura de los procesos de la metodología RUP pueden ser descritos en dos dimensiones o ejes (19):

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Figura 2 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Como se mencionó anteriormente cada fase se subdivide, a la vez, en iteraciones.

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

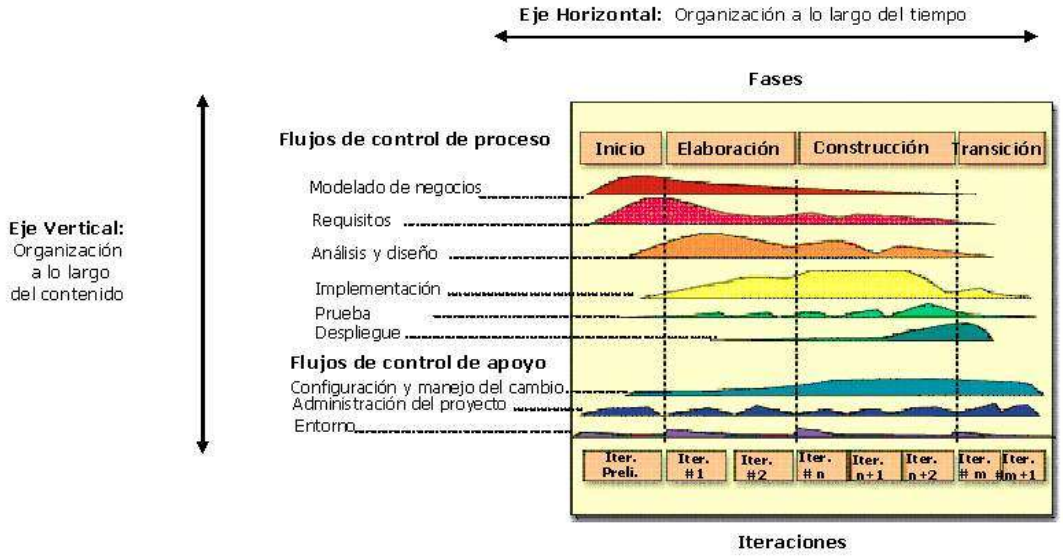

Figura 2. Ciclo de vida de RUP, Flujos de trabajos, Fases.

Durante la fase de Inicio y Elaboración se definen el modelado del negocio y los requisitos del proyecto, estos son los dos flujos fundamentales en estas dos fases, aunque todas las demás actividades también se ven presentes.

En la fase de Elaboración y Construcción, donde el objetivo de la fase de Elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan de proyecto, eliminar los mayores errores y crear un plan fiable para la fase de Construcción, esta última fase mencionada tiene como finalidad principal alcanzar la capacidad operacional del producto de forma incremental. Estas dos fases tienen cuatro flujos de trabajo que son los que más se destacan, estos son: Análisis y Diseño, Implementación, Prueba y Despliegue, este último flujo también se resalta en la fase Transición, donde su finalidad es poner el producto en manos de los usuarios finales.

Otras de las metodologías es la XP (Extreme Programming), es una de las metodologías de desarrollo de software más exitosa, en la actualidad utilizada para proyectos a corto plazo. Consiste en una programación rápida o extrema, cuya

particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Esta metodología posee características que se basan en(20):

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir.
- Refabricación: hace énfasis en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lo más importante en esta metodología es la comunicación entre los usuarios y los desarrolladores y la simplificación al desarrollar y codificar los módulos del sistema.

Los ciclos de vida de un proyecto en XP y en RUP no son exactamente iguales, aunque sin duda tienen bastantes similitudes, ambas son metodologías iterativas con éxito en el desarrollo de software. Además las dos crean como base UseCase y UserStories, y describen los requerimientos de la aplicación desde el punto de vista del usuario. Ambos definen los requerimientos técnicos sin meterse en detalles de implementación (21). Entre sus grandes diferencias se encuentra que RUP es creado para proyectos y equipos grandes en cuanto a tamaño y duración y XP se implementa mejor para proyectos y equipos cortos.

RUP proporciona muchas ventajas sobre XP, le da énfasis en los requisitos y el diseño, además se basa principalmente en las mejores prácticas que se han intentado y probado en el campo, existe un contacto prefijado con el cliente donde este interactúa con el equipo de desarrollo mediante reuniones, a diferencia que XP el cliente forma parte del equipo y se fundamenta en las prácticas inestables, que

utilizándolas juntas se evita que se derribe (22), por todo lo dicho con anterioridad se puede llegar a la conclusión de que RUP es extremadamente mejor que XP.

1.3.3. ISO 9126

La ISO 9126, 1992 es un estándar internacional para evaluar la calidad del software, fue publicado en 1992, el cual establece 6 características de calidad para un producto de software. Cada una de ellas presentan sub-características que permiten profundizar sobre la evaluación de prueba del software y además un conjunto de atributos que las clasifica, evalúa su nivel de funcionamiento o cantidad de recursos usados, realiza modificaciones que consisten en corregir errores o incrementar su funcionalidad, la capacidad de instalación del software y sus resultados o efectos, para saber si alcanzaron los objetivos por el cual fue creado (23).

1.3.4. ISO 9000 1, 3

La ISO 9000 3 es un estándar internacional, el cual se caracteriza por llevar el control de la calidad de todas las fases de la producción del software, incluyendo el mantenimiento y las tareas posteriores a su implantación y además debe existir una estricta colaboración entre la organización que adquiere el software y el proveedor. Esta norma proporciona una guía útil que sirve para detectar y corregir una serie de errores de los productos de software, consiguiendo una mejora en la calidad de los mismos (24).

La ISO 9000-1 y IEEE -73 son los que más se relacionan con la ISO 90000-3.

La (ISO 9001, 2000) es un estándar internacional que especifica los requisitos para un sistema de gestión de la calidad en los requisitos del cliente. Esta norma exige que las organizaciones midan y hagan un aseguramiento de las características del producto para verificar que se cumplan los requisitos del mismo, además se verifican y se validan los productos que las empresas adquieren.

Esta norma tiene definida entre sus secciones una dedicada a la instrucción y pruebas, que guarda estrecha relación con el proceso de prueba que se desarrolla en determinados productos de software a lo largo de un ciclo de vida. Propone que la empresa debe asegurar que los productos adquiridos no se utilicen o procesen hasta

que sean inspeccionados o verificados y que cumplan con los requerimientos específicos. La verificación debe estar de acuerdo con el plan de calidad y los procedimientos documentados que ya están establecidos. Para aquellos productos que son enviados a productos por situaciones de urgencias sin ser antes inspeccionados, estos deben identificarse y registrarse para que en caso de no conformidad sean rápidamente reconocidos y reemplazados. La empresa debe establecer o mantener registros que tengan el criterio de aceptación del producto (15).

1.3.5. CMMI

El CMMI (Maturity Model Integration) es la evolución de CMM. Este fue lanzado en el 2002, para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistema de software. Existen varias versiones pero la actual es la versión v1.2 que está formada por tres constelaciones. Es un modelo de calidad del software que clasifica las empresas en 6 niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software (25).

Nivel 0 (Incompleto): los objetivos no están bien definidos.

Nivel 1 (Ejecutada): este es el nivel donde todas las empresas no tienen procesos.

Nivel 2 (Administrado): en este nivel el proyecto es gestionado y controlado durante el desarrollo.

Nivel 3 (Definido): indica la forma de desarrollar proyectos (gestión e ingeniería) que están definidos, lo que quiere decir que está establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos concretos.

Nivel 4 (Cuantitativamente Administrado): los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización. Se usan métricas para gestionar la organización.

Nivel 5 (Optimizado): los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica.

1.3.6. ISO 12 207

Es un estándar de modelo establecido para gestionar el ciclo de vida del software. Esta norma indica que tiene procesos y estos procesos tienen tareas que señalan acciones que transforman entradas (requerimientos) en salidas (producto del software). Estas tareas se implantan con las metodologías de gestión de proyectos (PMI) y metodología desarrollo de software (RUP, XP, MSF), las cuales tienen fases, planes, entregables, artefactos, cronogramas y etapas (26).

La estructura del estándar ha sido concebida de manera flexible y modular de manera que pueda ser adaptada a las necesidades de cualquiera que lo use. Para conseguirlo, el estándar se basa en dos principios fundamentales: Modularidad y Responsabilidad. Con la Modularidad se pretende conseguir procesos con un mínimo acoplamiento y una máxima cohesión. En cuanto a la Responsabilidad, se busca establecer un responsable para cada proceso, facilitando la aplicación del estándar en proyectos en los que pueden existir distintas personas u organizaciones involucradas.

Los procesos se clasifican en tres tipos: principales, de soporte y de la organización. Los procesos de soporte y de organización deben existir independientemente de la organización y del proyecto ejecutado. Los procesos principales se instancian de acuerdo con la situación particular (25).

- Procesos principales:
    - Adquisición.
    - Suministro.
    - Desarrollo.
    - Operación.
    - Mantenimiento.
- Procesos de soporte:
    - Documentación.
    - Gestión de la configuración.
    - Aseguramiento de calidad.
    - Verificación.

- Validación.
- Revisión conjunta.
- Auditoría.
- Resolución de problemas.

- Procesos de la organización:
  - Gestión.
  - Infraestructura.
  - Mejora.
  - Recursos Humanos.

1.4. Modelos de pruebas

El modelo en V (La V indica también Verificación y Validación) es el modelo más comúnmente extendido por su simplicidad.

A diferencia de los modelos clásicos, extiende las pruebas a lo largo de todo el ciclo de vida del software. La Figura 3 y la Figura 4 muestran las diferencias entre procesos que aplican las pruebas en momentos concretos o a lo largo del ciclo de vida.

Figura 3. Modelo de pruebas clásico.

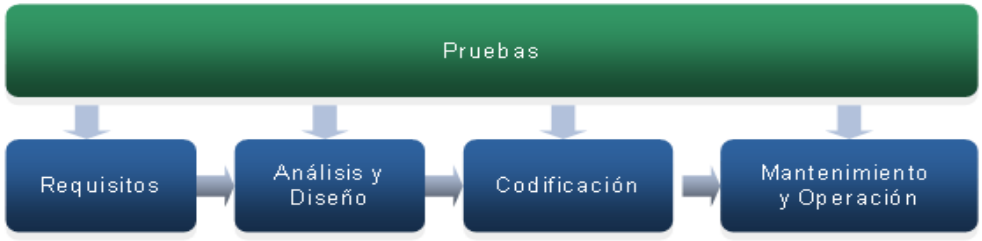

Figura 4. Modelo de pruebas a lo largo del ciclo de vida.

El modelo en V deriva directamente de la aplicación de pruebas de Verificación y Validación a un ciclo de vida de desarrollo en cascada.

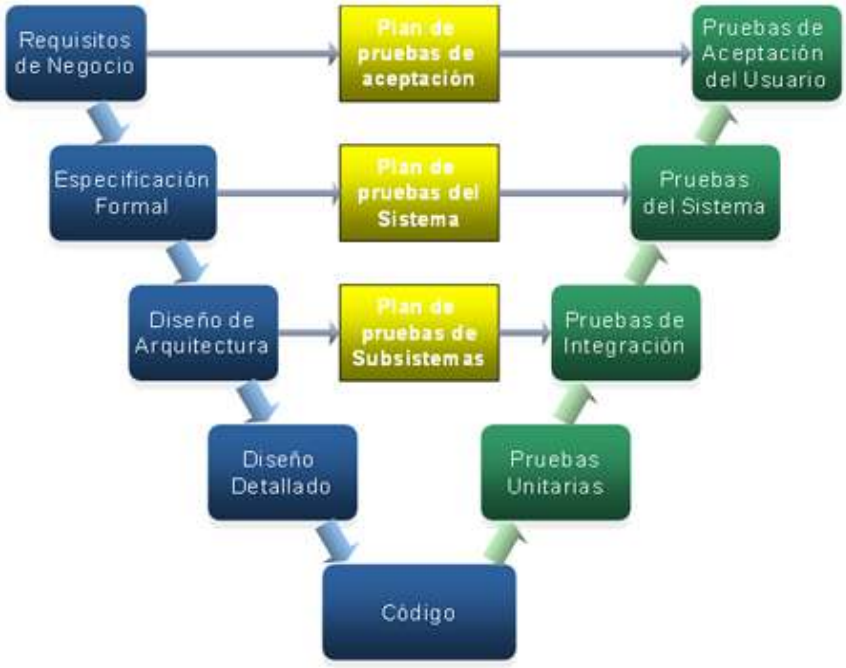

Figura 5. Modelo de Pruebas en V

Si se considera la V como una iteración del proyecto completo se puede decir que es extensible a modelos de desarrollo iterativo.

El plan de prueba se encuentra dentro del proceso de desarrollo iterativo el cual irá evolucionando conforme al proyecto que va transcurriendo por sus sucesivas iteraciones.

En este modelo se diseñan las pruebas desde etapas tempranas, donde más tarde serán ejecutadas en el ciclo de vida del software.

El *modelo en W* surge como mejora del *modelo en V*. Reflejando mejor la interdependencia que existe entre el equipo de desarrollo y el equipo de pruebas a lo largo de todo el proceso de desarrollo del sistema, destacando los siguientes dos puntos:

- En las primeras etapas se consideran labores de testing que no aparecen reflejadas en el modelo original, como son la revisión de los requisitos, la revisión de la especificación del sistema, la revisión de la arquitectura y del diseño de detalle y las revisiones de código.

- En las etapas finales también se distingue del modelo original en V, porque se desglosan las tareas de pruebas propiamente dichas y las labores de depuración y corrección de los errores detectados, que serán llevadas a cabo por desarrolladores.

Figura 6. Modelo de Pruebas en W

Según la Figura 6 del modelo en W, donde los bloques verdes indican las labores de pruebas y los bloques azules representan el resto de actividades del ciclo de vida del software antes de su puesta en producción. En las primeras etapas se concentran labores de verificación de productos no software y de preparación para la Verificación y Validación del software. En las etapas finales se desarrolla todo el esfuerzo de Verificación y Validación del software producido.

Desde el punto de vista del equipo que realiza las pruebas, se pueden distinguir dos tipos de actividades:

- Pruebas sobre productos no software:
    - Revisión de requisitos.
    - Revisión de la especificación de casos de uso.
    - Revisión del diseño de la arquitectura.

    - Revisión del diseño detallado.
- Pruebas sobre el software:
    - Elaboración de las pruebas de aceptación.
    - Elaboración de las pruebas de sistema.
    - Elaboración de las pruebas de integración de subsistemas.
    - Elaboración de las pruebas unitarias.
    - Revisiones del código.
    - Ejecución de las pruebas.
    - Depuración e introducción de cambios.

Las pruebas de Verificación y Validación de productos no software cubren la mayor parte de productos en formas de documentos. Se basan fundamentalmente en revisiones por iguales y en las que no necesariamente debe intervenir personal dedicado a pruebas. Los documentos se someten a un proceso de revisión establecido en el plan de calidad del proyecto, donde se define quién lo revisa (normalmente personal con al menos el mismo nivel técnico que el autor del documento y eventualmente el cliente), cómo se corrigen los defectos y cómo finalmente se aprueba el documento para su uso en las siguientes etapas del ciclo de vida (27) (28).

1.5. PSP y TSP

El Proceso Personal de Software (PSP) es una versión pequeña de CMM donde se preocupa sólo por un conjunto de las capas. El PSP se caracteriza porque es de uso personal y se aplica a programas pequeños de menos de 10.000 líneas de código. Se centra en la administración del tiempo y en la administración de la calidad a través de la eliminación temprana de defectos. En el PSP se excluyen los siguientes temas: trabajo en equipo, administración de configuraciones y administración de requerimientos.

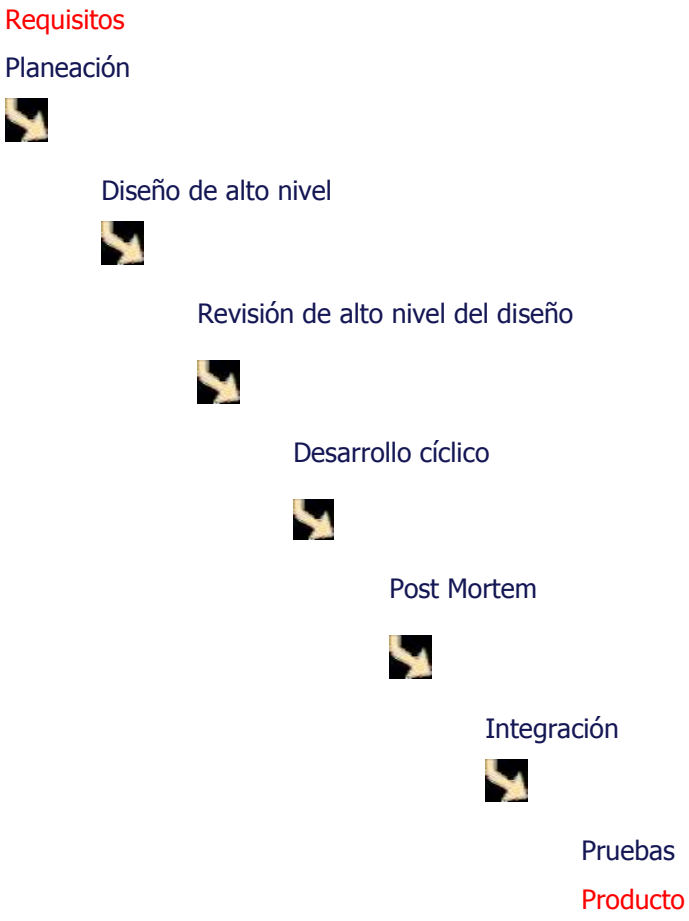

Figura 7. Ciclo de vida PSP, Fases.

El diseño de PSP se basa en los siguientes principios de planeación y de calidad (29).

- Cada ingeniero es esencialmente diferente; para ser más precisos, los ingenieros deben planear su trabajo y basar sus planes en sus propios datos personales.
- Para mejorar constantemente su funcionamiento, los ingenieros deben utilizar personalmente procesos bien definidos y medidos.
- Para desarrollar productos de calidad, los ingenieros deben sentirse personalmente comprometidos con la calidad de sus productos.
- Cuesta menos encontrar y arreglar errores en la etapa inicial del proyecto que encontrarlos en las etapas subsecuentes.
- Es más eficiente prevenir defectos que encontrarlos y arreglarlos.

- La manera correcta de hacer las cosas es siempre la manera más rápida y más barata de hacer un trabajo.

El TSP (Equipo Procesamiento de Software) es la mejora en la estimación, calidad y productividad de los equipos en la gestión del software. El TSP se enfatiza en las fases tempranas del proyecto, dedicando más tiempo a los requerimientos, diseño de alto nivel y revisando e inspeccionando el diseño. Asimismo el equipo se va consolidando con una visión compartida para el logro de las metas y objetivos. TSP está basado en los principios siguientes según (30):

- Los ingenieros conocen más acerca de su trabajo y pueden hacer mejores planes.
- Cuando los ingenieros planifican su propio trabajo, ellos se comprometen con el plan.
- Para minimizar la duración del proyecto, los ingenieros pueden balancear su carga de trabajo.
- Sólo la gente que hace el trabajo, puede recoger datos precisos.
- Para maximizar la productividad, hay que centrarse primero en la calidad.

1.6. Rol del Ingeniero de Prueba

El Ingeniero de Prueba de software es el rol fundamental para un proceso de prueba, el cual es responsable de ejecutar el conjunto de pruebas establecido para el software, registrar los defectos hallados y ejecutar las pruebas de regresión que garanticen la solución de los defectos y la estabilidad del resto del software. También existe responsabilidad compartida en el mejoramiento de procesos, junto con el equipo de desarrollo, quienes deben realizar la identificación de las causas de no conformidades y la definición de estrategias para el mejoramiento de procesos y por consecuencia de producto (2).

1.7. Herramientas

Una forma de hacer las pruebas es realizarlas de forma automatizadas, para esto existen herramientas que dan soporte al proceso de prueba, ejemplo de ellas (15):

Tabla 1. Herramientas para el soporte del proceso de prueba.

Administración de Prueba	Rational TestManager
Seguimiento de Errores	Rational ClearQuest
Administración de Proyectos	Microsoft Project.

Rational TestManager: Permite al equipo comenzar, dar seguimiento y probar toda la funcionalidad requerida de una aplicación, ayudando asegurar que ningún requerimiento crítico del negocio quede sin probar.

Rational ClearQuest: Captura de seguimiento y maneja eficientemente los diferentes tipos de cambios. Permite manejar de forma eficiente la trazabilidad dentro del proyecto.

Microsoft Project: Software para la administración de proyectos, fue diseñado para asistir a administradores de proyecto en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administra presupuesto y analiza carga de trabajo.

1.8. Conclusiones Parciales

En este capítulo se analizó detenidamente todo el proceso de prueba para el desarrollo de software, además de todas aquellas normas y estándares que se relacionan con el ciclo de vida de todo proyecto para asegurar que su calidad sea la más exacta posible y no tenga errores en su etapa de ejecución.

Este estudio permitió profundizar en el tema de cuáles son los tipos de pruebas que existen, sus niveles y métodos por los que las empresas deben regirse para evaluar el software en su etapa de ejecución. Además, se hizo un análisis detallado de los diferentes modelos, metodologías y herramientas para tener un mejor conocimiento en el momento de desarrollar un manual para el Ingeniero de Prueba. Quedando como resultado el Modelo W para plantear la estrategia apoyado en el estándar IEEE 1012-2004 que permite validar y verificar el desarrollo de software y paralelo a él la Metodología RUP, y será evaluado por el proceso personal de software.

Capítulo 2 – Manual de Prueba para el Ingeniero de software.

2.1. Introducción.

En este capítulo se le da una estructura al manual propuesto, donde se analizarán la metodología RUP y las actividades de V&V del estándar IEEE 1012-2004, además se evaluarán los conocimientos asociados que debe tener el ingeniero de prueba para enfrentarse a dicho manual. También se darán a conocer los conceptos referentes al ciclo de vida de software y los tipos de prueba, revisión e inspección que se le deben realizar al software para un mejor desarrollo.

2.2. Manual.

2.2.1. Nivel del Conocimiento.

Para poder ser un Ingeniero de Prueba es preciso tener un nivel de conocimiento de todo lo relacionado con el proceso de pruebas. Dicho conocimiento puede ser Básico, Intermedio y Avanzado. En la Tabla 2 se propone una serie de preguntas para dar la oportunidad a los ingenieros de prueba de evaluar sus conocimientos. Según la cantidad de preguntas que sea capaz de contestar será su nivel de conocimiento, más adelante se explicarán.

Preguntas de Auto evaluación

Tabla 2. Nivel de Conocimiento, Preguntas y Respuestas.

	Preguntas	Respuestas
1	¿Qué es Calidad?	
2	¿Qué son las pruebas?	
3	¿Qué es el ciclo de vida de un software?	
4	¿Conoces alguna herramienta para automatizar el proceso de prueba?	
5	¿Quién aplica las pruebas en el proceso de desarrollo de software?	
6	¿Qué es una metodología?	
7	¿Conoces algunas de las Actividades de V&V?	
8	¿Conoces métodos de pruebas? ¿Cuáles?	
9	¿Sabes identificar niveles de pruebas?	

10 ¿Sabes identificar tipos de pruebas?	
11 ¿Cuándo se aplican las pruebas?	

Nota: En la columna de Respuestas debe marcar con una X las preguntas que usted tiene su conocimiento previo.

Después de respondidas las preguntas anteriores, el ingeniero de prueba podrá saber si está apto para poder enfrentarse al manual de prueba, quedando definidos 3 niveles de conocimiento:

Si es capaz de contestar al menos 6 de las preguntas se puede decir que tiene un nivel básico, si contesta 9 un nivel intermedio y si contesta las 11 un nivel avanzado.

Si el ingeniero de prueba no alcanza el nivel básico, es necesario que estudie más acerca de lo que son las pruebas ya que es su objetivo primordial tener un breve conocimiento acerca de su labor como profesional.

2.2.2. Alcance

Establecer una guía para desarrollar el proceso personal de prueba de los software que son elaborados en la UCF para conseguir una elevada calidad del producto, cumplir con los requerimientos funcionales, los requerimientos no funcionales establecidos y además la satisfacción del cliente.

2.2.3. Objetivo

Proponer un Manual de Prueba para Ingenieros de software en la Facultad de Informática en la Universidad de Cienfuegos “Carlos Rafael Rodríguez.

2.2.4. Propósito

Al confeccionar el manual el ingeniero de prueba eleva el conocimiento, obtiene un comportamiento deseado para mejorar así el proceso personal de prueba de software en la Universidad de Cienfuegos “Carlos Rafael Rodríguez”.

2.2.5. Punto de partida

En este epígrafe se presenta el manual propuesto en forma de guía para su mejor entendimiento a la hora de aplicarlo. Fuente Bibliográfica (11).

Tabla 3. Flujos de trabajo de RUP, Actividades y Tareas V&V

RUP (Flujos de Trabajo)	**Actividades V&V**	**Tareas V&V**
Modelamiento del Negocio.	V&V de la Conceptualización.	Plan de las Tareas de V&V.
		Evaluación de la documentación.
Requisitos.	V&V de los Requerimiento.	Análisis de la Interfaz.
		Plan V&V de Generación de Pruebas y Verificación del Sistema.
		Plan V&V de Generación de Pruebas y Verificación de Aceptación.
		Revisiones e Inspecciones de los Requerimientos del Software.
Análisis y Diseño.	V&V de Diseño.	Evaluación del Diseño del Software.
		Análisis de la Interfaz.
		Plan V&V de Generación de Prueba y Verificación de Componente.
		Plan V&V de Generación de Prueba y Verificación de Integración.
		Plan V&V de Generación de Prueba y Verificación del Diseño.
		Revisión e Inspección de la arquitectura.
Implementación.	V&V de Implementación.	Evaluación del Código Fuente y documentación.
		Análisis de la Interfaz.
		V&V Generación de casos de prueba y de verificación.
		V&V Generación de Procedimientos de Pruebas y Verificación.
		V&V Prueba de Ejecución y Verificación de Componentes.
Prueba.	V&V de Prueba.	V&V Generación de Procedimientos de Pruebas y

		Verificación de la Aceptación.
		V&V Ejecución y Verificación de las Pruebas de Integración.
		V&V Prueba de Ejecución y Verificación del Sistema.
		V&V Prueba de Ejecución y Verificación de Aceptación.
		V&V de la Documentación.
Despliegue.	V&V de Instalación y Chequeo.	Auditoría de Instalación de Configuración.
		Chequeo de instalación.
		V&V Informe Final de Generación.
Configuración de los cambios.	V&V de Mantenimiento.	Revisión del SVVP.
		Evaluación de la migración.
		Evaluación del cambio.

Esta tabla muestra los flujos de trabajo del ciclo de vida de la metodología RUP, luego qué actividades de V&V se relacionan en cada uno y las tareas de V&V que indican cómo se efectuarán las actividades durante el ciclo de vida del software. Es necesario que se conozcan las entradas de cada una de las tareas que se realizarán, porque los productos que se crean constituyen la salida de dicha tarea.

En la Tabla 4 muestra la relación que existe entre el tipo de prueba a realizar en cada una de las actividades de V&V. En el epígrafe se explican los tipos de prueba propuestas, así como las revisiones e inspecciones al software.

Tabla 4. Actividades de V&V y Tipos de pruebas.

Actividades de V&V / **Tipos de pruebas**	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Revisión de la documentación y requisitos del negocio.	*X*	*X*					
Pruebas de Unidad.			*X*	*X*			
Pruebas de Integración.			*X*	*X*			
Revisión al Diseño.			*X*				

Pruebas de Integridad.				*X*	*X*		
Prueba de Sistema: Usabilidad.		*X*					
Prueba de Sistema: Rendimiento o Carga.					*X*	*X*	
Prueba de Sistema: Fiabilidad.					*X*	*X*	
Prueba de Sistema: Seguridad.				*X*	*X*	*X*	
Prueba de Sistema: Funcionales.				*X*	*X*		
Prueba de Sistema: Estrés.						*X*	
Prueba de Configuración.						*X*	
Pruebas de Instalación.						*X*	
Pruebas de Aceptación Alfa, Beta.						*X*	
Depuración.							*X*

Leyenda:

A: V&V de la Conceptualización. (Proceso de Desarrollo)

B: V&V de los Requerimientos. (Proceso de Desarrollo)

C: V&V de Diseño. (Proceso de Desarrollo)

D: V&V de Implementación. (Proceso de Desarrollo)

E: V&V de Prueba. (Proceso de Desarrollo)

F: V&V de Instalación y Chequeo. (Proceso de Desarrollo)

G: V&V de Mantenimiento. (Proceso de Mantenimiento)

2.3. Teoría del contenido.

El manual reúne todas las característica desarrolladas en la metodología RUP con respecto al ciclo de vida del software y define una Estrategia establecida en el proceso de V&V del estándar de la IEEE 1012-2004, además es necesario conocer qué tareas realizan cada una de ellas y los distintos tipos de prueba que pueden aplicar a dicha estrategia.

2.3.1. Ciclo de vida de RUP

Ciclo de vida del Software: "Es una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software" (25).

También es definido por la ISO/IEC 12 207 como: "Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso" (26).

Según IEEE Std 610 el ciclo de vida del software es el período de tiempo que comienza cuando el producto de software es concebido y termina cuando el software no está disponible permanentemente para el usuario (31).

En este documento se hace referencia a algunas definiciones dadas por diferentes estándares acerca de lo que es el ciclo de vida de un software, además como la metodología RUP tiene aspectos en común con el estándar de Verificación y Validación (11), para poder llegar a esta conclusión se tuvo que estudiar detalladamente sus flujos de trabajo los cuales definen el ciclo de vida del software. Estos se dividen en dos procesos: uno de Desarrollo y otro de Apoyo y a partir de aquí se arribó a que la semejanza con el Estándar de Verificación y Validación es que dentro de sus procesos primarios tiene incluido el Proceso de Desarrollo y de Mantenimiento. Se proponen estos dos últimos porque son los que presentan una gran relación con el proceso de prueba. Quedando así el Desarrollo y el Mantenimiento, como los procesos definidos a formar parte de la Estrategia propuesta porque a partir del desarrollo es que se planifican las pruebas.

A continuación se muestra un mapa conceptual, con todos los procesos definidos por RUP, reflejando cómo sólo se van a utilizar la parte de los procesos de Desarrollo anteriormente mencionada y definiendo, para ellos de manera general, las tareas V&V.

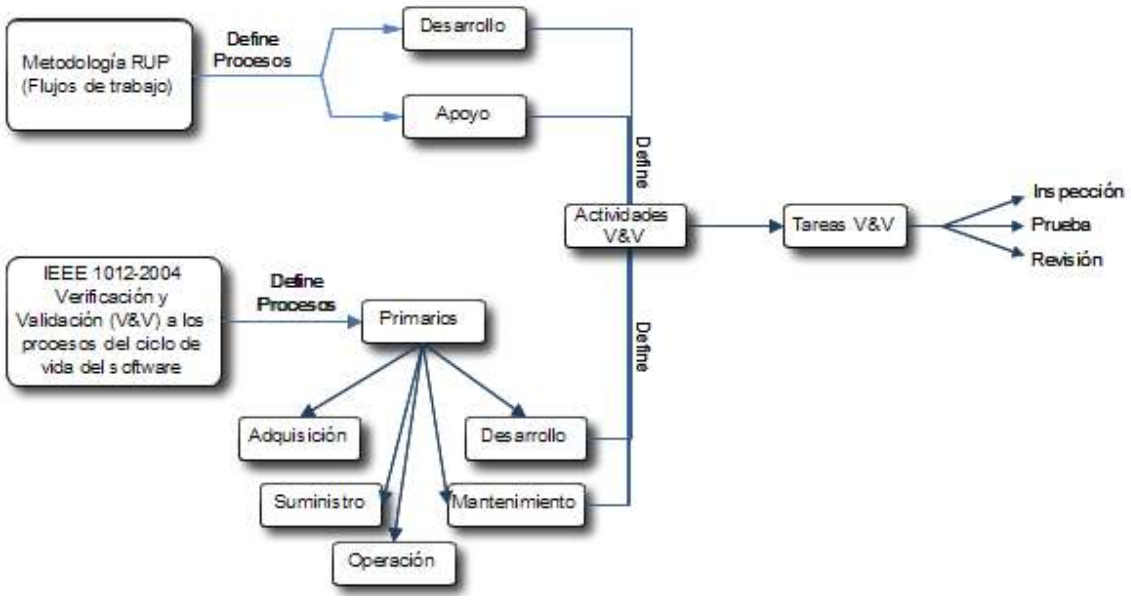

Figura 8. Marco de Trabajo entre Metodología RUP, Proceso de V&V, Actividades y tareas.

2.3.2. Proceso de V&V

Para poder determinar que un programa funciona correctamente y cumple con los requisitos especificados por el cliente, se hace necesario aplicar una de las técnicas de Verificación y Validación que no son más que las pruebas. Muchas de estas técnicas han sido utilizadas por más de treinta años, pero todavía no se han convertido en una tendencia principal de la ingeniería del software, una de estas técnicas, a la que se puede hacer mención, es la inspección de programa.

Boehm en 1979 expresó la diferencia entre ellas:

Validación:

"¿Estamos construyendo el producto correcto?"

Verificación:

"¿Estamos construyendo el producto correctamente?"

Estas diferencias dicen que el papel de la Verificación implica comprobar que el software está de acuerdo con sus especificaciones, además cumple con los requerimientos funcionales y no funcionales. La Validación, sin embargo, es un proceso más general, tiene como objetivo, asegurar que el sistema de software satisface las expectativas del cliente.

La Verificación y Validación tienen lugar en cada etapa del ciclo de vida del software, comenzando con las revisiones de los requerimientos y luego con revisiones del diseño e inspección de código hasta la prueba del producto (32).

Es el proceso de todo un ciclo vida, tiene dos objetivos principales:

El descubrimiento de defectos en el sistema;

La evaluación de si el sistema es útil y utilizable en una situación operacional o no.

2.3.2.1 Verificación dinámica y estática.

Existen dos técnicas dentro del proceso de V&V para el análisis y la comprobación del software:

- Inspección del software: Implica que las personas examinen la representación de la fuente con el propósito de descubrir anomalías y defectos. Además no requiere la ejecución de un sistema por lo que debe utilizarse antes de la implementación. Pueden estar aplicados a cualquier representación del sistema (requerimientos, diseño, configuración, datos, pruebas de datos).Se ha demostrado que es una técnica efectiva para descubrir errores del programa. Es una técnica de Verificación estática, donde no se necesita ejecutar el software en una computadora.
- Prueba de software: Puede revelar la presencia de errores NO su ausencia. Es la única técnica de validación para requerimientos no funcionales ya que el software tiene que ser ejecutado para ver su comportamiento. Debería utilizarse en conjunción con la verificación estática para proporcionar una cobertura de V&V total. Es una técnica de Verificación dinámica, donde el sistema se ejecuta con datos de prueba y se observa.

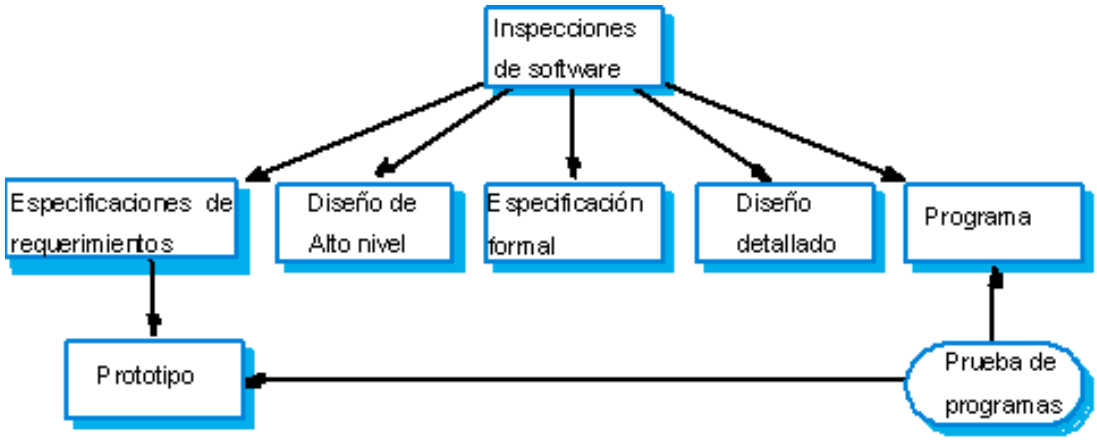

Figura 9. Técnica de Verificación dinámica (Inspección de Software).

Esta Figura muestra cómo las inspecciones y las pruebas son complementarias y no técnicas opuestas de Verificación, ambas deben utilizarse durante el proceso V&V. Las flechas indican las etapas en el proceso en que se pueden utilizar dichas técnicas. La inspección del software se puede utilizar en toda las etapas del proceso de desarrollo, pudiéndose solamente probar un sistema cuando está disponible un prototipo o una versión ejecutable del programa. Las pruebas de programas siempre serán la principal técnica de Verificación y Validación específicamente para requerimientos no funcionales ya que el software tiene que ser ejecutado para ver su comportamiento y debería utilizarse en conjunción con la Verificación estática para proporcional una cobertura de V&V total (17).

2.3.3. Tipos de pruebas, revisiones e inspecciones.

En este epígrafe se darán a conocer cuáles son los tipos de pruebas que se proponen aplicar a cada una de las actividades V&V propuestas en la estrategia:

1. *Prueba de Unidad:* esta prueba analiza los módulos de un programa, además se prueban todos los caminos de control para hallar los errores dentro del módulo. La prueba de unidad usa las técnicas de Caja Blanca.
2. *Prueba de Integración:* se toman los módulos que fueron probados por la prueba de unidad, y se elabora una estructura del programa de acuerdo con el diseño especificado, luego verifica los errores que se pueden detectar en la interacción

de los módulos, es decir, después de haberlos agrupados, ya que su objetivo principal es el diseño y la construcción de la arquitectura del software.

Existen dos tipos de Integración:

- *Integración no incremental:* se combinan todos los módulos por anticipado y se prueba todo el programa en conjunto.
- *Integración incremental:* el programa se construye y se prueba en pequeños segmentos.

3. *Prueba de Sistema:* analiza que todo esté de forma adecuada y que su funcionalidad al igual que el rendimiento del sistema estén correctamente. Esta prueba está conformada por una serie de pruebas diferentes que tiene como objetivo adiestrar al sistema basado en computadora. Algunas de estas son*:*
   - *Prueba de Funcionalidad:* valida que los requisitos funcionales (Casos de uso), la navegación, entrada de datos, procesamiento y obtención de resultados se realice de forma apropiada. Verifica además, que la implementación de las reglas del negocio sean las adecuadas (33).
   - *Prueba de Rendimiento o Carga:* se ejecuta para comprender el comportamiento de una aplicación ante una carga determinada. Esta carga puede ser el número de usuarios esperado ejecutando, o un número de transacciones durante un tiempo determinado. El resultado de esta prueba dará el tiempo de respuesta de todas las transacciones críticas.
   - *Prueba de Seguridad:* intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de accesos impropios (10).
   - *Prueba de Disponibilidad y Red:* verifica que la aplicación esté funcionando correctamente, cambiando la infraestructura de red al aplicar diferentes configuraciones y retardo. Valida que la solución esperada tenga la disponibilidad del sistema y no haya sido reducida.
   - *Prueba de Compatibilidad:* verifica el funcionamiento del sistema sobre diferentes componentes de software.
   - *Prueba de Estrés:* se refiere a cargas extremas, memoria insuficiente, no disponibilidad de servicio y de hardware o recursos compartidos limitados. Este tipo de prueba permite comprender mejor cómo y qué área del sistema

colapsarán, de este modo es posible planificar contingencias, actualizar el mantenimiento, planear y asignar recursos de antemano.

- *Prueba de Usabilidad:* verifica la estética, la consistencia de la interfaz de usuario y documentación.
- *Pruebas de Fiabilidad:* verifica la probabilidad de que el sistema funcione o desarrolle una determinada función, bajo condiciones prefijadas y durante un período de tiempo determinado.
- *Pruebas de Configuración:* verifica que el sistema funciona correctamente en diferentes configuraciones; por ejemplo, en diferentes configuraciones de red y en diferentes sistemas operativos. Para obtener los Casos de Prueba de las pruebas de configuración se pueden seguir las siguientes recomendaciones:
  - ✓ Definir al menos un Caso de Prueba que identifique cada configuración crítica.
  - ✓ Definir al menos un Caso de Prueba que compruebe que cada configuración tendrá problemas, es decir, pruebas negativas, por ejemplo, recursos insuficientes, configuraciones de red incorrectas y capacidad de hardware insuficiente.
- *Prueba de Instalación:* verifica que el sistema puede ser instalado en la plataforma del cliente y que el sistema funcionará correctamente cuando sea instalado. Hay que probar todos los escenarios que describen todos los casos posibles que se dan en la instalación.

4. *Prueba de Aceptación:* el objetivo de esta prueba es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Para permitir que el cliente valide todos los requisitos se realizan los siguientes tipos de pruebas:
   - *Prueba Alfa*: es realizada por un cliente, el software debe estar en su forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso, se llevan a cabo en un entorno controlado.

- *Prueba Beta*: la realizan los usuarios finales del software en los lugares de trabajo de los clientes. El desarrollador no está presente a la hora de la prueba. El cliente es el encargado de registrar todos los problemas detectados.

5. *Pruebas de Integridad*: son diseñadas para probar la robustez (resistencia a fallas) y el uso adecuado del lenguaje, sintaxis y uso de recursos. Este tipo de prueba puede aplicarse tanto a unidades como a integración de unidades.
6. *Pruebas de Regresión*: son una estrategia de prueba en la cual las pruebas que se han ejecutado anteriormente se vuelven a realizar en la nueva versión modificada, para asegurar la calidad después de añadir la nueva funcionalidad.
7. *Revisión de la documentación del negocio*: verifica que la documentación del negocio cumpla las necesidades de los usuarios y que es coherente.
8. *Revisión de la documentación y requisitos del sistema*: los requisitos del software deben tener una explicación clara, precisa y completa del problema que facilite el análisis de errores y la generación de casos de pruebas. Un asunto de gran importancia es asegurar la corrección, coherencia y exactitud de los requisitos. Revisará el documento de especificación de requerimientos: con la lista de chequeo general del documento y la lista de chequeo de requerimiento.
9. *Revisión al diseño*: evaluar los elementos de diseño (Descripción del Diseño de software y Documento de Diseño de la interfaz para corrección, coherencia, integridad, precisión, legibilidad y la comprobabilidad).
10. *Las inspecciones a programas*: se realizan con el propósito de descubrir anomalías y defectos, no requiere la ejecución de un sistema por lo que debe utilizarse antes de la implementación. Pueden estar aplicados a cualquier representación del sistema (requerimiento, diseño, configuración, dato, pruebas de datos). Durante una inspección, a menudo se utiliza una lista de comprobación de errores de programas comunes para centrar el análisis. Además se puede decir que las inspecciones son una forma de análisis estático:
11. *Walkthrough*: técnica de análisis estático en la que un diseñador o programador dirige miembros del equipo de desarrollo y otras partes interesadas a través de un producto de software y los participantes formulan preguntas y realizan

comentarios acerca de posibles errores, violación de estándares de desarrollo y otros problemas (34).

12. *Técnica de diseño de casos de prueba*: las técnicas de diseño de casos de prueba se utilizan con el objetivo de facilitar la búsqueda de errores con el mínimo consumo de tiempo y recursos garantizando, de esta forma, la obtención de un producto correcto.
13. *Casos de pruebas*: es una forma de especificar cómo probar el sistema, incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que ha de probarse.
14. *Prueba de Caja Blanca*: la prueba de Caja Blanca, denominada, a veces, *prueba de Caja de Cristal* es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de Baja Blanca, según el ingeniero del software puede obtener casos de prueba que:
    - garanticen la ejecución, por lo menos una vez, de todos los caminos independientes de cada módulo;
    - ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas;
    - ejecuten todos los bucles en sus límites y con sus límites operacionales; y
    - ejerciten las estructuras internas de datos para asegurar su validez.

    Existen diversas técnicas de Caja Blanca, entre ellas se encuentran:

    - *Prueba del camino básico o cubrimiento*: primeramente se genera un grafo de flujo o grafo del programa, se halla su complejidad ciclomática definiendo el número de caminos independientes del conjunto básico de un programa con el objetivo de ejecutar, al menos una vez, cada sentencia del programa.
    - *Prueba de Condición*: se centra en las pruebas de cada una de las condiciones lógicas contenida en el módulo de un programa, los tipos de componentes donde se pueden detectar errores son: Operadores lógicos, una variable lógica, un par de paréntesis, un operador racional o una

expresión aritmética. Además se pueden encontrar otros errores en dicho programa.

Estrategias de pruebas de condición para detectar errores en una condición:

- ✓ *Prueba de ramificación*: para una condición compuesta C, es necesario ejecutar, al menos una vez, la rama verdadera y falsa de C y cada condición simple de C.
- ✓ *Prueba de Dominio*: requiere la realización de tres o cuatro pruebas para una expresión relacional.

- *Prueba de Bucles*: se centra en la validez de las construcciones de bucles, estos se dividen en cuatro clases diferentes:
  - ✓ *Bucles Simples*: se le aplican una serie de pruebas donde "n" es el número máximo de pasos permitidos por el bucle:
    - pasar por alto totalmente el bucle
    - pasar una sola vez por el bucle
    - pasar dos veces por el bucle
    - hacer ***m*** pasos por el bucle con ***m*** < ***n***
    - hacer ***n*** - 1, ***n*** y ***n+l*** pasos por el bucle
  - ✓ *Bucles Anidados:*
    - Comenzar por el bucle más interior. Establecer o configurar los demás bucles con sus valores mínimos.
    - Llevar a cabo las pruebas de bucles simples para el bucle más interior, mientras se mantienen los parámetros de iteración (por ejemplo, contador del bucle) de los bucles externos en sus valores mínimos. Añadir otras pruebas para valores fuera de rango o excluidos.
    - Progresar hacia fuera, llevando a cabo pruebas para el siguiente bucle, pero manteniendo todos los bucles externos en sus valores mínimos y los demás bucles anidados en sus valores típicos.
    - Continuar hasta que se hayan probado todos los bucles.

- ✓ *Bucles concatenados:* Los bucles concatenados se pueden probar mediante los bucles simples, mientras cada uno de los bucles sea independiente del resto, cuando esto no sucede se usan los bucles anidados (bucles no independientes).
- ✓ *Bucles no estructurados:* esta clase de bucles se deben rediseñar para que se ajusten a las construcciones de programación estructurada (10).

15. *Prueba de Caja Negra*: también denominadas prueba de comportamiento, se centran en los requisitos funcionales del software, sin tener mucho en cuenta la estructura lógica interna del software.

- *Método de prueba basado en grafos*: es entender los objetos que se modelan en el software y las relaciones que conectan a estos objetos. Para llevar a cabo estos pasos, el ingeniero del software empieza creando un grafo -una colección de nodos que representan objetos; enlaces que representan las relaciones entre los objetos; pesos de nodos que describen las propiedades de un nodo (por ejemplo, un valor específico de datos o comportamiento de estado) y pesos de enlaces que describen alguna característica de un enlace.
- *Partición equivalente*: se basa en una evaluación de las clases equivalentes para una condición de entrada que estas representan un conjunto de estados válidos y no válidos.

  Estas clases poseen las siguientes directrices (10):

  Si una condición de entrada:

  - ✓ Caracteriza un rango, se define una clase equivalente válida y dos no válidas.
  - ✓ Requiere un valor específico, define una clase de equivalencia válida y dos no válidas.
  - ✓ Especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
  - ✓ Es lógica, se define una clase de equivalencia válida y una no válida.

Tabla 5. Clases equivalentes y sus condiciones de entradas.

	Clases Equivalentes	
Condición de entrada	**Válida**	**No válida**
Un Rango	1	2
Un valor específico	1	2
Un miembro de un conjunto	1	1
Es lógica	1	1

- *Análisis de Valores Límites (AVL):* es una técnica de diseño que complementa a la partición equivalente, donde en vez de centrarse en las condiciones de entradas, obtiene casos de pruebas también para el campo de salida.

  Directrices para identificar casos de pruebas (35):

  - ✓ Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justo por debajo y justo por encima de a y b, respectivamente.
  - ✓ Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.
  - ✓ Aplicar las directrices anteriores a las condiciones de salida.
  - ✓ Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.
- *Prueba de mano a mano o comparación*: sucede cuando se desarrollan varias versiones de un mismo software, se deben probar todas las versiones con los mismos datos de pruebas, para ver si todos proporcionan una salida idéntica, si esto ocurriera se puede decir que todas la versiones son correctas, de lo contrario se analizan todas las aplicaciones para determinar el error responsable de la diferencia en una o más versiones.
- *Prueba de la tabla ortogonal*: puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño (36).

2.4. Estrategia

Para ir eliminando los errores o cualquier otro problema que se presente durante la creación del producto, se propone llevar el proceso de V&V de manera paralela al ciclo de vida de software.

Quedando definida la siguiente Estrategia:

- V&V de la Conceptualización.
- V&V de los Requerimientos.
- V&V de Diseño.
- V&V de Implementación.
- V&V de Prueba.
- V&V de Instalación y Chequeo.
- V&V de Mantenimiento.

2.4.1. Roles

En este epígrafe se conocerá cuáles son los responsables a realizar cada una de estas actividades de V&V (37).

- *Administrador de Prueba*: Es el responsable del éxito de la prueba, involucra al defensor de prueba y calidad, planificación y administración de recursos, resolución de problemas que impiden las pruebas.
- *Analista de Prueba:* Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de la prueba y el resultado en cada ciclo de prueba y evaluando la calidad total experimentada como un resultado de las actividades de prueba. Lleva la responsabilidad para representar apropiadamente las necesidades de los stakeholder[14] que no tienen representación regular y directa en el proyecto.
- *Diseñador de prueba*: Es el responsable de definir el método de prueba y asegurar su implementación exitosa, incluye identificación de técnicas

[14] Ver Anexo 1: Glosario de Témino del Manual de Prueba para Ingeniero de software.

apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas.

- *Probador:* Es el responsable durante las actividades principales de las pruebas, el cual incluye la conducción de las pruebas necesarias y el registro del resultado de la prueba.
- *Ingeniero de Prueba:* (Ver epígrafe 1.6)

2.4.2. Desarrollo

2.4.2.1. V&V de la Conceptualización.

La actividad V&V del negocio delimita una solución específica para resolver el problema del usuario. Durante el negocio, la arquitectura del sistema está seleccionada, los requisitos del sistema son asignados a hardware, software y componentes de interfaz de usuario. El objetivo de la actividad V&V del negocio es validar y verificar la asignación de los requisitos del sistema, la solución elegida, garantizar que no existan falsas suposiciones y además que sean incorporados en la solución.

- Tareas a realizar por el Ingeniero de Prueba.

1- *Plan de las Tareas de V&V*

Entradas:

- ✓ Informe de tareas.
- ✓ Evaluación de la documentación

Salidas:

- ✓ Modelos y planes de desarrollo.
- ✓ Definir/Elegir Modelo(s) de ciclo de vida.
- ✓ Plan de prueba.

2- *Evaluación de la documentación.*

Entradas:

- ✓ Plan de desarrollo de proveedores y anexos.
- ✓ Necesidades de los usuarios.

✓ Las necesidades de adquisición.

Salidas:

✓ Informe Tarea.

✓ Evaluación de la Documentación del Negocio

2.4.2.2. V&V de Requerimiento.

La actividad V&V de Requerimiento responde al análisis de los requisitos funcionales y de rendimiento, las interfaces externas al software, los requisitos de cuantificación de seguridad, los factores humanos, las definiciones de datos, la documentación del usuario para el software, la instalación y aceptación, las operaciones del usuario, los requisitos de ejecución y requisitos de mantenimiento de usuario. El objetivo de V&V de Requerimiento es garantizar la exactitud, comprobabilidad, capacidad de prueba y la coherencia de los requisitos del sistema de software.

- Tareas a realizar por el Ingeniero de Prueba.

1- *Análisis de la Interfaz.*

Entradas:

✓ Documentación del negocio.

✓ Especificación de requisitos del software. (SRS)

✓ Especificación de requisitos de la interfaz. (IRS)

Salidas:

✓ Tarea Informe(s) – Análisis de la Interfaz

✓ Informe(s) anomalía.

2- *Plan V&V de Generación de Pruebas y Verificación del Sistema.*

Entradas:

✓ La Documentación del negocio. (los requisitos del Sistema)

✓ Especificación de requisitos del software.

✓ Especificación de requisitos de la interfaz.

✓ La Documentación del usuario.

✓ El Plan de Prueba del Sistema.

Salidas:

- ✓ Informe de V&V del Plan de Prueba del Sistema.

3- *Plan V&V de Generación de Pruebas y de Verificación de Aceptación.*

Entradas:

- ✓ La documentación del Sistema.
- ✓ Especificación de requisitos del software.
- ✓ Especificación de requisitos de la interfaz.
- ✓ La documentación del usuario del sistema.
- ✓ Plan de Prueba de Aceptación.

Salidas:

- ✓ Informe del Plan de Prueba de Aceptación.

4- *Revisiones e Inspecciones de los Requerimientos del Software.*

Entradas:

- ✓ La documentación del sistema.
- ✓ Especificación de requisitos del software.
- ✓ Especificación de requisitos de la interfaz

Salidas:

- ✓ Informe de Evaluación de requisitos de software.

2.4.2.3. V&V del Diseño.

La actividad de V&V del Diseño transforma los requisitos del software en una arquitectura y en un diseño detallado para cada componente de software. El diseño incluye bases de datos e interfaces de sistema (por ejemplo, el hardware, el operador / usuario, los componentes de software, y de los subsistemas). El objetivo de V&V del Diseño es demostrar que es correcto, preciso y la transformación completa de los requisitos del software y que no se introducen las características no deseadas.

- Tareas a realizar por el Ingeniero de Prueba.

1- *Evaluación del Diseño del Software*

Entradas:

- ✓ Especificación de requisitos del software.

- ✓ Especificación de requisitos de la interfaz
- ✓ Documentación del Diseño de software.(SDD)
- ✓ Documentación del Diseño de la interfaz.(IDD)
- ✓ Estándares de Diseño (normas, métricas y convenios).

Salidas:

- ✓ Informe de Evaluación del Diseño.

2- *Análisis de la Interfaz.*

Entradas:

- ✓ Documentación del Negocio (requisitos del sistema).
- ✓ Especificación de requisitos del software.
- ✓ Especificación de requisitos de la interfaz
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.

Salidas:

- ✓ Informe del Análisis de la Interfaz.

3- *Plan V&V de Generación de prueba y Verificación de Componente.*

Entradas:

- ✓ Especificación de requisitos del software.
- ✓ Especificación de requisitos de la interfaz
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ Plan de Prueba de componente.

Salidas:

- ✓ Informe V&V del Plan de Prueba de componente al software.

4- *Plan V&V de Generación de Prueba y Verificación de Integración.*

Entradas:

- ✓ Especificación de requisitos del software.
- ✓ Especificación de requisitos de la interfaz.
- ✓ Documentación del Diseño de software.

- ✓ Documentación del Diseño de la interfaz.
- ✓ Plan de prueba de Integración.

Salidas:

- ✓ Informe V&V del Plan de Prueba de integración.

5- *Plan V&V de Generación de Prueba y Verificación del Diseño.*

Entradas:

- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ Documentación del usuario.
- ✓ Prueba de Diseño.
- ✓ Plan de prueba.

Salidas:

- ✓ Componente V&V Diseño de las pruebas.
- ✓ Integración V&V Diseño de las pruebas.
- ✓ Sistema V&V Diseño de las pruebas.
- ✓ Aceptación V&V Diseño de las pruebas.

6- *Revisión e Inspección de la arquitectura.*

Entradas:

- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.

Salidas:

- ✓ Diseño de la arquitectura del software.

2.4.2.4. V&V de Implementación.

En la actividad de V&V de Implementación se transforma el diseño del sistema en código, estructuras de base de datos y equipo relacionados con las representaciones ejecutables. La actividad de V&V de Implementación de las direcciones de codificación y prueba de software, incluyendo la incorporación de productos de software reutilizado. El objetivo de V&V de Implementación es verificar y validar que estas transformaciones son correctas, exactas y completas.

- Tareas a realizar por el Ingeniero de Prueba.

1- *Evaluación del Código Fuente y documentación.*

Entradas:

- ✓ El Código Fuente.
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ Normas de codificación (normas, prácticas, restricciones del proyecto, y convenios).
- ✓ Documentación del usuario.

Salidas:

- ✓ Informe de las tareas del Código Fuente, la Evaluación de la documentación y la Evaluación del código.

2- *Análisis de la Interfaz*

Entradas:

- ✓ La Documentación de los Requisitos del Sistema.
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ El Código Fuente.
- ✓ La Documentación del usuario.

Salidas:

- ✓ Informe de las tareas del Análisis de la interfaz.

3- *V&V Generación de casos de prueba y Verificación.*

Entradas:

- ✓ Especificación de requisitos del software.
- ✓ Especificación de requisitos de la interfaz.
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ La Documentación de usuario.
- ✓ El Diseño de la pruebas.

- ✓ Los Casos de pruebas.

Salidas:

- ✓ Informe de Componente V&V de los casos de pruebas.
- ✓ Informe de Integración V&V de los casos de pruebas.
- ✓ Informe de Sistema V&V de los casos de pruebas.
- ✓ Informe de Aceptación V&V de los casos de pruebas.

4- *V&V Generación de Procedimientos de Pruebas y Verificación.*

Entradas:

- ✓ Especificación de los requisitos del software.
- ✓ Especificación de los requisitos de la interfaz
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ La Documentación de usuario.
- ✓ Los Casos de pruebas.
- ✓ El Diseño del procedimiento de las pruebas.

Salidas:

- ✓ Informe de Componente V&V del procedimiento de pruebas.
- ✓ Informe de Integración V&V del procedimiento de pruebas.
- ✓ Informe de Sistema V&V del procedimiento de pruebas.

5- *V&V Prueba de Ejecución y Verificación de Componentes.*

Entradas:

- ✓ El Código fuente.
- ✓ El Código ejecutable.
- ✓ Documentación del Diseño de software.
- ✓ Documentación del Diseño de la interfaz.
- ✓ Componente de los planes de prueba.
- ✓ Los Procedimientos de prueba de componente.
- ✓ Los Resultados de prueba de componente.

Salidas:

- ✓ Informe de los Resultados de las pruebas.

2.4.2.5. V&V de Prueba.

La actividad de V&V de Prueba incluye al software las pruebas de integración, las pruebas de validación, pruebas de integración de sistemas y las pruebas de calificación del sistema. El objetivo V&V de Prueba es garantizar que los requisitos del software y los requisitos del sistema asignados al software son validados por la ejecución de la integración, sistema, y pruebas de aceptación.

- Tareas a realizar por el Ingeniero de Prueba.

1- *V&V Generación de Procedimientos de Pruebas y Verificación de la Aceptación.*

Entradas:

- ✓ Documentación del Diseño de la interfaz.
- ✓ Componente de los planes de prueba.
- ✓ El Código fuente.
- ✓ Documentación de usuario.
- ✓ El Plan de Prueba de Aceptación.
- ✓ La Aceptación de los Procedimientos de Prueba.

Salidas:

- ✓ Aceptación V&V de los Procedimientos de Prueba.

2- *V&V Ejecución y Verificación de las Pruebas de Integración.*

Entradas:

- ✓ El Código fuente.
- ✓ El Código ejecutable.
- ✓ El Plan de Prueba de Integración.
- ✓ Los Procedimientos de Prueba de Integración.
- ✓ Los Resultados de Prueba de Integración.

Salidas:

- ✓ Informe del Resultado de las Pruebas.

3- *V&V Prueba de Ejecución y Verificación del Sistema.*

Entradas:

- ✓ El Código fuente.

- ✓ El Código ejecutable.
- ✓ El Plan de Prueba del Sistema.
- ✓ Los Procedimientos de Prueba del Sistema.
- ✓ Los Resultados de Prueba del Sistema.

Salidas:

- ✓ Informe del Resultado de las Pruebas.

4- *V&V Prueba de Ejecución y Verificación de Aceptación.*

Entradas:

- ✓ El Código fuente.
- ✓ El Código ejecutable.
- ✓ Documentación del usuario.
- ✓ El Plan de Prueba de Aceptación.
- ✓ Los Procedimientos de Prueba de Aceptación.
- ✓ Los Resultados de Prueba de Aceptación.
- ✓ V&V Resultados de las tareas.

Salidas:

- ✓ Informe del Resultado de las Pruebas.

5- *V&V de la Documentación.*

Entradas:

- ✓ Documentación de usuario.
- ✓ Documentación del Diseño de Software.
- ✓ Documentación del Diseño de la Interfaz.
- ✓ Documentación del Sistema.
- ✓ Documentación del Negocio.

Salidas:

- ✓ Plan de la Documentación.
- ✓ Documentos preparados.
- ✓ Documentos producidos.
- ✓ Documentos Modificados.

2.4.2.6. V&V Instalación y Chequeo.

En la actividad V&V Instalación y Chequeo, el producto de software está instalado y probado en el entorno de ambiente. El objetivo de V&V Instalación y Chequeo es verificar y validar la exactitud de la instalación del software en el entorno de ambiente.

- Tareas a realizar por el Ingeniero de Prueba.

1- *Auditoría de Instalación de Configuración.*

Entradas:

- ✓ El Paquete de la instalación (por ejemplo, Código Fuente, Código Ejecutable, la Documentación de Usuario, SDD, IDD, SRS, IRS, la Documentación del Negocio, instalación de procedimientos específicos de cada sitio, parámetros, los ensayos con la instalación, Configuración y Gestión de datos)

Salidas:

- ✓ Auditoría de la Instalación de la Configuración.

2- *Chequeo de instalación.*

Entradas:

- ✓ Documentación de usuario
- ✓ Instalación del paquete.

Salidas:

- ✓ Informe del Chequeo de la instalación.

3- *V&V Informe Final de Generación.*

Entradas:

- ✓ Informe V&V del Resumen de actividades.

Salidas:

- ✓ Informe V&V último.

2.4.3. Mantenimiento

2.4.3.1. V&V de Mantenimiento.

En la actividad de V&V de Mantenimiento pueden derivarse las modificaciones del sistema de los requisitos específicos para corregir los errores del software (por ejemplo, correctivo); para adaptar el cambio (por ejemplo, adaptable); o para responder a las demandas adicionales o perfeccionamientos del usuario.

El objetivo de V&V de Mantenimiento se evaluarán los cambios planteados y sus efectos en el software, la evaluación de las anomalías que se descubran durante el funcionamiento, requisitos de migración, evaluar los requisitos de jubilación y volver a realizar las tareas de V&V.

- Tareas a realizar por el Ingeniero de Prueba.

1- *Revisión del SVVP*

Entradas:

- ✓ Plan de V&V.
- ✓ Los Cambios Aprobados.
- ✓ Paquete de instalación.
- ✓ Planes de desarrollo de proveedores y horarios.

Salidas:

- ✓ Actualización del Plan V&V

2- *Evaluación de la migración*

Entradas:

- ✓ Evaluación del cambio

Salidas:

- ✓ Los cambios aprobados.

3- *Evaluación del cambio*.

Entradas:

- ✓ Los cambios aprobados.

Salidas:

- ✓ Evaluación de las migraciones.

2.5. Modelo W

La propuesta del modelo que se realiza es una modificación al modelo W, donde desde el principio se representa la iniciación de las actividades de Verificación y Validación definidas en la Estrategia, reflejando así la relación que existe entre el ciclo de vida del software y el proceso de prueba. Siendo la siguiente figura el modelo propuesto, que en conjunto con el manual y la Estrategia potenciarán la calidad del producto final y el proceso de pruebas llevado a cabo en cada elaboración de software.

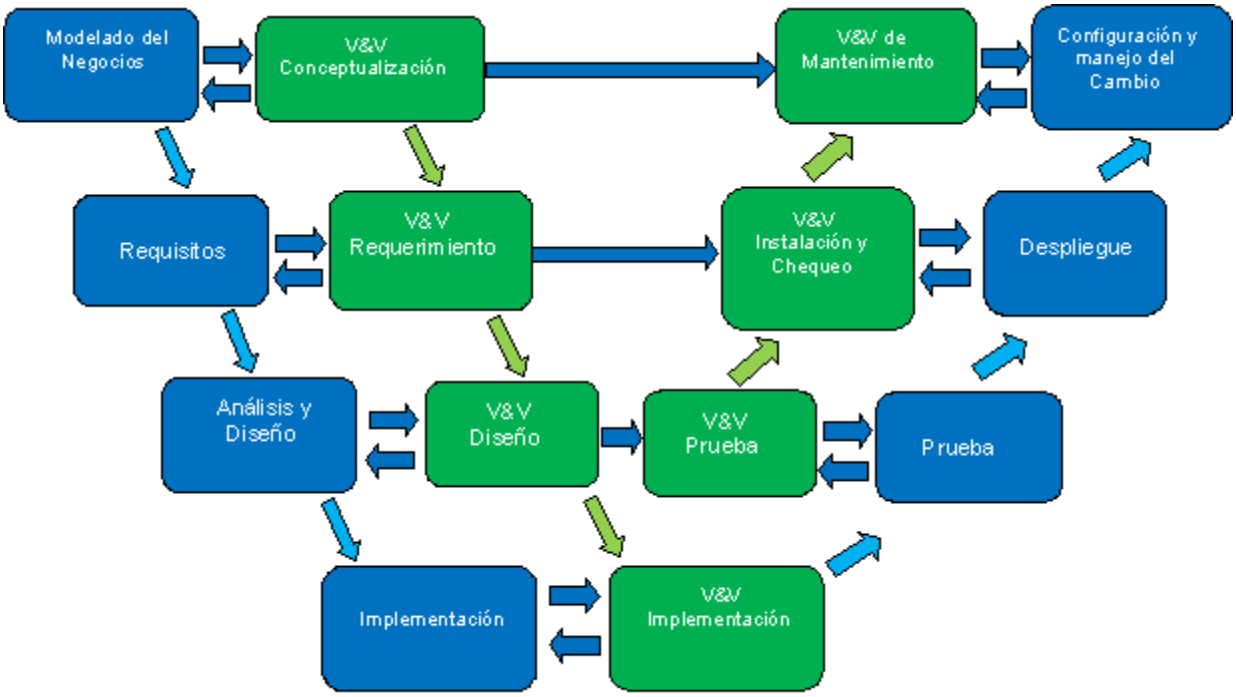

Figura 10. MIPFI-SW Manual de Pruebas para Ingeniero de software en la Facultad de Informática.

2.6. Glosario de términos

Se brinda un glosario de términos con la finalidad de enriquecer y facilitar el entendimiento del manual propuesto (Ver Anexo 1).

2.7. Conclusiones Parciales

Las pruebas constituyen un aspecto esencial dentro del ciclo de vida del software porque son una parte fundamental de este proceso y se realizan de forma conjunta. Se hizo necesario plasmar un Manual donde se archiven todas las actividades elaboradas por el Ingeniero de Prueba, logrando así gran eficiencia en el producto, evitando que este sea rechazado al final del ciclo de desarrollo. En este proceso se estableció el concepto de ciclo de vida de un software y los tipos de pruebas que miden los objetivos y aquellas metas fundamentales; también se tuvo en cuenta las actividades de V&V para ejecutar las pruebas y obtener resultados satisfactorios.

Capítulo 3 –Validación del Manual de Prueba para el Ingeniero de software.

3.1. Introducción.

En este capítulo se hará un estudio acerca de la factibilidad del manual propuesto. Se utilizarán cincos criterios para evaluar los conocimientos destacados en dicho manual y qué efectos proporciona a los Ingenieros de software. Además de los resultados afrontados después de este riguroso análisis.

3.2. Proceso de evaluación del contenido del Manual.

Para poder realizar el proceso de evaluación se hizo énfasis en dos grandes paradigmas del enfoque práctico citado por Pérez Gómez, estos son el modelo de evaluación experimental y el cualitativo. Estos modelos ayudan a evaluar el Manual de Prueba para Ingenieros de software en la Facultad de Informática de la UCF a partir de su diseño y práctica profesional que deben ser afrontados para demostrar que la información que presenta está relacionada con los ingenieros de pruebas. Este procedimiento proporciona que se haga un diagnóstico crítico en cuanto al contenido y algunos elementos que pudieran utilizarse respecto a los temas del Manual en lo que se refiere a conocimientos, habilidades y valores.

Para poder evaluar el contenido de un manual existen criterios, pautas, puntos de vista y concepciones, que analizan profundamente el contenido de una ciencia, procesos, programas y libros. Es necesario conocer que todo contenido aporta ideas, habilidades y valores desde diferentes ángulos. En este caso se aplicarán los cinco criterios expuestos por Hernández Sampieri (2003) en el libro de Metodología de la Investigación para valorar el contenido del Manual de prueba para ingenieros de software en la Facultad de Informática de la UCF (38).

3.2.1. Criterios para la evaluación del contenido del Manual

1. Capacidad de descripción, explicación, predicción.
2. La consistencia lógica.
3. La perspectiva.

4. La heurística.
5. Parsimonia.

3.2.1.1. Criterio de Capacidad de descripción, explicación, predicción:

Describir implica: definir las condiciones en que se presenta y las distintas maneras en que puede manifestarse. Además en el aspecto de descripciones se evaluará el entendimiento del contenido para verificar que se ve apoyado por la práctica y si la teoría propuesta se explica a través de datos y referencias.

Predicción, está asociado con el segundo término de explicación, que depende de la evidencia empírica de las proposiciones de la teoría y pronósticos en la variedad del trabajo.

Cuanta más evidencia empírica apoye a la teoría de este primer criterio, mejor se podrá describir, explicar y predecir el contenido del Manual.

3.2.1.2. Criterio de Consistencia lógica.

Verifica que el contenido del Manual debe ser lógicamente consistente. Además debe existir una interrelación entre todos los conocimiento presentados, ser mutuamente excluyentes (no puede haber repetición o duplicación) y no caer en contradicciones internas o incoherencias.

3.2.1.3. Criterio de la Perspectiva.

Se analizará si el Manual refiere al nivel de generalidad. Este conocimiento posee más perspectiva mientras mayor cantidad de fenómenos explique y mayor número de aplicaciones admita.

3.2.1.4. Criterio de la Heurística.

Este criterio evaluará en el Manual, la capacidad que tiene de generar nuevas interrogantes y descubrimientos. Esto hace posible que la búsqueda de nuevos conocimiento permita que exista un avance en la ciencia.

3.2.1.5. Criterio de Parsimonia.

Este criterio evalúa que el contenido del Manual sea simple, sencillo y no debe ser un requisito, sino una cualidad deseable. Además resalta la asequibilidad y no la superficialidad.

3.3. Análisis de los instrumentos utilizados para aplicar los criterios de evaluación.

3.3.1. Procedimiento estadístico.

Para la recopilación de los datos se utilizó un método empírico de investigación, por medio de preguntas escritas organizadas en un formulario impreso, aplicado a los especialistas, donde se midieron los cincos criterios que fueron diseñados cumpliendo los requisitos de redacción, motivación, preguntas claras y secuencia lógica.

Se evaluó en una escala de 5 alternativas de respuestas donde se mide la importancia del criterio desde “Toral desacuerdo” hasta “Total acuerdo”.

Para procesar la información se utilizó el paquete estadístico SPSS vs 15.0 realizando un análisis descriptivo de dicha información recogida en los cuestionarios mencionados anteriormente. En el Anexo 2 se adjunta un ejemplar de la encuesta aplicada.

Se definieron variables de acuerdos a los ítem relacionados en el cuestionario aplicado, para procesar frecuencialmente y descriptivamente la información obtenida por los mismos.

El SPSS facilita determinar la fiabilidad del cuestionario, mediante el coeficiente Alpha de Cronbach. La fiabilidad se refiere al grado de puntuación alcanzada en las diferentes preguntas del cuestionario y si están altamente interrelacionadas. Este coeficiente oscila entre 0 y 1. Está basada en la consistencia interna de la fiabilidad de la escala, mientras más cercano a la unidad la fiabilidad será superior.

El coeficiente alpha obtenido fue 0.739 lo que permitió considerar luego del análisis, que las puntuaciones del cuestionario, están adecuadamente interrelacionadas.

Para saber el acuerdo estadístico entre los evaluadores se realizó la prueba no paramétrica W de Kendall o también llamada de concordancia, donde se complementa el análisis anterior.

Dicha prueba contrasta la hipótesis H_0 (Nula), que no hay acuerdo entre los evaluados, contra la hipótesis H_1 (Alternativa), que sí se considera que hay acuerdo entre los evaluados. Luego de haber declarado las hipótesis es necesario decidir cuál es la aceptada. Se analiza el nivel de significación prefijado y se compara con la significación asintótica del estadígrafo que brinda el SPSS. Si la significación asintótica es menor que el nivel de significación se acepta la hipótesis alternativa.

3.3.2. Análisis de los resultados del cuestionario aplicado a los especialistas.

Para el procesamiento de los datos según el cuestionario se definieron las siguientes variables:

X1: Redacción.
X2: Lenguaje adecuado.
X3: Capacidad de descripción, explicación y predicción.
X4: Consistencia lógica.
X5: Motivación.
X6: Confiabilidad Psicopedagógica.
X7: Expectativas.

A estas variables se le asignaron etiquetas de valores correspondientes a las diferentes alternativas de cada pregunta.

Se encuestaron 12 especialistas, donde 3 son profesores vinculados a la Ingeniería de software y los demás de la Empresa Desoft, vinculados al área de producción de software.

- Criterio 1 capacidad de descripción, explicación y predicción:

El 58.3% de los especialistas consideró una puntuación de 5, mientras que el 33.3% otorgó una puntuación de 4 y el resto de 3, por lo que se considera que existe coherencia entre los objetivos del manual.

Criterio#1	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Capacidad de descripción, explicación y predicción	7	58,3	4	33.3	1	8.3			

- Criterio 2 Consistencia lógica:

El 75% de los especialistas evaluados consideró que el contenido del manual posee una correcta relación entre la teoría y la práctica otorgando una puntuación de 5 y el resto de 4 en un 25%. Se considera como satisfactorio este criterio del manual. Según la opinión de los especialistas es el mejor evaluado.

Criterio#2	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Consistencia lógica	9	75	3	25					

- Criterio 3 perspectiva:

El 58,3% de los especialistas consideró que el manual satisface las expectativas dando una puntuación de 5, mientras que el 41.7% restante otorgó una puntuación de 4 por lo que se considera como satisfactorio. No se encontraron otras evidencias de este criterio en el manual.

Criterio#3	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Expectativa	7	58.3	5	41.7					

- Criterio 4 Heurística:

El 8,3% de los especialistas aportó una puntuación respecto a la motivación de 5, el 83,3% otorgó una puntuación de 4 y el 8,3% restante una puntuación de 3, siendo

este criterio dentro del manual, según la opinión de los especialistas, el peor evaluado, por lo que se considera como el menos satisfactorio.

El 58,3% de los especialistas otorgó una puntuación de 5 y el 41,7% de 4 respecto a la confiabilidad psicopedagógica, por lo que el manual posee una estrategia instructiva. Según la opinión de los especialistas, este criterio dentro del manual se considera satisfactorio.

El 58,3% de los especialistas, respecto al lenguaje adecuado, otorgó una puntuación de 5. El 33,3% otorgó una puntuación de 4 y el resto de 3, por lo que se considera como satisfactorio este criterio dentro del manual, según la opinión de los especialistas.

Criterio#4	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Motivación	1	8,3	10	83,3	1	8,3			
Lenguaje adecuado	7	58,3	5	33,3	1	8,3			
Confiabilidad psicopedagógica	7	58,3	8	41,7					

- Criterio 5 Parsimonia:

El 50% de los especialistas, respecto a la redacción, otorgó una puntuación de 5 y el resto de 4 considerando que las expresiones de las ideas son claras y precisas, por lo que se considera como satisfactorio este criterio dentro del manual, según la opinión de los especialistas.

Criterio#5	**Escala de importancia de la encuesta**								
	5	**%**	**4**	**%**	**3**	**%**	**2**	**%**	**n/r**
Redacción	6	50	6	50					

De acuerdo a la prueba no paramétrica de concordancia de W de Kendall, teniendo en cuenta las siguientes hipótesis H_0 (No hay acuerdo entre los especialistas) y H_1 (Hay acuerdo entre los especialistas), para poder decir cuál es la hipótesis que se debe aceptar, se ha tomado un nivel de significación de 0.05 que supera la significación asintótica del estadístico calculado (0,013), por lo que se puede concluir

que se acepta la hipótesis H_1 y por tanto existe acuerdo entre las opiniones dadas por los especialistas (Ver Anexo 3).

Por otra parte los rangos obtenidos en dicha prueba permiten ordenar los criterios según su importancia y quedan de la siguiente forma:

- Consistencia Lógica (4,92).
- Expectativas (4,33).
- Confiabilidad Psicopedagógica (4,21).
- Capacidad de descripción, explicación y predicción (4,08).
- Lenguaje adecuado (4,08).
- Redacción (4,04).
- Motivación (2,33).

Después de concluido el análisis de los resultados, se puede decir que la opinión de los especialistas en cuanto al manual fue concurrente en la mayoría de los aspectos, otorgándole una gran importancia al mismo, la puntuación estuvo entre 4 (de acuerdo) y 5 (total acuerdo), menos en algunos casos como en el lenguaje adecuado y la motivación que obtuvieron puntuaciones de 3. En cuanto a estas dos observaciones se pueden hacer mejoras para futuras modificaciones en el manual.

3.4. Conclusiones Parciales.

En este capítulo se tuvieron en cuenta los criterios para evaluar el Manual de Prueba para Ingenieros de software en la Facultad de Informática de la UCF. La recogida de la información fue mediante una encuesta, la cual fue procesada en el paquete estadístico SPSS, donde se obtuvieron resultados satisfactorios y se logró concluir que el manual está listo para ser utilizado por los desarrolladores de la Facultad de Informática de la UCF o por cualquier otra entidad que desee usarla. Las sugerencias dadas por los especialistas permitieron enriquecer el contenido del manual.

Conclusiones Generales

Teniendo en cuenta el análisis realizado en esta investigación, se puede arribar a las siguientes conclusiones:

- Luego de un análisis detallado y minucioso sobre el proceso de prueba para el desarrollo de software, so logró decidir que el empleo de la metodología RUP unido al estándar IEEE 1012 (Verificación y Validación) ha sido conveniente y productivo, porque se aplicaron las fases del ciclo de vida del software, así como las actividades y tareas a realizar en cada una de ellas que sirvieron como guía al proceso personal de prueba de la Facultad de Informática.
- Teniendo en cuenta las pautas trazadas, se tomó como base para la estrategia el Modelo W, debido a que poseía todos los requisitos necesarios para obtener un modelo sobre la modificación del anterior, el cual centra su atención en el ciclo de vida del software y en las actividades de Verificación y Validación(V&V).
- Con la confección de este Manual de prueba para Ingeniero de software en la Facultad de Informática, se lograron los objetivos propuestos porque proporcionó una guía bien planificada para verificar y validar sus productos y además facilitará el proceso personal de prueba llevadas a cabo en la Facultad de Informática de la UCF.

Recomendaciones

El desarrollo de esta investigación ha facilitado realizar un manual de pruebas para ingenieros de software en la Facultad de Informática de la UCF, que permitirá que se obtengan software con una mejor calidad, además se recomienda:

- Que el Manual se ponga a prueba estrictamente como está estructurado, para comprobar su efectividad.
- Realizar un estudio para verificar si el manual realizado puede ser adaptado a las metodologías ágiles y ver su efectividad.
- Extender su uso a todas las empresas que sean desarrolladoras de software.

Referencias bibliográficas

(1). GreenSQA. GreenSQA . *Software Quality Assurance.* [En línea] 2008. [Citado el: 2 de 5 de 2010.] http://grensqa.com//.

(2). Gustavo A, Valencia R. *Subcontratación del servicio de software.Empresa Especializada en Aseguramiento de Calidad de software.* pdf.

(3). Gómez, Liliana. *Eficiencia de la caraterización de técnicas de prueba en un contexto real de aplicación. PARQUESOFT.* 2005. pdf.

(4). Gómez Arenas, Liliana del Sol. *Metodología para Evaluar la Calidad de los Sistemas de información.* Octubre de 2004.

(5). Soto, Lauro. Como Obtener Clidad De Software, Métodos, Metodologia, Estandares. [En línea] http://www.mitecnologico.com/Main/ComoObtenerCalidadDeSoftwareMetodosMetodologiasEstandares.

(6). I-Sol. S.A Intelligent Solution. [En línea] Versión 5.1.003, 2008. http://www.i-sol.com.ar/pg005.html.

(7). Desoft. Desoft. Empresa nacional de Software. Cuba. [En línea] 5 de 1 de 2004. [Citado el: 25 de 4 de 2010.] http://www.desoft.com.

(8). Acuña, César Javier. *Prueba del Software. Ingeniería del software.Universidad Rey Juan Carlos.* 2008. 33 pág.

(9). Javier Gutiérrez. *Introducción al Proceso de Pruebas.* pág. 22, pdf.

(10). Pressman, R. *Ingeniería del Software. Un enfoque práctico.* 2005.

(11). IEEE 1012. *Draft Standard for software Verification and Validation. September 12.* 2004. pág. 108, pdf.

(12). Amber, Scott W. Ambysoft. Métodos de pruebas Orientadas a objetos para el Ciclo de Vida Completo (FLOOT). [En línea] 2009. http://www.ambysoft.com/essays/flootSpanish.html.

(13). Ferron María, José. Sistemas de programas(CI - 3711). Testing (Pruebas). [En línea] 22 de 6 de 2005. http://www.ldc.usb.ve/~teruel/ci3711/test1/index.html.

(14). Collado, Manuel. *Pruebas del software.Técnica de prueba del software. Estrategía de pruebas del software.* 2003.

(15). Moreno, Yuliet. *Propuesta del Manual del Ingeniero de Prueba para el proceso de prueba de la Universidad de las Ciencias Informática.* Cienfuegos : s.n., 2008.

(16). Myers, G. *" Fase de Elaboración. FT Prueba (procedimiento genérico y aplicación de algunos tipos de pruebas).".* 2005.

(17). Drake, José M. Ingeniería software. 4to de Física.Verificación y Validación. [En línea] 2009. http://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion.pdf.

(18). Anaya, Sandra Lorena. RUP vs XP. [En línea] [Citado el: 30 de 5 de 2010.] url:http://www.iered.org/archivos/Grupo_Vultur/Seminario/3-mecs/sesion3/RUP-Vs-XP.pdf.

(19). Letelier, P. *Rational Unified Process(RUP).* Departamento de Sistema Informáticos y Computación, Universidad Politécnica de Valencia. pág. 18, doc.

(20). Mendoza Sánchez, María A. Metodología de Desarrollo de Software. [En línea] 7 de Junio de 2004. [Citado el: 26 de 5 de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

(21). Molpeceres, Alberto. Metodología de desarrollo: RUP, XP y FDD. [En línea] v1.0, 15 de 12 de 2003.

(22). Díaz Flores, Miriam Milagro. "Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia...". [En línea] url:http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

(23). ISO 9126. *Calidad en la Industria del software.* 1992.

(24). ISO 9000 3. Laboratorio de Sistema de Informática. [En línea] [Citado el: 20 de 5 de 2010.] http://users.dsic.upv.es/asignaturas/facultad/lsi/trabajos/102000.doc.

(25). López Quesada, Juan Antonio. *05BM- Fundamentos de Ingeniería del software. Tema 6. El proceso de sofware. Paradigma del ciclo de vida.* Informática y Sistema. 2007. pág. 75, pdf.

(26). ISO/IEC 12 207. *Técnología de la Información. Proceso del ciclo de vida del software.* Lima, Perú : 2 da Edición, 2006. pág. 194, pdf.

(27). Glossary of Testing. Standard glossary of term used in Software Testing. Version 1.1(dd. September, 29th 2005). [En línea] Erick Van Veenendaal.

(28). Knetia Software for business solutions. Metodología de Pruebas. [En línea] 2007. http://sp.kynetia.com/calidad/metodologia-de-pruebas.html.

(29). Deming, W Edwards. *Capitulo 2. Proceso Personal de Software.* pág. 12, pdf.

(30). Calvo Manzano, Bayona. *Team Software Process(TSP): Mejoras en la estimación, calidad y profundidad de los equipos en la Gestión del software.* Lenguaje y Sistema Informáticos e Ingeniería del software.Facultad de informática. Madrid, España : s.n., 2007.

(31). IEEE Std 610. *Standard Glossary of Software Engineering Terminology.* 12-1990. pág. 93, pdf.

(32). Ian Sommerville. *Ingeniería de software.Séptima Edición.* Madrid: PEARSON EDUCACIÓN, S.A.,. : s.n., 2005. pág. 712.

(33). Pérez Lamandra, Beatriz. *Qestión de las Pruebas Funcionales.* Universidad de República Montevideo. Uruguay : s.n., 2007.

(34). Std 1028. *IEEE Satandard for Software Review.* New York: Software Engenieering Standards Committee : s.n., 1997.

(35). Pressman, R. *Ingeniería del Software.Un enfoque práctico.* 2002.

(36). Grupo ARQUISOFT. [En línea] Rojas, Johanna; Barrios, Emilio ;, 2007. [Citado el: 1 de 6 de 2010.]

http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

(37). Letelier. *Sexta Parte. Prueba.Fase en que se Desarrolla.* pág. 11, doc.

(38). Davila Cabo de Villa, Evangelina. *Evaluación del contenido del Manual para Enfermeros Auxiliares de Anestesia.* Cienfuegos : s.n., 2007.

(39). Cueva Lovelle, Juan Manuel. *Calidad del Software.* Informática, Universidad de Oviedo. España : s.n. pág. 12, pdf.

(40). ISO 9001. *Sistema de gestión de la calidad- Requisitos.* Ginebra, Suisa; : Impreso en la Secretaría Central de ISO, 2000. pág. 34 pág.

(41). Fillottrani, Pablo R. *Calidad en el desarrollo del software. Modelo de calidad del software.* Ciencia e Ingeniería de la Computación, Universidad Nacional del Sur. 2007.

Bibliografía

1. GreenSQA. GreenSQA . *Software Quality Assurance.* [Online] 2008. [Cited: 5 2, 2010.] http://grensqa.com//.

2. Gustavo A, Valencia R. *Subcontratación del servicio de software.Empresa Especializada en Aseguramiento de Calidad de software.* pdf.

3. Gómez, Liliana. *Eficiencia de la caraterización de técnicas de prueba en un contexto real de aplicación. PARQUESOFT.* 2005. pdf.

4. Gómez Arenas, Liliana del Sol. *Metodología para Evaluar la Calidad de los Sistemas de información.* Octubre de 2004.

5. Soto, Lauro. Como Obtener Clidad De Software, Métodos, Metodologia, Estandares. [Online] http://www.mitecnologico.com/Main/ComoObtenerCalidadDeSoftwareMetodosMetodologiasEstandares.

6. I-Sol. S.A Intelligent Solution. [Online] Versión 5.1.003, 2008. http://www.i-sol.com.ar/pg005.html.

7. Desoft. Desoft. Empresa nacional de Software. Cuba. [Online] 1 5, 2004. [Cited: 4 25, 2010.] http://www.desoft.com.

8. Acuña, César Javier. *Prueba del Software. Ingeniería del software.Universidad Rey Juan Carlos.* 2008. 33 pág.

9. Javier Gutiérrez. *Introducción al Proceso de Pruebas.* p. 22, pdf.

10. Pressman, R. *Ingeniería del Software. Un enfoque práctico.* 2005.

11. IEEE 1012. *Draft Standard for software Verification and Validation. September 12.* 2004. p. 108, pdf.

12. Amber, Scott W. Ambysoft. Métodos de pruebas Orientadas a objetos para el Ciclo de Vida Completo (FLOOT). [Online] 2009. http://www.ambysoft.com/essays/flootSpanish.html.

13. Ferron María, José. Sistemas de programas(CI - 3711). Testing (Pruebas). [Online] 6 22, 2005. http://www.ldc.usb.ve/~teruel/ci3711/test1/index.html.

14. Collado, Manuel. *Pruebas del software.Técnica de prueba del software. Estrategía de pruebas del software.* 2003.

15. Moreno, Yuliet. *Propuesta del Manual del Ingeniero de Prueba para el proceso de prueba de la Universidad de las Ciencias Informática.* Cienfuegos : s.n., 2008.

16. Myers, G. *" Fase de Elaboración. FT Prueba (procedimiento genérico y aplicación de algunos tipos de pruebas).".* 2005.

17. Drake, José M. Ingeniería software. 4to de Física.Verificación y Validación. [Online] 2009. http://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion.pdf.

18. Anaya, Sandra Lorena. RUP vs XP. [Online] [Cited: 5 30, 2010.] url:http://www.iered.org/archivos/Grupo_Vultur/Seminario/3-mecs/sesion3/RUP-Vs-XP.pdf.

19. Letelier, P. *Rational Unified Process(RUP).* Departamento de Sistema Informáticos y Computación, Universidad Politécnica de Valencia. p. 18, doc.

20. Mendoza Sánchez, María A. Metodología de Desarrollo de Software. [Online] Junio 7, 2004. [Cited: 5 26, 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

21. Molpeceres, Alberto. Metodología de desarrollo: RUP, XP y FDD. [Online] v1.0, 12 15, 2003.

22. Díaz Flores, Miriam Milagro. "Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia...". [Online] url:http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

23. ISO 9126. *Calidad en la Industria del software.* 1992.

24. ISO 9000 3. Laboratorio de Sistema de Informática. [Online] [Cited: 5 20, 2010.] http://users.dsic.upv.es/asignaturas/facultad/lsi/trabajos/102000.doc.

25. López Quesada, Juan Antonio. *05BM- Fundamentos de Ingeniería del software. Tema 6. El proceso de sofware. Paradigma del ciclo de vida.* Informática y Sistema. 2007. p. 75, pdf.

26. ISO/IEC 12 207. *Técnología de la Información. Proceso del ciclo de vida del software.* Lima, Perú : 2 da Edición, 2006. p. 194, pdf.

27. Glossary of Testing. Standard glossary of term used in Software Testing. Version 1.1(dd. September, 29th 2005). [Online] Erick Van Veenendaal.

28. Knetia Software for business solutions. Metodología de Pruebas. [Online] 2007. http://sp.kynetia.com/calidad/metodologia-de-pruebas.html.

29. Deming, W Edwards. *Capitulo 2. Proceso Personal de Software.* p. 12, pdf.

30. Calvo Manzano, Bayona. *Team Software Process(TSP): Mejoras en la estimación, calidad y profundidad de los equipos en la Gestión del software.* Lenguaje y Sistema Informáticos e Ingeniería del software.Facultad de informática. Madrid, España : s.n., 2007.

31. IEEE Std 610. *Standard Glossary of Software Engineering Terminology.* 12-1990. p. 93, pdf.

32. Ian Sommerville. *Ingeniería de software.Séptima Edición.* Madrid: PEARSON EDUCACIÓN, S.A.,. : s.n., 2005. p. 712.

33. Pérez Lamandra, Beatriz. *Qestión de las Pruebas Funcionales.* Universidad de República Montevideo. Uruguay : s.n., 2007.

34. Std 1028. *IEEE Satandard for Software Review.* New York: Software Engenieering Standards Committee : s.n., 1997.

35. Pressman, R. *Ingeniería del Software.Un enfoque práctico.* 2002.

36. Grupo ARQUISOFT. [Online] Rojas, Johanna; Barrios, Emilio ;, 2007. [Cited: 6 1, 2010.]

http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

37. Letelier. *Sexta Parte. Prueba.Fase en que se Desarrolla.* p. 11, doc.

38. Davila Cabo de Villa, Evangelina. *Evaluación del contenido del Manual para Enfermeros Auxiliares de Anestesia.* Cienfuegos : s.n., 2007.

39. Cueva Lovelle, Juan Manuel. *Calidad del Software.* Informática, Universidad de Oviedo. España : s.n. p. 12, pdf.

40. ISO 9001. *Sistema de gestión de la calidad- Requisitos.* Ginebra, Suisa; : Impreso en la Secretaría Central de ISO, 2000. p. 34 pág.

41. Fillottrani, Pablo R. *Calidad en el desarrollo del software. Modelo de calidad del software.* Ciencia e Ingeniería de la Computación, Universidad Nacional del Sur. 2007.

42. Sierra, Miguel. Calidad en el desarrollo de software. [Online] 4 28, 2009. http://geeks.ms/blogs/msierra/archive/2009/04/28/Calidad_3A00_-C_F300_mo-mejorar-en-el-desarrollo-de-software-_2D00_-How-to-improve-in-software-development.aspx.

43. Ruiz Morilla, José Juaquín. ISO 9126 vs. SQuaRE. Calidad y Medición de Sistema Informáticos. [Online] 1 9, 2008. http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Joaquin%20Ruiz.pdf.

44. Letelier, Patricio and Panadés, Ma Carmen. Metodologías Ágiles para el desarrollo. Extreme Programming. [Online] http://www.willydev.net/descargas/masyxp.pdf.

45. Humphrey, Watt S. The Team Software Process. [Online] November 2000. http://www.lsi.us.es/docs/doctorado/memorias/MemoInvestigArturoHTorres.pdf .

46. Hassan Montero, Yusef. Introducción a la Usabilidad. [Online] 11 1, 2002. http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm.

47. Gil, Denis and Monteverde, Alejandro. Pruebas de Soportabilidad. [Online] http://carolina.terna.net/ingsw3/datos/Pruebas_de_Soporte.pdf.

48. Fernández Escribano, Gerardo. Introducción a Extreme Programming. Ingeniería de Software II. [Online] 12 9, 2002. http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

49. De Seta, Leonardo. Tendencia en el mundo del software, según Kent Beck. [Online] 2 5, 2009. http://www.dosideas.com/noticias/reflexiones/409-tendencias-en-el-mundo-del-software-segun-kent-beck.html.

50. Choucair Cárdenas, Maria Clara. Pruebas de software ¿la salvación, un proceso sin utilidad, trivial, simplemente una moda, o ....? XXVII Salón de Informática. [Online] Septiembre 2007. url:http://www.acis.org.co/fileadmin/Base_de_Conocimiento/XXVII_Salon_Informatica/MariaClaraChoucair-PruebasDeSoftware.pdf .

51. Carrasco, Luis de Salvador. Verificación y Validación. [Online] http://www.luisdesalvador.com/Oposicion/T029_032_VerificacionValidacion.pdf.

52. Blank, Isabel, Herrera, Larissa and Ortiz, Miguel. Pruebas de Funcionalidad. [Online] Mayo 2005. http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf.

53. Robertoyudice. Los 5 tipos de pruebas del software. [Online] Abril 8, 2007. http://robertoyudice.com/general/los-5-tipos-de-prueba-del-software/.

54. Verificación y Validación. [Online] [Cited: 4 20, 2010.] http://wwwdi.ujaen.es/asignaturas/computacionestadistica/pdfs/tema6.pdf.

55. Verificación y Validación. [Online] [Cited: 5 12, 2010.] http://www.eici.ucm.cl/Academicos/ygomez/descargas/calidad/cap22verificacionyvalidacion.ppt.

56. Prueba de Software. [Online] [Cited: 5 9, 2010.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

57. Luis Artola. Los tipos de Pruebas automatizados de software. [Online] 6 5, 2009. http://www.programania.net/diseno-de-software/tipos-de-pruebas-automatizadas-de-software/.

58. Tecnología de la Información y Servicios Telemáticos. CITMATEL. [Online] 2005. http://www.citmatel.cu/empresa.php.

59. Casos de prueba. Producto de desarrollo de software. [Online] [Cited: 4 22, 2010.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/prod_des/documento/casos_prueba_d.php.

60. *Calidad de software.* Ingeniería del Software I, Universidad Rey Juan Carlos. p. 61, ppt.

Glosario de términos

Actividades: es una unidad de trabajo que una persona que desempeñe un rol puede ser solicitado a que realice.

Artefacto: es una parte de una información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que van creando y usando hasta obtener el producto final (ej. Un modelo, un documento y un elemento de un modelo).

Aseguramiento de Calidad: "Es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisface los requisitos dados de calidad" (39).

Configuración: testeo de un sistema bajo diferentes configuraciones de hardware, disco duro, CPU. impresora, tarjeta de sonido, sistemas gráficos y/o revisiones de un sistema operativo y sus diferentes versiones de Base de Datos.

Defectos: un defecto se encuentra en un artefacto y puede definirse como una deficiencia entre la versión correcta del artefacto y una versión incorrecta. Coincide con la definición de diccionario, "imperfección".

Error: una acción humana que puede producir resultados incorrectos.

Especificación del software: "Se debe definir las funcionalidades y restricciones operacionales que debe cumplir el software" (15).

Fallo: "La incapacidad de un sistema o de algunos de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados" (8).

Módulos: es una abstracción del sistema, especificando el sistema desde un punto de vista y un determinado nivel de abstracción.

Software: conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Proceso: es una definición del conjunto completo de actividades necesarias para transformar los requisitos de usuario en un proyecto.

Producto: artefacto que se crea durante la vida del proyecto, como los modelos, código fuente, ejecutable y documentación.

Proyecto: elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto.

Verificación y Validación: Es una disciplina técnica de ingeniería de sistema. Proporciona una evaluación objetiva de los productos de software en todo su ciclo de Vida.

Anexos

Anexo 1 – Glosario de Término del Manual de Prueba para Ingeniero de software.

A

*Alfa: "*Acción simulada o real de pruebas por parte de los potenciales usuarios / clientes o un encargado de prueba en el equipo de desarrolladores, pero fuera de la organización de desarrollo" (15).

Anomalía: "Algo observó en la documentación o funcionamiento de software que se desvía de las expectativas basados en productos del software previamente verificados o documentos de la referencia" (11).

*Auditoría: "*Una evaluación independiente de productos de software o procesos para determinar el cumplimiento de las normas, las directrices, las especificaciones, y/o procedimiento sobre la base de criterios objetivos, incluidos los documentos que especifican; la forma o el contenido de los productos que se producen, la manera en que se medirán el cumplimiento de las normas o directrices" (34).

B

*Beta: "*Operativo de prueba por parte de los potenciales y/o actuales usuarios/clientes en el exterior sitio no involucrado con los desarrolladores, para determinar si procede o no un componente o el sistema cumple con el usuario/cliente necesita y encaja dentro de los procesos de negocio. La prueba Beta es a menudo empleada como una forma externa de la prueba de aceptación de off-the-sheft software con el fin de adquirir la información del mercado" (15).

C

*Calidad: "*Es el grado en que un conjunto de características inherentes cumple con unos requisitos" (40).

Calidad del software: “La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” (31) (35).

Caso de prueba: un conjunto de entradas de la prueba, de ejecución y de resultados desarrollados para un objetivo particular, como ejercer un camino del programa particular o verificar la complacencia con un requisito específico. Documentación que especifica las entradas, resultados, y un juego de condiciones de la ejecución para un artículo de la prueba.

Código Fuente: es un conjunto de líneas de texto que son implementadas en un lenguaje de programación, el que debe seguir una computadora para ejecutar dicho programa.

Código Ejecutable: archivo que contiene instrucciones que comprende el CPU para ejecutarlas, su extensión de tres letras en terminología PC es EXE.

Componente: uno de las partes que constituyen un sistema. Un componente puede ser hardware o software y puede subdividirse en otros componentes.

D

Defectos: un defecto se encuentra en un artefacto y puede definirse como una deficiencia entre la versión correcta del artefacto y una versión incorrecta. Coincide con la definición de diccionario, “imperfección”.

Descripción del software de diseño (SDD): “Una representación de software creado para facilitar el análisis, planificación, ejecución y toma de decisiones. La descripción del diseño de software se utiliza como un medio para comunicar información de diseño de software, y puede ser pensado como una guía o modelo del sistema” (11).

Documento de diseño de interfaz (DDI): “Documentación que describe la arquitectura y el diseño de interfaces entre el sistema y componentes. Estas descripciones incluyen algoritmos de control, los protocolos, los contenidos y formatos de datos y el rendimiento” (11).

E

Eficiencia: conjunto de características que determinan la relación entre el nivel de rendimiento del software y el número de recursos usados, bajo ciertas condiciones dadas. Se divide en las sub-características de comportamientos temporales, utilización de recursos.

Error: una acción humana que puede producir resultados incorrectos.

Prueba exitosa: una prueba que descubre defectos.

Especificación de requisitos de interfaz (IRS): "Documentación que especifica los requisitos para las interfaces entre los sistemas y componentes. Estos requisitos incluyen restricciones sobre los formatos y el momento" (11).

*Estabilidad: "*Sub-característica de mantenimiento, que indica volumen de riesgos de efectos inesperados tras una modificación" (23).

F

Fallo: "La incapacidad de un sistema o de algunos de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados" (8).

Fiabilidad: el grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.

*Flujos de Trabajo: "*Es una relación de actividades que nos producen unos resultados observables" (19).

G

Gestión de la calidad: ‘‘Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento (garantía) de la calidad y la mejora de la calidad y en el marco del sistema de calidad" (40).

I

*Inspección: "*Una inspección en que el líder prepara un "checklist" que sirve como guía de la reunión y contiene los puntos en que los revisores se tienen que fijar. El líder distribuye el checklist, el artefacto bajo testeo y otros materiales a los

participantes antes de la reunión. Los revisores tienen que estudiar el checklist y el artefacto bajo testeo antes de la reunión" (15).

Interfaz de prueba: es un tipo de prueba de integración que se refiere a las pruebas de las interfaces entre componentes o sistema.

M

Madurez: "Sub-característica de fiabilidad, que indica la frecuencia con que ocurren los fallos" (23).

*Mantenimiento: "*Esfuerzo requerido para implementar cambios. Se divide en las sub-características capacidad de ser analizado, cambiabilidad, estabilidad, facilidad de prueba" (23).

Migración: acción de pasar de un sistema a otro para establecer en él.

O

*Operatividad: "*Sub-característica de facilidad de uso, que indica las característica del software que influyen en el esfuerzo del usuario para un control operacional" (23).

P

Peligro: una propiedad intrínseca o condición que tienen el potencial para causar daño.

*Performance: "*El grado en que un sistema componente realiza sus designados. Dado funciones dentro de las limitaciones en cuanto tiempo de procesamiento y capacidad de tratamiento" (31).

Plan de prueba: es un documento que describe el alcance, acercamiento, recursos, y horario de actividades de la prueba intencionales. Identifica los artículos de la prueba, los rasgos a ser probados, la comprobación de las tareas que se harán y cualquier riesgo que requiere la planificación de contingencia.

Precisión:" Sub-característica de funcionalidad, que indica el grado de exactitud de los defectos del sistema" (23).

Prueba: acción y efecto de probar. Operación que se ejecuta para comprobar que otra ya hecha es correcta.

Prueba de Aceptación: la comprobación formal dirigida para determinar si un sistema satisface su criterio de aceptación y para permitirle al cliente que determine si satisface sus necesidades.

Prueba de Integración: se prueba la parte del software, la del hardware por independiente y luego la combinación de los dos para evaluar la interacción entre ellos.

Prueba de Sistema: prueba dirigieron en un sistema completo, integrado para evaluar la complacencia del sistema con sus requisitos especificados.

R

*Requerimientos Funcionales: "*Capacidades o condiciones que el sistema debe cumplir*"* (19).

*Requerimientos no Funcionales: "*Son propiedades o cualidades que el producto debe tener, es decir lo que lo hace atractivo, usable, rápido y confiable. Son una parte significativa de la especificación" (19).

*Revisión: "*Es la noción de corrección de errores, esto es, hacer cambios a un programa que corrigen solo errores en diseño lógico pero no afectan las capacidades funcionales documentadas, dado que ningún requerimiento ha cambiado" (19).

Rol: papel que desempeña una persona en un determinado momento.

S

Software: conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Stakeholder: son individuos y organizaciones involucradas en el proyecto o cuyos intereses pueden ser afectados positivas o negativamente como resultado de la ejecución o conclusión del proyecto, no son más que los interesados en el proyecto.

U

Usabilidad: “Consiste de un conjunto de atributos que permiten evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema” (23) (41).

*Utilización de recursos: “*Sub-característica de eficiencia, que indica las características del software que influyen en el número de recursos usados, u la duración de su uso, cuando se lleva a cabo su función” (23).

V

Validación: “El proceso de evaluar un sistema o componente durante o al final del proceso de desarrollo para determinar si satisface los requisitos especificados” (9).

Verificación: “El proceso de evaluar un sistema o componente para determinar si los productos de una fase de desarrollo dada, satisfacen las condiciones impuestas a la salida de esa fase” (9).

W

Walkthrough: participa en los procesos de la evaluación en que la primacía de personal de desarrollo otros a través de un examen estructurado de un producto. Asegure que los participantes se califican para examinar los productos y no están sujetos a la influencia indebida. El líder se llama coordinador y es el autor, presentador, secretario. No hay checklist como en las inspecciones.

Anexo 2 – Encuesta

Usted ha sido seleccionado como experto para ofrecer sus criterios valorativos acerca del contenido mostrado en la Estrategia del Manual de Prueba para Ingeniero de software elaborados en la Facultad de Informática de la UCF. De antemano le agradecemos por su cooperación.

Instrucciones: Para llenar éste cuestionario de evaluación es importante que siga los siguientes pasos:

Evalúe los criterios que se resaltan utilizando para ello las variables que se adjuntan a cada uno.

Marque con una (X) en la escala de evaluación que se adjunta a cada variable utilizando la siguiente escala:

1___ Total desacuerdo 2 ___ 3___ 4___ 5___ Total acuerdo.

Cuando el experto no tiene elementos suficientes para emitir un criterio de valor sobre el ítem debe marcar en el valor 3 que indica que No aplica.

Cuando lo considere pertinente escriba sus criterios en la celda correspondiente a las Observaciones.

Lenguaje y redacción.
REDACCIÓN 1___ 2___ 3___ 4___ 5___
a) La expresión de las ideas es clara y precisa.
b) Las estructuras gramaticales se utilizan correctamente.
Observaciones:
LENGUAJE ADECUADO 1___ 2___ 3___ 4___ 5___
a) El lenguaje utilizado requiere de un conocimiento elemental sobre las pruebas.
b) El lenguaje es fácilmente comprensible.
Observaciones:
Contenido.
CAPACIDAD DE DESCRIPCIÓN, EXPLICACIÓN Y PREDICCIÓN 1___ 2___ 3___ 4___ 5___
a) Se observa coherencia de los objetivos con el contenido.
b) Favorece el desarrollo de habilidades en el desempeño.
c) Ética adecuada.
d) Evidente utilidad práctica.
e) La profundidad y generalidad del contenido se adecua a los Ingenieros de pruebas.

f) Permite elevar los conocimientos de los usuarios e Ingenieros de prueba sobre el proceso de prueba.
Observaciones:
CONSISTENCIA LOGICA 1___ 2___ 3___ 4___ 5___
a) Adecuada interrelación y coherencia entre los aspectos tratados.
b) Las actividades están estructuradas con inicio, desarrollo y cierre.
c) Correcta relación entre la teoría y la práctica.
d) Las ilustraciones son oportunas y se corresponden con el contenido.
Observaciones:
Aspectos pedagógicos.
MOTIVACIÓN 1___ 2___ 3___ 4___ 5___
a) Logra motivar por su originalidad.
b) La interactividad es apropiada para el usuario.
c) El contenido estimula su utilización.
Observaciones:
CONFIABILIDAD PSICOPEDAGÓGICA 1___ 2___ 3___ 4___ 5___
a) La estrategia es eficaz instructivamente.
b) La estrategia se adapta a las exigencias comunicativas del usuario.
Observaciones:
Perspectiva
EXPECTATIVAS 1___ 2___ 3___ 4___ 5___
a) La estrategia satisface las expectativas del usuario.
b) Constituye una fuente de consulta necesaria para el Ingeniero de Prueba.
Observaciones:

Anexo 3 – Análisis Estadístico

Fiabilidad

Resumen del procesamiento de los casos

		N	%
Casos	Válidos	12	100.0
	Excluidos(a)	0	.0
	Total	12	100.0

a Eliminación por lista basada en todas las variables del procedimiento.

Estadísticos de fiabilidad

Alfa de Cronbach	N de elementos
.739	7

Resultados Descriptivos

Estadísticos

		Redacción	Lenguaje adecuado	Capacidad de descripción, explicación y predicción	Consistencia Lógica	Motivación	Confiabilidad Psicopedagógica	Expectativas
N	**Válidos**	**12**	**12**	**12**	**12**	**12**	**12**	**12**
	Perdidos	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Mediana		**4.50**	**5.00**	**5.00**	**5.00**	**4.00**	**5.00**	**5.00**
Mínimo		**4**	**3**	**3**	**4**	**3**	**4**	**4**
Máximo		**5**	**5**	**5**	**5**	**5**	**5**	**5**

Frecuencia

Redacción

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	En acuerdo	6	50.0	50.0	50.0
	Total acuerdo	6	50.0	50.0	100.0
	Total	12	100.0	100.0	

Lenguaje adecuado

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	NA	1	8.3	8.3	8.3
	En acuerdo	4	33.3	33.3	41.7
	Total acuerdo	7	58.3	58.3	100.0
	Total	12	100.0	100.0	

Capacidad de descripción, explicación y predicción

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	NA	1	8.3	8.3	8.3
	En acuerdo	4	33.3	33.3	41.7
	Total acuerdo	7	58.3	58.3	100.0
	Total	12	100.0	100.0	

Consistencia Lógica

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	En acuerdo	3	25.0	25.0	25.0
	Total acuerdo	9	75.0	75.0	100.0
	Total	12	100.0	100.0	

Motivación

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	NA	1	8.3	8.3	8.3
	En acuerdo	10	83.3	83.3	91.7
	Total acuerdo	1	8.3	8.3	100.0
	Total	12	100.0	100.0	

Confiabilidad Psicopedagógica

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	En acuerdo	5	41.7	41.7	41.7
	Total acuerdo	7	58.3	58.3	100.0
	Total	12	100.0	100.0	

Expectativas

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	En acuerdo	5	41.7	41.7	41.7
	Total acuerdo	7	58.3	58.3	100.0
	Total	12	100.0	100.0	

Prueba W de Kendall

Rangos

	Rango promedio
Redacción	4.04
Lenguaje adecuado	4.08
Capacidad de descripción, explicación y predicción	4.08
Consistencia Lógica	4.92
Motivación	2.33
Confiabilidad Psicopedagógica	4.21
Expectativas	4.33

Estadísticos de contraste

N	12
W de Kendall(a)	.224
Chi-cuadrado	16.163
gl	6
Sig. asintót.	.013

a Coeficiente de concordancia de Kendall