Universidad de las Ciencias Informáticas
Facultad 3.

Título: *"Propuesta del Manual del Ingeniero de Prueba para el proceso de prueba de la Universidad de las Ciencias Informáticas."*

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

Autor(es):

Yuliet Moreno Argüelles Anabel Martín Aparicio.

Tutor(es):

Msc.: Michael González Jorrín. Ing.: Olia Rodríguez Villanueva.

Junio 2008

"Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes."
Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2008.

Anabel Martín Aparicio

_____
Firma del Autor

Yuliet Moreno Arguelles

_____
Firma del Autor

Michael González Jorrín

_____
Firma del Tutor

Olia Rodriguez Villanueva

_____
Firma del Tutor

AGRADECIMIENTOS

A mis padres, por tanto amor y confianza depositados en mí, por enseñarme a luchar por mis sueños, tenerme en sus oraciones todos los días y brindarme su apoyo en todo momento de mi vida y más aún durante la realización de esta investigación, ¡Los Quiero Mucho! que Dios le de mucha salud.

A mi esposo, por su amor eterno, apoyo, comprensión y que disfruta cada día mis logros personales. Mil gracias por existir.

A familia y en especial a mis abuelos queridos, Ana, Niní, Milda y Moreno por haber depositado en mí toda su confianza, por su interés y preocupación.

A mi tutor Michael, por tener tanta paciencia y tratar de hacer de nosotras excelentes profesionales.

A mi tutora Olía, por todo el empeño que puso en ayudarnos y el gran apoyo en esta etapa tan importante y difícil.

A mis amigas de toda la vida Mailis, Maylen y Dainerys, quienes en estos años me comprendieron y apoyaron en todo momento, nunca las voy a olvidar.

A mis amigos Coppen, Omel, Iosmel, Alain, Wilder, Heney, por todo este tiempo compartido.

A María y Arnoldo por preocuparse siempre por mí y brindarme mucho apoyo espiritual, les deseo mucha suerte en la vida.

A Ida por brindarme su cariño y cuidarme como si fuera su propia hija. Muchas gracias y que el señor te bendiga.

A los que leyeron mi tesis porque con sus comentarios y observaciones me permitieron enriquecerla.

A mis amigas de siempre Roxana, Dailen, Dailin y Yadira, por hacer perdurar esta amistad a pesar de las dificultades y distancia.

A todos mis compañeros de la UCI, por su apoyo cuando lo necesité y sus críticas constructivas que me ayudaron a ser mejor cada día.

A todos los profesores que durante todos estos años contribuyeron con mi formación profesional.

Especialmente a aquellas personas que de un modo u otro han hecho posible que este proyecto sea una linda verdad.

Mil Gracias…Yuliet.

AGRADECIMIENTOS

A mis padres por apoyarme siempre, por la dedicación y atención que han tenido conmigo durante toda mi vida y mis estudios.

A mi papá Rafael Martín por ser una persona especial en mi vida, quererme tanto y dedicarse a mí de esa forma tan exclusiva.

A mi mamá Enma Aparicio por quererme, consentirme, ayudarme, malcriarme y ser tan buena y noble.

A mi abuela Teresa por siempre mimarme tanto, protegerme y estar a mi lado.

A mi abuelo Felipe y a mi abuela Felicia por ayudarme mucho y soportarme todo.

A Jose por ayudarme siempre y ser como otro papá.

A Roberto por apoyarme y ayudar mi mamá.

A Gui por quererme, mimarme, ser especial, amarme y estar conmigo.

A mi primo Felipe por ser tan bueno, quererme tanto y ser mi amigo además.

A Sheila por ser una prima especial.

A mi prima Mary y a Renata por dejarme ser una tía aunque no tenga hermanos.

A Glafira e Ingrid por ayudarme mucho y ser buenas.

A mis tíos/as Barbará, Martica, Milla, Mary, María E, Carmen, Dulce, José, Felipe, Jorge, Migue, Naranjo, René por ser siempre buenos.

A mis primos Juana, Adis, Kiki, Eduardo, Marianne y Rodolfito por siempre estar conmigo.

A Iosmel, Carlos, y Yonelbys por demostrarme q todavía quedan mejores amigos en el mundo que nunca vas a olvidar.

A Liset por ser como mi hermana.

A mis amigos por darme todo el amor del mundo, por permitirme compartir con ellos todos estos años de estudios y ayudarme siempre.

A mi grupo 3503 por brindarme un espacio tan lindo donde estar. A mi antiguo grupo 3112 por seguir siendo muy buenos compañeros.

A Pascual por ayudarme tanto en estos 5 años y en los momentos difíciles.

A Fabio, Yenner, Jorge, Sol, Gretty, Hugo, Livan, Yoandy, Maikel, Reynaldo, Tamara, Yanisbeli, Emilio, Dianet por ayudarme tanto en mis últimos días como estudiante y demostrarme que siempre se pueden hacer amigos nuevos y que se conocen personas buenas que recordarás toda la vida.

A Yllen por ser una vieja amiga, a Lidy por ser tan amable y ayudarme siempre, a Jasmín por estar siempre conmigo, a Susana por estar ahí, a Dania por ser tan buena, a Maylen por tener siempre

AGRADECIMIENTOS.

alegría, a Yuli por estar ahí y llevarnos bien, a Reynold por ser tranquilo, a Maikel por ayudarme siempre. A todos por ser mis amigos estos 5 años.

A mis tutores por su dedicación y paciencia, en especial a Olia, por ayudarme tanto.

A mis amigos del pre Yudis, Yane, Yusi, Yaniu, Dianka, Yeni, Yaime, Adan, Yunet, Reinier, Willy, por seguir siendo los mismos siempre.

A todas las personas que conocí en la UCI, por ayudarme y apoyarme.

A todos los que conozco porque aunque sea una vez en la vida estuvieron allí para mí.

Gracias por ser especiales y estar ahí… Ana.

DEDICATORIA.

A mi mamá y mi papá porque de no ser por su amor, dedicación y sacrificio no hubiera sido posible hoy estar aquí, porque en cada hoja de este trabajo hay reflejada una palabra de aliento suya y me ayudaron a conseguir mi sueño y a pesar de estar lejos de mi, me cuidaron y mimaron como lo más preciado de su vida. Gracias a Dios que me dio uno padres como ustedes. ¡Los quiero mucho!

A mi esposo por hacer una diferencia en todos mis días, porque estoy hecha de su carne, mis sueños son sus sueños, siempre por siempre te amaré.

A mi familia y en especial mis abuelitos del alma, que los quiero mucho y confiaron en mi todo el tiempo.

A todas las personas que me brindaron una sonrisa o un gesto de cariño, y que contribuyeron con mi educación. Gracias por existir y apoyarme en todo momento.

Yuliet.

A mis padres por ser especiales, a ti mami por ayudarme en todos los momentos que lo necesite y quererme mucho y a ti papito por ser tan único conmigo y quererme de esa forma particular. A ambos por el apoyo moral y espiritual que me han dado y porque los quiero mucho con el corazón y no hay nada mas grande en el mundo que mi amor por ustedes.

A mi novio Gui por ser tan especial, bueno, lindo, y dejarme que lo quiera tanto, además de ser mi amigo.

A mi familia por haber estado conmigo toda mi vida. A todas las personas que quiero por ser mi apoyo y darme es luz que siempre se necesita.

Gracias por siempre estar allí para mí.

Los quiero mucho… Ana.

RESUMEN

Con la intención de definir un proceso de prueba organizado, para el desarrollo del software en la Universidad de Ciencias Informáticas (UCI), la presente investigación centra como su objeto de estudio el proceso de pruebas de software, y dentro de este más específicamente el proceso personal de prueba. Se propone un manual para el ingeniero de prueba, con el objetivo de establecer una guía para la aplicación de pruebas en cada uno de los proyectos de la UCI y así lograr la calidad requerida en los productos y con ello la satisfacción de los clientes. Seleccionando parte del estándar de la IEEE 1012 (IEEE 1012-1998) Proceso de Verificación y Validación, como estrategia a ser aplicada de manera paralela al ciclo de vida del software, definiendo para dicho ciclo, las tareas y tipos de pruebas a realizar en cada una de sus actividades. Además propone un modelo de pruebas, como vía para aplicar la estrategia propuesta para ganar en organización, madurez, habilidad y control de las pruebas en los Proyectos.

PALABRAS CLAVE.

Pruebas, verificación, validación, proceso de prueba, desarrollo del software, manual del ingeniero, ciclo de vida del software.

ÍNDICE.

INTRODUCCIÓN.

Hoy en día, se buscan software no solo que sean idóneos para realizar determinadas funciones, sino que tengan además un nivel de calidad, debido a que el interés por la misma crece de forma continua, volviéndose los clientes más selectivos y capaces de rechazar productos poco fiables o que realmente no dan respuesta a sus necesidades.

La calidad del software "es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente" (Pressman, 2002.).

Convirtiéndose así la calidad en una disciplina más dentro de la Ingeniería del Software y uno de los elementos principales para medirla en un producto son las pruebas. Actualmente para lograr la inserción de un software en el mercado debemos contar con un buen desarrollo de pruebas realizadas al mismo, con el objetivo de detectar errores en la aplicación y en la documentación; este proceso da una medida de la calidad del producto, siempre que se lleve a cabo de forma apropiada.

A nivel mundial la industria del software cuenta con sólo algunas décadas y está aún definiendo sus caminos, buscando la manera adecuada de desarrollarse. Tiempo atrás las pruebas de software se consideraban sólo una actividad que realizaba el programador para encontrar fallas en sus productos; pero con el paso de los años se han convertido en un proceso elemental cuyo propósito principal es evaluar la funcionalidad del software respecto a los requerimientos establecidos, utilizando un conjunto de herramientas, técnicas y métodos que hacen a la excelencia, el desempeño de un programa y ayudan a aliviar el peso y el costo del tiempo del desarrollo del producto.

Surgiendo entonces una nueva tendencia a nivel mundial a la hora de aplicar las pruebas, centrada principalmente a lo largo del ciclo de desarrollo de vida del software, tratando de evitar que se introduzca la menor cantidad de errores posibles, capacitando para ello especialistas responsables de esta actividad.

Muchas empresas en el mundo han definido sus propias metodologías y herramientas que permiten elevar el nivel de calidad del software y llevar a cabo un proceso de pruebas de forma más efectiva con la detección y corrección temprana de los errores.

Una de las vías encontradas para lograr que el software producido cuente con un alto nivel de calidad son los Modelos de Calidad, algunos han sido elaborados a partir de otros existentes con el objetivo

de lograr un modelo más completo. El mundo se inclina hacia las propuestas de la ISO[1] y del SEI[2] mayoritariamente.

Se han creado una gran cantidad de metodologías, modelos y estándares por empresas como PROFit[3] la cual ha desarrollado un modelo de verificación y validación (V&V) basado en el estándar IEEE Std. 1012-1998 y que además cumple con los requisitos especificados en el modelo CMMI[4], uno de los más extendidos en las áreas de desarrollo. Este modelo tiene su propio ciclo de vida, que se ejecuta en paralelo con el ciclo de vida clásico del desarrollo. Así la verificación y validación (V&V) abarca a todas las fases de éste.

Otra de las metodologías creadas para la mejora del proceso de pruebas en el desarrollo del software es la BaQEM[5], definida por GreenSQA[6] , con el objetivo de utilizar un mecanismo que sea efectivo para la evaluación de la calidad de los productos de software desarrollados en ParqueSoft[7]. Esta metodología también se emplea en todo el ciclo de vida del producto. GreenSQA ofrece servicios de Pruebas de Software y Consultoría de clase mundial a la industria latinoamericana de software, servicios que contribuyen al aumento de la productividad y competitividad de las empresas mediante la definición y optimización de sus procesos de desarrollo de software.

Existen compañías especializadas donde el Control de la Calidad y las Pruebas de Software son brindados como servicio. La compañía SQS S.A, se especializa en servicios de Consultoría de Calidad de Software y Pruebas. Para la promoción de este tipo de actividades se organizan encuentros periódicos con profesionales como es el caso de la celebración anual de las Conferencias Internacionales QA&TEST. Este evento se ha convertido en una cita ineludible al que acuden empresas de todo el mundo para exponer y conocer experiencias relacionadas con la Calidad del

[1] International Standards Organization: Su principal función es buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

[2] Software Engeniering Institute: Instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software.

[3] Programa de Fomento de la Investigación Técnica: Es un instrumento mediante el cual el Gobierno articula un conjunto de convocatorias de ayudas públicas, destinadas a estimular a las empresas y a otras entidades a llevar a cabo actividades de investigación y desarrollo tecnológico.

[4] Capability Maturity Model Integration (Modelo de Integración de la Capacidad de Madurez del proceso de desarrollo de software). Es un modelo para la mejora de procesos que proporciona a las organizaciones los elementos esenciales para procesos eficaces.

[5] Business Application Quality Evaluation Method: Con el propósito de aportar una metodología cuantitativa, práctica y efectiva para evaluar y analizar la calidad de los sistemas de información desarrollados en ParqueSoft, independiente de la metodología y plataforma tecnológica utilizada para su desarrollo.

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

[7] Parque Tecnológico de Software: Clúster de empresas de base tecnológica especializadas en la industria del conocimiento, a través del desarrollo de productos, soluciones y servicios de software.

Software y las Pruebas. Otra entidad vinculada a las pruebas de software es TestAbil, que se dedica a la especialización en control de calidad del mismo.

El Instituto Tecnológico de Informática organiza eventos todos los años y en este año se realiza la quinta edición de las Jornadas Internacionales sobre Testeo de Software, JTS2008. El propósito de las mismas es divulgar entre las empresas del sector TIC[8] la importancia del testeo de software. En estas ediciones, se cuenta con la participación de expertos nacionales e internacionales en software testing, quienes exponen las tendencias actuales más relevantes en ésta área y enseñan prácticas y conocimientos que directamente pueden aplicarse en el trabajo diario. Con estas jornadas, el ITI[9] apuesta por contenidos que ayuden al asistente a implantar y mejorar técnicas.

Otro evento importante es Expo: QA, un espacio privilegiado en el sector de la Calidad del Software en España, que facilita intercambios entre los profesionales que buscan soluciones y los proveedores líderes del mercado que las proporcionan. SOGETI, empresa líder en software control & testing es la organizadora de este gran evento.

La otra tendencia respecto a las pruebas de software, es la automatización de sus procesos, para lo cual se han creado una serie de herramientas como JUnit (para aplicaciones en Java) y NUnit (para aplicaciones en .NET), que ayudan al diseño y ejecución de las pruebas de caja blanca; el JMeter y el Empirix sirven para la realización de pruebas funcionales y de carga y el VPerformer Software, que permite evaluar aplicaciones Web sometiéndolas a pruebas de carga y rendimiento. Se considera que estas herramientas ayudan en gran medida a la realización de las pruebas de software ya que en nuestros días los sistemas que se necesitan son cada vez más grandes y éstos llegan al límite de la capacidad humana, además refuerza el desarrollo de las pruebas ya que permite que se ejecuten al mismo tiempo que el software que se desea desarrollar disminuyendo tiempo, esfuerzo y costo.

Cuba también ha visto la importancia de que sus productos tengan calidad por lo cual se esfuerza cada día más para obtenerla, y así poder introducirse en el mercado internacional, llegando a destacarse varias empresas dedicadas a la producción de software, ejemplo de ellas: SOFTEL[10], DESOFT[11], SEGURMATICA[12] .

[8] Tecnología de la Informática y Comunicación: Son herramientas teórico-conceptuales, soportes y canales que procesan, almacenan, sintetizan, recuperan y presentan información de la forma más variada.

[9] Instituto Tecnológico de Informática: Es un centro tecnológico especializado en investigación, desarrollo e innovación en tecnologías software.

[10] Soluciones Informáticas: Empresa que ofrece soluciones informáticas para el Sistema de Salud en Cuba.

[11] Empresa Nacional de Software: Empresa dedicada al desarrollo de software en cuba, así como la informatización de la sociedad cubana.

[12] Consultoría y Seguridad Informática: Empresa cubana de consultoría y seguridad informática.

INTRODUCCIÓN.

En este momento está en pleno desarrollo de la industria de software y como parte importante de ese desarrollo se encuentra la Universidad de las Ciencias Informáticas (UCI), primera universidad nacida en el fragor de la batalla de ideas, la cuál fue creada en el curso 2002-2003 con dos misiones fundamentales: formar profesionales comprometidos con su Patria, y altamente calificados en la rama de la informática. Y producir software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación predominante (RUZ, 2002.)

Como parte de los esfuerzos por mejorar la calidad de los productos desarrollados en Cuba se creó CALISOFT, el cual se encarga, por el momento, de la realización de Controles de Calidad y llevar a cabo las pruebas de software. Este organismo ha participado en las Pruebas de Aceptación a importantes productos desarrollados en la UCI, sirviendo de tercero entre el Proveedor y el Cliente, viabilizando, organizando y desarrollando el proceso de pruebas, para lo cual define, prepara y presenta todo el aseguramiento tecnológico necesario referente a este proceso.

Dicho esto, surge la necesidad inmediata de que los software producidos por la Universidad de Ciencias Informáticas cuenten con un alto nivel de calidad para su uso posterior.

En este sentido, la universidad se dio la tarea de controlar la calidad del producto de software. Emergiendo así el Grupo de Investigación de Calidad e Ingeniería de Software de la UCI, la Dirección de Calidad de la Infraestructura Productiva (IP) y el Laboratorio de Certificación, el cual permite la realización de pruebas, pero todavía no está avalado para dar certificación.

El grupo de investigación se dedica a indagar sobre temas referentes a los procesos de desarrollo de software, metodologías y calidad. A partir de los resultados de la investigación que se realizan en el grupo, la Dirección de Calidad de la IP dicta las políticas por la que se deben regir los proyectos que tienen como eje fundamental las distintas facultades. Al mismo tiempo, la Dirección de Calidad se encarga de dirigir y llevar a cabo la planificación y ejecución del control de la calidad a que deben ser sometidos todos los proyectos de desarrollo de software y aplicar los principios de las revisiones e inspecciones con vistas a la mejora continua de los productos a obtener, ya sea de la aplicación o de toda la documentación asociada al producto a través del trabajo del Laboratorio de Certificación.

Aún con la existencia de este grupo se necesita madurar en el proceso de desarrollo de software, según la investigación realizada por (Michael G.J., 2007.) en su tesis de maestría, se han detectado problemas con el proceso de prueba en los proyectos debido a que en algunos casos se realizan las pruebas de forma empírica sin tener en cuenta las consecuencias que esto pueda traer consigo, por lo que no existe confiabilidad en cuanto a los resultados a los que se arriban después de la aplicación de las pruebas porque los recursos humanos no se sienten seguros, ni cuentan con un procedimiento que

los guie por todo el proceso de pruebas, dando una probabilidad bastante alta de realizar pruebas mal enfocadas. Para realizar este proceso es necesario tener primeramente conocimientos previos de las pruebas a realizar sobre dichos productos, de los cuales el personal responsable carece ya que existe poca experiencia laboral y preparación en el tema. Se suma además la no existencia de un compendio de plantillas de casos de pruebas estándar para todos los proyectos y que los procedimientos de pruebas no son aplicados en cada facultad de la misma forma.

La situación anteriormente descrita provoca que los productos no sean entregados en tiempo, que no cumplan con los debidos requerimientos tanto funcionales como no funcionales, especificados por los clientes y que la documentación que puede ser útil en un posterior mantenimiento y futuros proyectos no sea confiable.

Todo esto demuestra la ausencia de elementos en el proceso de pruebas definido; provocando que en la entrega de los productos no se evidencie una buena calidad en la ejecución de las pruebas, es por ello que surge la necesidad de la existencia de un manual para el ingeniero de pruebas ajustado a la realidad de la Universidad de Ciencias Informáticas, que permita detectar errores y enfocar los objetivos de las pruebas, trascendiendo positivamente en la calidad de los productos.

De esta manera se define el siguiente problema:

¿Cómo plasmar, el comportamiento deseado de los ingenieros de pruebas con el fin de facilitar el proceso personal de pruebas de software llevadas a cabo en la Universidad de Ciencias Informáticas?

El objeto de estudio que se plantea es: El proceso de pruebas de software.

El campo de acción es: El proceso personal de pruebas de software en la Universidad de Ciencias Informáticas.

Y la siguiente hipótesis: Si se propone un manual para plasmar el comportamiento de los ingenieros de pruebas entonces mejorará el proceso personal de pruebas de software llevadas a cavo en la UCI, elevando la calidad de los artefactos construidos.

Quedando definido así el siguiente objetivo General:

Proponer un manual para plasmar el comportamiento deseado de los ingenieros de pruebas, con el fin de facilitar el proceso personal de pruebas de software llevadas a cabo en la Universidad de Ciencias Informáticas.

Dentro de las *tareas de investigación* que se proponen desarrollar para el cumplimiento del objetivo se señalan:

- Realización de una investigación sobre las normas y estándares internacionales de calidad de software y su adaptación a las realidades concretas de la UCI.

- Realización de una investigación sobre el rol y responsabilidad del ingeniero de pruebas a lo largo del ciclo de vida de un software.
- Realización de un análisis de cómo se controla el proceso de pruebas hasta la fecha en la Universidad.
- Realización de una investigación sobre las herramientas de soporte para el proceso de pruebas y como se ajustarían a la realidad de la UCI.
- Desarrollo de un Manual del Ingeniero de Prueba, para concebir las actividades a realizar por el mismo en esta etapa, para facilitar el desarrollo del proceso de pruebas de la UCI.

Al concluir esta investigación se obtendrá, el desarrollo de un manual para los ingenieros de prueba. Además de portes significativos para las clases de ingeniería y gestión de software en los temas relacionados con las pruebas, así como a las asignaturas del segundo perfil de calidad, rol específico dentro del proceso de pruebas a desarrollarse por un estudiante en dependencia de su formación curricular. Facilitando también el proceso personal de pruebas de software llevadas a cabo en la Universidad de Ciencias Informáticas, elevando la calidad de los artefactos construidos.

Para llevar a cabo esta investigación se definió la utilización de métodos teóricos, los cuales posibilitaron el desarrollo del contenido esencial de la misma, permitiendo realizar un estudio del estado del arte de la ingeniería de prueba, su evolución y relación con otros elementos del proceso de desarrollo del software. Se utilizó en este trabajo el método de análisis histórico lógico que permitió conocer el desarrollo y evolución de las pruebas a lo largo de toda su historia. El análisis de toda la documentación encontrada se realizó a través del método analítico-sintético, pudiendo con el mismo sintetizar toda la información para obtener los aspectos más importantes de todos los documentos que se analizaron. La modelación permitió la definición de un modelo o estrategia para investigar la realidad de la ingeniería de prueba en el mundo actual y descubrir nuevas cualidades de la misma.

El uso de algunos métodos empíricos ofreció la posibilidad de verificar y comprobar varias concepciones teóricas agilizando la obtención de las principales características del objeto de estudio definido en la investigación. El método más apropiado para la investigación que se realizó fue el cuestionario, pues por mediación del mismo se pudo obtener información verídica del impacto de la propuesta, lográndose el objetivo de la misma.

El trabajo se estructuró en tres capítulos. En el primer capítulo, “Fundamentación Teórica”, se precisa el concepto de prueba de software partiendo de sus principios y características, así como la clasificación de las mismas en cuanto método, tipo y niveles. Se brinda una descripción general de las pruebas dentro del desarrollo del ciclo de vida del software, las pruebas como un concepto mas amplio

que es validación y verificación y como se llevan a cabo dentro de una organización. Además se identifican herramientas de soporte al proceso de pruebas. El segundo, se titula “Manual del Ingeniero de Prueba”, en el cual se describe las características de la solución que se propone, detallando su estructura y principales componentes, basado en el proceso de verificación y validación propuesto en IEEE 1012 -1998 Estándar para Verificación y Validación de Software. Además de analizar las actividades que lo sustentan, los roles vinculado al proceso de pruebas. Finalmente el capítulo 3, “Validación del Manual del Ingeniero de Prueba”, muestra la validación del manual propuesto a través del Método Delphi.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA.

1. Introducción.

Una necesidad imperiosa y creciente en la actualidad, a la hora de crear un software con calidad, son las pruebas que se le realizan al mismo, para garantizar una calidad adecuada en dicho producto. Es por esto que se hace necesario conocer los conceptos, principios y características de las mismas, así como su clasificación en métodos, niveles y tipos de pruebas. Se analiza además como son tratadas las pruebas dentro del desarrollo del software. Siendo necesario realizar un análisis del proceso de verificación y validación que se debe llevar a cabo de forma paralela a dicho desarrollo. Realizando un estudio de los modelos V, W y algunas herramientas que sirven de soporte a las pruebas, como el Rational TestManager, Rational ClearQuest, Rational Robot, Microsof Project, Motor de Base de Datos.

1.1 Las Pruebas de software.

Con el fin de elevar la calidad en el desarrollo de ciertos productos, se realizan un conjunto de actividades, jugando las pruebas un papel fundamental; razón por la cual están enmarcadas dentro del proceso de desarrollo del software, con la intención de garantizar la calidad del mismo y satisfacer los requerimientos especificados. Mediante las pruebas es posible localizar la mayor cantidad de deficiencias de un producto, permitiendo que se puedan subsanar en el menor tiempo posible. Existen normas internacionales que se encargan de definirlas como la (IEEE 1012-1998), la (IEEE 829-1998) de la IEEE[13] y por algunos autores como (Pressman, 2005.)

Según (Pressman, 2005) las pruebas son: El proceso de ejecución de un programa con la intención de descubrir un error.

La (IEEE 1012-1986), define las pruebas como una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

Las pruebas (Glossary of Testing Terms, 2005) son sets de uno o más casos de pruebas.

[13] Instituto de Ingenieros Eléctricos y Electrónicos: Sociedad Profesional con membrecía en todo el mundo. Se empeña en actividades técnicas y educacionales y profesionales que promueven la teoría y la práctica de la electro tecnología para el desarrollo personal y profesional.

Un proceso de pruebas son conceptos de pruebas, estrategia, técnicas y medidas que deben integrarse en un determinado proceso controlado y dirigido por personas. Apoyando las actividades de prueba y proporcionando una guía para los equipos de pruebas, desde la planificación de prueba, para probar la salida de evaluación, de tal forma que proporcione garantía de que los objetivos de las pruebas se cumplan de manera rentable. (Guide to the Software Engineering Body of Knowledge, 2004)

Tomando como base los conceptos anteriormente citados se puede llegar a la conclusión de que las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, donde se valida la calidad y funcionalidad del producto, mediante el análisis y evaluación de los resultados de dicha ejecución; con el fin de detectar errores, que impidan que dicho producto responda con las necesidades especificadas previamente. Y un proceso de prueba es el establecimiento de los objetivos de las pruebas, el diseño de los casos de pruebas a aplicar, la codificación de los casos de pruebas, la ejecución de los mismos y el análisis de los resultados de dicha ejecución.

1.1.1 ¿Que evalúa la 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.

En el modelo de calidad de McCall, se describe la calidad como un concepto elaborado mediante relaciones jerárquicas entre factores de calidad, en base a criterios y métricas. El término factor de calidad define características claves que un producto debe exhibir. En el que se identifican once factores de calidad orientados al producto y agrupados en un entorno de tres grupos: operación, revisión y reutilización (Anexo 2: Modelo de McCall). El gran aporte de este modelo fue el establecimiento de relaciones entre características de calidad y métricas, aunque haya sido criticado por la objetividad de algunas de las métricas propuestas.

El modelo de Boehm (MONTERO, 2005) (SCALONE, 2006.) (Anexo 3: Modelo de Boehm) es similar al de McCall, pero se destaca por ser uno de los mejores definidos. Es de naturaleza jerárquica y los criterios de calidad se presentan en tres grandes divisiones. La primera división es hecha acorde a los servicios que el sistema ofrece (Portabilidad). La segunda se hace de acuerdo a la operación del

producto (Usabilidad) y la tercera división se hace de acuerdo a la mantenibilidad del producto de software. Además, en este modelo se consideran tanto las necesidades del usuario como características de calidad en el terreno del hardware que no están presentes en el modelo de McCall.
El estándar ISO/IEC 9126 (Anexo 4: Modelo ISO 9126) ha sido desarrollado en un intento de identificar los atributos claves de calidad para un producto software, permitiendo evaluar y comparar productos sobre la misma base. Este estándar es una simplificación del Modelo de McCall, y es considerado un modelo universal que identifica seis características básicas de calidad las que propone que se dividan en sub-características y que pueden estar presentes en cualquier producto de software. En Cuba se adoptó una primera versión de la norma en 1995 para la evaluación del software médico y como norma cubana en el 2005.
Todo esto nos lleva a la conclusión de que las características de calidad que usaremos para lograr y evaluar los diferentes objetivos de las pruebas son las presentes en la propuesta de la ISO/IEC 9126 debido a que para cada uno de ellos, se ejecutan o implementan una serie de pruebas en diferentes niveles.

1.1.2 Características de las pruebas.

Para poder calificar una prueba como idónea las mismas deben cumplir con ciertas características que son vitales, dentro de las cuales no pueden faltar las siguientes (Moreno, 2003):

- Ha de tener una alta 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 sabemos lo que ha fallado.

1.1.3 Principios de las pruebas.

Para poder desarrollar un buen proceso de pruebas, es muy útil conocer sus principios, según (Pressman R. S., 2002) los mismos son:

- Se recomienda hacer un seguimiento hasta los requisitos del cliente. Donde el principal objetivo es encontrar errores.
- Las pruebas deben planificarse mucho antes de que empiecen, pudiendo comenzar tan pronto como esté completo el modelo de requisitos.

- El principio de Pareto es aplicable a las pruebas del software. Este principio plantea que el 80 por ciento de todos los errores descubiertos durante las pruebas surgen al hacer un seguimiento de sólo el 20 por ciento de todos los módulos del programa.
- Es recomendable empezar a probar por lo pequeño y progresar hacia lo más grande. Las primeras pruebas se deben centrar en módulos individuales del programa y a medida que avanzan las pruebas, se concentran en encontrar errores en grupos integrados de módulos y finalmente al sistema entero.
- No son posibles las pruebas exhaustivas.
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente al de desarrollo.

Un proceso de pruebas desarrollado bajo estos principios tiene una alta probabilidad de cumplir su objetivo fundamental, que es encontrar errores y contribuir a la calidad del producto.

1.1.4 Clasificación de las pruebas.

Para una mejor organización de las pruebas, las mismas se agruparon en métodos, niveles y tipos, con el objetivo de organizarlas a la hora de ser aplicadas para probar que el software satisface las necesidades del usuario y ha sido probado correctamente y en su cabalidad.

1.1.4.1 Niveles.

Las pruebas se dirigen con diferentes objetivos, en dependencia de los escenarios de trabajo, por esta razón es conveniente agruparlas por niveles; en dependencia de las etapas del proceso de desarrollo del proyecto.

Los niveles de prueba, no son más que un grupo de actividades experimentales que se organizan y administran de forma unida, ya que presentan una estrecha relación con las responsabilidades, normas y modelos que se decidan emplear en cada proyecto.

Según (Glossary of Testing Terms, 2005). Se clasifican de la siguiente manera:

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

- Pruebas de Sistema.

Una de las maneras de ver la aplicación de los niveles de prueba es a la hora de seleccionar los modelos, como el V y el W; en los cuales se especifican las pruebas que se van a realizar a lo largo del clico de vida del software organizadas por niveles.

1.1.4.2 Tipos de prueba.

Del mismo modo que las pruebas se agrupan por niveles también se clasifican en tipos, debido que cada uno de ellos responden específicamente a evaluar una funcionalidad del producto en específico.
Los tipos de pruebas son un grupo de actividades experimentales que permiten probar un componente o sistema. Pueden tener lugar en uno o más niveles de pruebas o fases. El (Glossary of Testing Terms, 2005) muestra las clasificaciones siguientes:

- Pruebas de Performance.
- Prueba de Volumen.
- Pruebas de Usabilidad.
- Pruebas de Integridad.
- Pruebas de Estructura.
- Pruebas de Estrés.
- Pruebas de Benchmarck.
- Pruebas de Contención.
- Pruebas de Carga.
- Pruebas Configuración.
- Pruebas Funcionales.
- Pruebas de Seguridad.
- Pruebas de Instalación.

1.1.4.3 Métodos.

Existen métodos de pruebas para buscar fallos en el software de acuerdo a criterios específicos, estos se denominan, Prueba de Caja Negra y Prueba de Caja Blanca a continuación se explican brevemente.
Las Pruebas de Caja Blanca se centran en la estructura de control del programa, obteniendo casos de prueba que aseguren que todas la líneas de código han sido ejecutadas por lo menos una vez. Mediante los métodos de prueba de la caja blanca el ingeniero puede obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se consideran las Pruebas de Caja Blanca como uno de los métodos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. (Pressman R. S., 2005).

Los métodos de Caja Blanca se pueden clasificar, a su vez, en dos grupos, los basados en *métricas de cobertura* y los basados en *métricas de complejidad.*

Métodos basados en métricas de cobertura: Todo programa se puede representar mediante un grafo de flujo de control, donde cada nodo es una sentencia o una secuencia de sentencias. Los arcos dirigidos en el grafo representan el flujo de control. En el que para cada conjunto de datos de entrada del programa se ejecutarán, a través de un camino concreto dentro de este grafo. Es conveniente resaltar que cuando el programa incluye estructuras iterativas, el número de posibles caminos en el grafo tiende a ser infinito.

Una Prueba de Caja Blanca exhaustiva requiere de la generación de un caso de prueba por cada posible camino. Como esto no es posible, por lo general, se utilizan métricas que dan una indicación de la calidad de un determinado conjunto de casos de prueba en función del grado de cobertura del grafo que consiguen. Las métricas más utilizadas son:

- Cobertura de sentencias
- Cobertura de segmentos entre decisiones.
- Cobertura de decisiones de ramificación
- Cobertura de condiciones
- Cobertura de todas las combinaciones de condiciones
- Cobertura de caminos

Las basadas métricas de complejidad más utilizadas en la generación de casos de prueba son las de McCabe:

- Complejidad ciclomática (arcos - nodos + 2 * número de componentes conexos)

- Complejidad esencial (complejidad ciclomática - número de subgrafos propios de entrada y salida única)
- Complejidad real (número de caminos ejecutados)

Las Pruebas de Caja Negra se centran en los requisitos funcionales del software, permitiendo al ingeniero del software obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Es válido aclarar que no suponen una alternativa a las pruebas de caja blanca, sino que permiten descubrir otra clase de errores, dentro de los que se encuentran los siguientes:

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

La prueba de caja negra intencionadamente ignora la estructura de control y se centra en el campo de la información. (Pressman R. S., 2005)

Una prueba de caja negra exhaustiva requeriría la generación de un caso de prueba por cada combinación posible (válida o no válida) de los valores de entrada, lo cual resulta imposible en la mayor parte de los casos por producirse una explosión combinatoria. Por lo que se utilizan diferentes métodos de selección a la hora de restringir el conjunto de casos de prueba a aplicar, dentro de los cuales se realizará una breve descripción de los más utilizados:

- *Método de clases de equivalencia:* Consiste en dividir las posibles entradas al sistema en clases de equivalencia, de tal forma que todos los miembros de una misma clase de equivalencia prueben las mismas propiedades en el sistema, por lo que sólo va a ser necesario seleccionar un elemento de cada clase de equivalencia.
- *Análisis de valores frontera o valores límites de calidad del software:* Consiste en seleccionar como casos de prueba aquellos valores de entrada que caen en la frontera de las clases de equivalencia (justo a un lado, justo al otro y justo en la frontera).
- *Grafos causa/efecto y tablas de decisión:* Consiste en crear un grafo causa/efecto a partir de las especificaciones, y seleccionar suficientes casos de prueba como para asegurar la cobertura del grafo. Se llama causas a las características de los datos de entrada y efectos a las clases de salidas que puede proporcionar el programa. A partir del grafo causa/efecto se construye

una tabla de decisión que refleje las dependencias entre causas y efectos. Lo que se hace entonces es reducir la tabla de decisión y seleccionar sólo un caso de prueba para todas las causas que producen el mismo efecto, o para cada columna de la tabla de decisión.

- *Adivinación de errores:* Consiste en tratar de imaginar cuáles son los errores que se pueden haber cometido con mayor probabilidad, y generar casos de prueba para comprobar dichos errores.

1.1.5 ¿Cuándo debe culminar la etapa de prueba?

Según el estándar (IEEE 1012-1998) 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 culminar 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 esta dado por un desfasaje de retardo en el cronograma de desarrollo del producto, se agota el dinero disponible para tal efecto o cuando los casos de pruebas no detectan errores. Según (Myers, 2005.): cuando se han detectados 4-8 errores por cada 100 líneas de código.

Existe un modelo llamado Modelo 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:

$$f(t) = (1/p)\ln(l_0 pt + 1) \qquad (1)$$

$f(t)$: 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 encuentran los errores y se van haciendo las correcciones.

La intensidad de fallo instantáneo, $l(t)$ se obtiene mediante la derivada de $f(t)$:

$$l(t) = l_0 /(l_0 pt + 1) \qquad (2)$$

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 del otro, sobre un número de puntos de datos, el modelo se puede usar para predecir el tiempo total requerido de prueba para alcanzar una intensidad de fallo aceptablemente baja Y poder determinar una fecha posible de culminación de las etapas de pruebas.

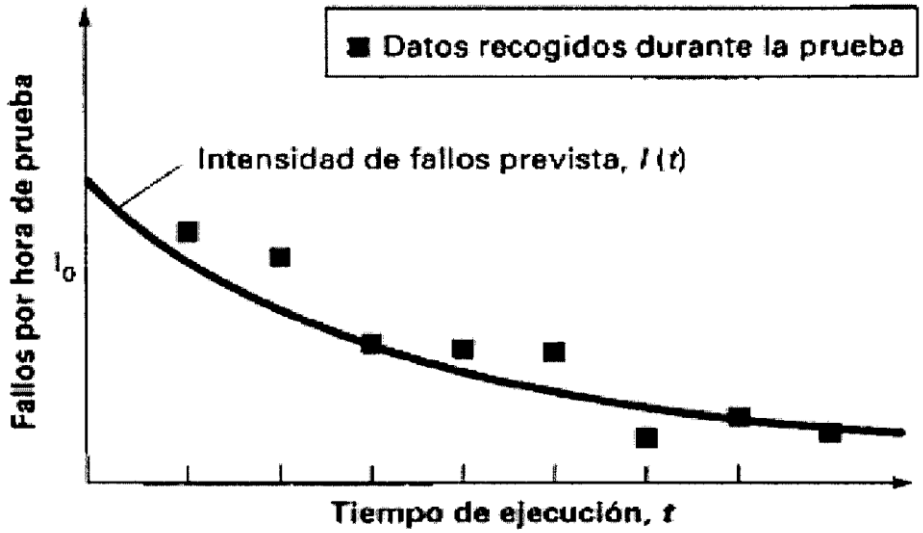


Figura1: Curva predicha.

Lo cierto es que 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.

1.2 El desarrollo del software y las pruebas.

El conjunto de actividades que se realizan de forma organizada y son iniciadas por la necesidad de un usuario, que al realizarlas contribuyen a obtener un producto como resultado final, puede dar la idea de que se está hablando del Proceso de Desarrollo de Software. Sin embargo según (ZAVALA, 2000) es aquel proceso en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño, el diseño implementado en código, el código es probado,

documentado y certificado para su uso operativo. Concretamente define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo.

El proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. (VALENCIA, 2003).

No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Pero si existen un conjunto de actividades fundamentales que se encuentran presentes en todos ellos, como son la especificación de software, el diseño e implementación, la validación, y la evolución del mismo.

A nivel mundial muy pocas empresas, dedicadas al desarrollo de software, poseen un alto nivel de capacidad sobre este, pero tienen definido sus propios procesos de desarrollo de software.

Ejemplo de ello es Hitachi, empresa japonesa dedicada a la producción y desarrollo de diversos tipos de aplicaciones para instituciones financieras, de seguros, control de inventarios y recursos humanos. En un principio Hitachi trabajó arduamente en la definición y medición de los procesos de desarrollo de software, incorporando elementos de medición y control de calidad. Sólo que no todo funcionó y surgieron múltiples problemas que tuvieron que resolver. Entre ellos la incapacidad de incorporar conceptos de fábrica de software como la estandarización del ciclo de desarrollo del proyecto y el re-uso de componentes. Un problema adicional fue que trataron de estandarizar un sólo proceso de desarrollo para cualquier tipo de aplicaciones. Lo que provocó proyectos fuera de fecha y presupuesto establecido.

Fue a finales de los años 70´s lograron organizar su fábrica de software alrededor de un manual que contenía enfoques de ingeniería y fábrica. En él, se incorporaron diversas técnicas del diseño y codificación estructurados así como tiempos estándares para cada actividad, inspecciones de productos y análisis de errores del proceso, entre otros elementos. Fue después de este esfuerzo que decidieron invertir en el desarrollo de sus propias herramientas de software para soportar sus funciones. A partir de este momento la fábrica logró una mejora impresionante en su desempeño, aumentado el aseguramiento de calidad en un 50%.

La Universidad de Ciencias Informáticas por su parte implementa el proceso de desarrollo de software a través de los proyectos[14] que se desarrollan en cada una de las facultades que la componen. Encontrándose cada proyecto compuesto por recursos, elementos de control y objetivos. Los recursos son los roles y el equipo de aseguramiento de la calidad. Los elementos de control son el tiempo y los

[14] Un proyecto es esencialmente un conjunto de actividades interrelacionadas, con un inicio y una finalización definida, que utiliza recursos limitados para lograr un objetivo deseado.

planes de iteraciones, de aseguramiento de la calidad del producto, de desarrollo, de prueba y de inspecciones. Mientras que el objetivo es alcanzar la calidad del software.

En la mayoría de las ocasiones se realiza un escaso o deficiente control en el progreso del proceso de desarrollo, estimaciones imprevistas de plazos y costos, no se realiza un proceso formal de pruebas, como tampoco revisiones técnicas formales e inspecciones de código y solo se tratan las pruebas en una parte de la elaboración del proyecto.

Lo que no satisface ni valida todo el proceso de desarrollo debido a que no se consiguen realizar pruebas que encuentren todos los errores o que por lo menos determinen una gran parte de los mismos. Por lo que se hace necesaria una estrategia que permita limar las incongruencias existentes y que le de un mayor peso a las pruebas para poder delimitar y eliminar los errores desde tiempos tempranos que atentan contra la calidad de los software producidos.

1.2.1 Ciclo de Vida del software.

El ciclo de vida del software es: “Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software” Según (IEEE 1074-1997).

Para desarrollar un ciclo de vida del software adecuado es necesaria la validación del producto que se está elaborando. Con esta intención la (ISO/IEC 12207.1-1997) propone una estructura común para los procesos del ciclo de vida del software, identificándolos como primarios, de soporte, y organizacionales. Sin definir un proceso de pruebas como tal, sino que aconseja, durante la ejecución de los procesos principales de la organización, utilizar los procesos de soporte, los cuales son:

El Proceso de Documentación es usado para registrar la información producida por un proceso o actividad del ciclo de vida. Contiene un conjunto de actividades, para planificar, diseñar, desarrollar, elaborar, editar, distribuir y mantener los documentos requeridos por todos los interesados tales como dirigentes, ingenieros y usuarios del sistema o del producto software.

El Proceso de Verificación es para determinar si los productos de una actividad cumplen las condiciones o los requisitos impuestos sobre ellos previamente. Las tareas fundamentales son: la verificación de contrato, del proceso, de los requerimientos, del diseño, del código, de la integración y de la documentación.

El Proceso de Validación es para determinar si los requisitos y el producto de software cumplen completamente la aplicación específica proyectada. Puede ser conducida desde las etapas más tempranas del proyecto.

El Proceso de Revisión Conjunta se emplea para evaluar si son adecuados los estados y productos de una actividad en un proyecto. Las revisiones conjuntas se realizan tanto a nivel del proyecto como a nivel técnico durante toda la vigencia del contrato. Este proceso puede ser empleado por cualquiera de las dos partes, donde una (parte revisadora) revisa a la otra (parte revisada).

1.2.2 Proceso de Verificación y Validación (V&V).

Un software o producto necesita ser controlado y revisado minuciosamente para evitar que sea entregado con errores. Por lo que se hace necesario realizarle pruebas, revisiones e inspecciones en cada momento. Con el objetivo de lograr mejoras en el trabajo técnico realizado sobre el producto, se debe realizar una verificación contra especificaciones, para determinar si se ha construido de forma correcta y además se debe validar, contra requisitos de usuario, para revisar si se ha construido el producto correcto.

La verificación soporta que las pruebas al software y sus productos asociados cumplan con los requisitos para todo el ciclo de vida, estas establecen una base para evaluar la terminación de cada actividad y poder iniciar otras.

Mientras que la validación provee una prueba para verificar que el software satisface requisitos de sistema, y soluciona el problema correcto.

El proceso de verificación y validación (V&V), en si, tiene dos objetivos fundamentales, el primero es el descubrimiento de los errores que pueda tener el sistema y segundo la evaluación del mismo para saber si es útil en situaciones operacionales o no. Tratando de establecer la confianza de que el software es adecuado para el objetivo con que se concibió, lo que no quiere decir que esté completamente libre de errores.

El estándar IEEE 1012 y CMMI definen el proceso de V&V, el cual pude ser aplicado en una organización con la intensión de mejorar la calidad del software que se quiera producir.

El estándar (IEEE 1012-1998) propone el proceso de V&V, precisando dentro del mismo actividades y tareas que se aplican al desarrollo, mantenimiento y reusabilidad del software, incluyendo además, su documentación. Dicho proceso debe tener interacciones con todos los componentes del sistema en cuestión; proveer una valoración objetiva de productos y los procesos a lo largo del ciclo de vida del software. Demostrando así que los requisitos del software son correctos, completos, precisos, coherentes, y que pueden ser probados, facilitando la detección temprana de errores. También propone que la V&V se realice junto al ciclo de vida del software debido a que las actividades y las tareas están relacionadas y son complementarias a cada fase del ciclo.

Por otro lado el Modelo CMMI, define la V&V como una de las áreas de proceso, en su nivel número 3 (Definido). La cual tiene el propósito de asegurar que los productos de trabajo seleccionados cumplan con los requisitos especificados. Involucrando la comprobación del producto, y de dichos requisitos. (SHRUM, Marzo 2006.).

Existen además algunas tareas definidas por el modelo CMMI para el proceso de V&V (Miguel Ángel Martínez), dentro de las que se tienen la verificación de requisitos, métodos y procesos, el establecimiento de procedimientos de verificación y criterios (Plan de verificación.), el análisis de los resultados de la verificación y la identificación de acciones correctivas. Además de la validación de requisitos del cliente y del producto (Plan de Validación de Productos) y el análisis de los resultados.

Debido a la necesidad de documentar los errores encontrados en todo el proceso de V&V que se este llevando a cabo, se vincula al mismo el estándar IEEE 829, que propone como redactar la documentación del desarrollo del software, especificando como realizar cada uno de los documentos de pruebas, los cuales sirven como lista de comprobación para el proceso de pruebas y como evaluación de las prácticas de documentación de dicho proceso. Facilitando así la comunicación entre el cliente y los desarrolladores. (IEEE 829-1998).

El proceso de V&V formulado por el estándar IEEE 1012 da soporte a los procesos primarios del ciclo de vida del software definido por la ISO/IEC 12207. Siendo efectivo cuando es conducido de forma paralela a dicho ciclo. Lo que constituye una vía de solución para los problemas existentes en la Universidad de Ciencias Informáticas a la hora de llevar a cabo el proceso de pruebas en el desarrollo de un software, debido a que demuestra que si desde el comienzo del proceso de creación de un producto, de forma paralela se comienza a llevar a cabo un proceso de V&V, los errores pueden irse detectando y tratando desde tiempos tempranos, proporcionando un software con mayor calidad.

1.2.2.1 Representación de la V&V.

La verificación dinámica y estática tiene dos representaciones las inspecciones y las pruebas de software, respectivamente.

Las inspecciones comprueban el ajuste con una especificación de los requerimientos pero no la confirmación con los requerimientos reales del cliente. Es decir no comprueban características no funcionales como rendimiento y usabilidad, debido a que se realizan sobre el código fuente, definiciones de diseño y planes de pruebas. Indicando además que no requieren de la ejecución del programa para su aplicación. Y son llevadas a cabo antes de las pruebas.

Las pruebas son una técnica de validación para requerimientos no funcionales. Ocupándose de la ejercitación y la observación del comportamiento del producto. Con el objetivo de detectar la mayor cantidad de errores en poco tiempo y con menos esfuerzo. Ejecutando el sistema con datos de pruebas para observar su comportamiento operativo.
Las representaciones de la verificación deberían utilizarse en el desarrollo de un software en conjunción con la intención de proporcionar al mismo una cobertura total de V & V.

1.2.3 Metodologías.

La manera de plasmar correctamente el ciclo de vida del software, es a través de la utilización de metodologías que no son más que el camino a seguir para desarrollar software de forma sistemática. Las metodologías son una guía para desarrollar un proceso de software con un nivel de calidad elevado. Y pueden seguir uno o varios modelos del ciclo de vida, englobando un conjunto de métodos para abarcar el ciclo de vida completo. Mientras que el ciclo de vida indica que obtener, pero no define como hacerlo. A continuación se realiza un estudio de algunas metodologías como XP, MSF y RUP.

1.2.3.1 RUP.

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, es un producto de Rational Software Corporation (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos que son los productos tangibles del proceso, como por ejemplo, el modelo de casos de uso, el código fuente y los roles.
Generalmente utilizada en la elaboración de proyectos grandes o a largo plazo. Divide en cuatro fases el ciclo de vida del software: Inicio, Elaboración, Construcción, Transmisión y a su vez cada fase de forma horizontal contiene flujos de trabajo, que se van desarrollando a medida que se va transitando por las mismas variando la complejidad en dependencia de la fase en que se encuentren.
Dentro en los flujos de trabajo se le dan gran importancia a las pruebas, que está presente en casi todo el proceso de desarrollo del software, con un enfoque disciplinado para asignar, gestionar tareas y responsabilidades en una organización de desarrollo. El mismo provee soporte para las prácticas de verificación y validación. Y generan una serie de artefactos como el Plan de pruebas, Diseño de pruebas, Línea de trabajo (revisiones a todos los productos en todos los flujos de trabajo). Además se ejecutan las pruebas, de integración y de sistema, y se corrigen errores. En el desarrollo iterativo que define RUP, las pruebas de regresión deben ocurrir en cada iteración. En caso que se necesite corregir bugs, se ejecutaría otra iteración con las mismas actividades de prueba, pero debe desarrollarse una

nueva versión del modelo de prueba que contendría entonces las pruebas anteriores (como pruebas de regresión) y las nuevas pruebas que tienen en cuenta las nuevas funcionalidades a probar.
RUP plantea que la validación debe comenzar desde horas tempranas con la revisión de los casos de uso con los usuarios y debe continuar con la evaluación del ejecutable obtenido en cada iteración mediante una configuración preliminar y con determinados usuarios.

1.2.3.2 XP (Extreme Programing).

Extreme Programing es una de las metodologías llamadas "ágiles", para el desarrollo de proyectos de software, en la actualidad es utilizada para proyectos de corto plazo, de pocos integrantes en el equipo y cuyo plazo de entrega era ayer. La misma 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. XP puede dividirse en cuatro principios sobre los que se va iterando hasta que el proyecto ha finalizado (el cliente aprueba el proyecto). Estas fases o principios son planificación, diseño, desarrollo y pruebas. Las pruebas se integran en un proceso continuo de construcción, lo que rinde una plataforma altamente estable para el desarrollo futuro. Constituyendo en la actualidad un proceso de diseño evolutivo que se basa en reorganizar un sistema simple en cada iteración, todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, combinando la disciplina con la adaptabilidad de una manera que la hace la más desarrollada entre todas las metodologías adaptables. (Beck, 2000).
XP se basa en las siguientes características que se explican a continuación, las mismas están vinculadas con el proceso de pruebas:

- Centra las pruebas Unitarias en el módulo. Usando como guía la descripción del diseño detallado, prueba los caminos de control importantes con la intención de descubrir errores dentro del ámbito del módulo. Haciendo uso intensivo de las técnicas de prueba de caja blanca.
- Se basa en la refabricación que es la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Una particularidad de esta metodología es que propone la programación en pares (peer review), la cual no es más que la participación de dos desarrolladores en el proyecto con la misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

1.2.3.3 Microsoft Solution Framework (MSF).

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de procesos y de equipo, dejando en un segundo plano las elecciones tecnológicas, es una metodología Adaptable, Escalable, Flexible, y de tecnología angosta.

En su definición consta de 5 fases, Visionado, Planificación, Desarrollo, Estabilización, Distribución, y en cada una de ellas le da a las pruebas la importancia que se merecen. En la fase Visionado propone retroalimentar al equipo sobre las meta de calidad de la solución y especifica las acciones que se necesitarán para alcanzar ese nivel de calidad. En la Planificación asegura que el plan cumpla con los requerimientos. Es responsable de evaluar el diseño para determinar qué características pueden ser probadas y de brindar un plan y calendario para dichas pruebas.

En el Desarrollo plantea la realización de la prueba funcional, la identificación de resultados, la documentación de pruebas, actualización del plan de pruebas, En Estabilización define la realización de pruebas como, las de detección y comunicación de errores y varias tareas a realizar como son probar la solución, implementación de planes de prueba para validar el producto. Además de algunas pruebas rigurosas como las pruebas de componentes, de seguridad, de integración, de aceptación y usabilidad del usuario, de estrés, capacidad y rendimiento, de regresión. En la fase de Distribución plantea la realización de pruebas de rendimiento y la resolución de problemas.

Definiendo varios entregables que están relacionados con el proceso de pruebas como el plan de pruebas, formatos de casos de prueba y listas de chequeo. (Sanchez, Junio 7 del 2004)

1.2.4 Modelos de Prueba.

Con el objetivo de representar el ciclo de vida del software se presentan como guía los modelos; los cuales están muy vinculados a la aplicación de las pruebas debido a que conjuga como las etapas de pruebas pueden ir aplicándose a lo largo del ciclo de vida. A continuación se explican algunos modelos como son: Modelo V y Modelo W.

1.2.4.1 Modelo V.

El modelo V es un marco para describir el programa de desarrollo de las actividades del ciclo de vida del software y la especificación de los requisitos para el mantenimiento. Este ilustra la forma en que las actividades de prueba se pueden integrar en cada fase del ciclo de vida del software. Mediante éste modelo se describe a un nivel muy alto de abstracción las fases de desarrollo en las que (idealmente) se involucra la prueba. (Glossary of Testing Terms, 2005).

Las actividades de prueba de la línea izquierda de la “V” se llevan a cabo en paralelo al desarrollo de software, las de la línea derecha involucran la terminación del diseño de los casos de prueba y la aplicación de los mismos. Siendo entonces la línea del medio el artefacto que organiza la relación entre las dos líneas anteriores.

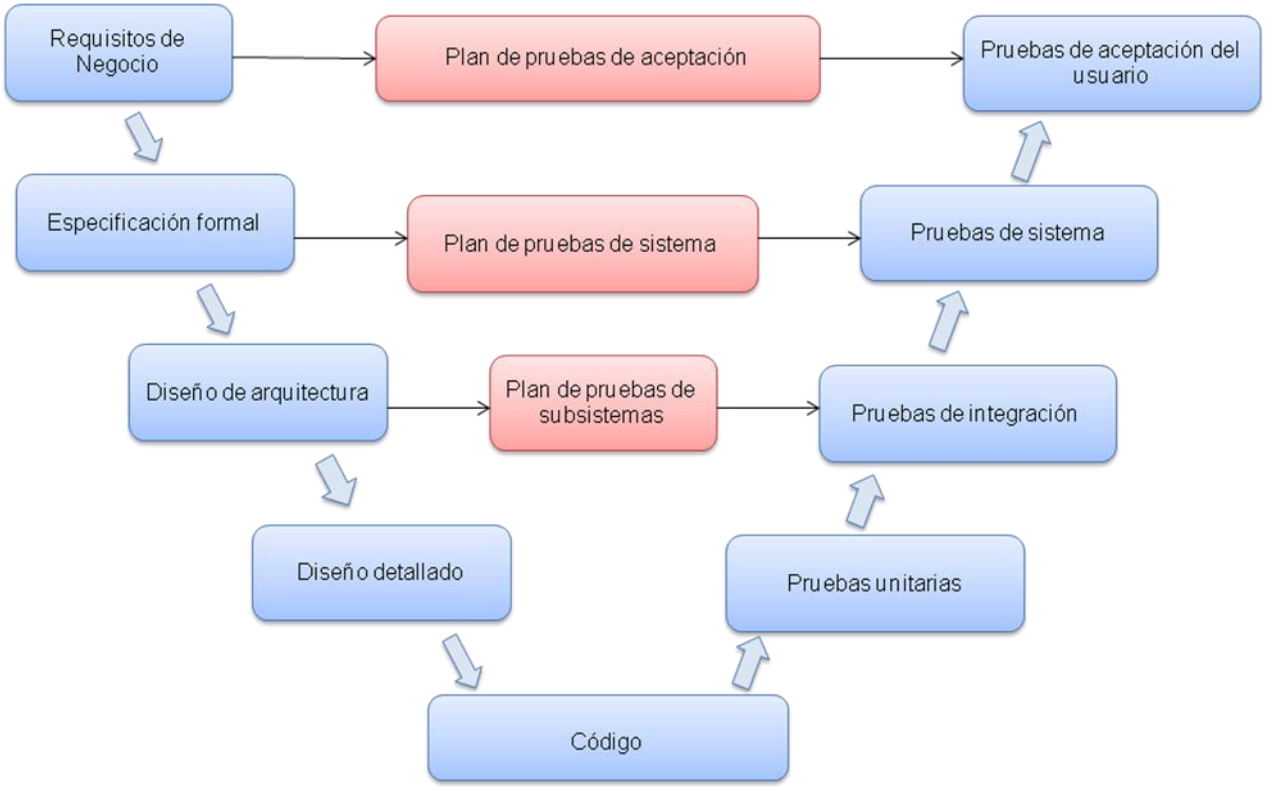


Figura 2: Modelo V.

Este modelo es bastante intuitivo, se comienza haciendo revisiones técnicas a los requerimientos, se esbozan los primeros casos de prueba de aceptación, pasando luego a revisar que la arquitectura satisfaga los requerimientos y a definir los primeros casos de prueba de sistema. Después se revisa la modularidad del diseño y se delinean casos de prueba de integración, para luego revisar los algoritmos y a desarrollar los casos de prueba de unidad.

Una desventaja del modelo es que requiere aún de mucho detalle para ser útil en la práctica. Además, de que la documentación de los procesos es una actividad en sí misma, y los documentos generados suelen ser muy propensos a quedar rápidamente inconsistentes (a causa de los cambios) y/o sin actualizar (por la dificultad para realizar esos cambios).

1.2.4.2 Modelo W

Surge como refinamiento del modelo en V. Refleja 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.

En las primeras etapas se consideran labores de pruebas 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, del diseño de detalle y las revisiones de código. En las etapas finales 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.

Las labores de pruebas aparecen en los bloques verdes. El resto de las actividades del ciclo de vida del software antes de su puesta en producción aparecen en color azul.

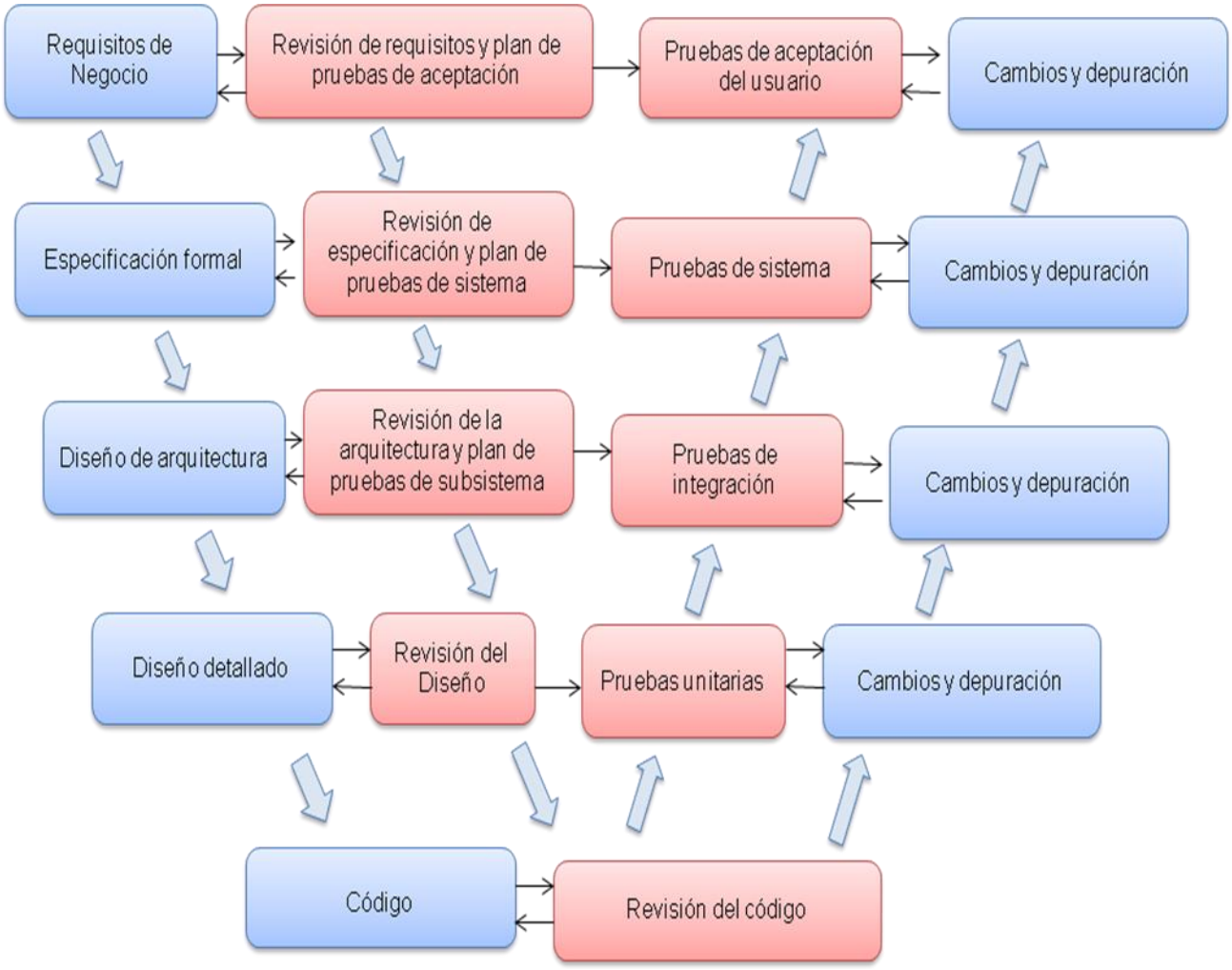


Figura 3: Modelo W.

Desde el punto de vista del equipo que realiza las pruebas, se pueden distinguir las siguientes 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.

Debido a las características que presenta este modelo y la detallada relación que propone entre el ciclo de vida del software y las pruebas, permitió decir que el manual y la estrategia propuesta se basen sobre el mismo.

1.3 Las pruebas en una organización.

El aseguramiento de la calidad de software se ha convertido en una necesidad prioritaria para las organizaciones que desarrollan software, ya sea para uso interno o para implementaciones externas a los clientes, los errores en el software repercuten directa o indirectamente trayendo graves consecuencias para la organización.

El uso de normas y modelos de calidad en la confección de productos software proporciona diversas ventajas a las organizaciones, pues ayuda a garantizar la calidad del software que se esté desarrollando. Considerándose una de las mejores maneras de garantizar que se cumpla con los requisitos solicitados por el cliente. Además su puesta en práctica reduce considerablemente la probabilidad o riesgo de ocurrencia de errores logrando una mayor fiabilidad.

Existen varias normas como la ISO 9001:2000, ISO 9000-3, SPICE y modelos como CMMI que tratan el proceso de pruebas dentro de una organización, evaluando en todo momento la posibilidad de

adaptar alguno de estos modelos a las unidades de producción de la UCI. A continuación se mencionan algunas.

1.3.1 ISO 9001:2000.

La ISO 9001:2000 es una norma internacional que se aplica a los sistemas de gestión de calidad (SGC), que se centra en los elementos de administración de calidad con los que una empresa debe contar para tener un sistema efectivo que le permita administrar y mejorar la calidad de sus productos o servicios. (ISO 9001:2000). La ISO 9001:2000 fue escrita para ser utilizada por toda clase de industrias, es regularmente difícil interpretarla para el desarrollo de software, por lo cual se publicó la ISO 9000-3 que sirve de guía para la aplicación de ISO 9001 para el desarrollo, implementación, funcionamiento y mantenimiento de software.

Esta norma establece que la empresa debe asegurar que los productos adquiridos no se utilicen o procesen hasta que sean inspeccionados o verificados que cumplen con los requerimientos específicos. Las verificaciones deben estar de acuerdo con el plan de calidad y los procedimientos documentados previamente establecidos. Para aquellos productos que son enviados a producción por situaciones de urgencia sin ser antes inspeccionados, éstos deben identificarse y registrarse para que en caso de no conformidad sean rápidamente reconocidos y reemplazados. La empresa debe establecer o mantener registros que contengan el criterio de aceptación del producto.

1.3.2 ISO 9000-3.

La ISO 9000-3 es una norma internacional, que plantea que el control de calidad debe ser aplicado a todas las fases de la producción de software, incluido el mantenimiento y tareas posteriores a su implantación. Que exista una estricta colaboración entre la organización que adquiere el software y el proveedor del mismo, que el proveedor del software debe definir su sistema de calidad y asegurarse que toda la organización ponga en práctica este sistema. (ISO 9000-3)

Permitiendo así una mejor documentación de los sistemas, un cambio cultural positivo, incrementar la eficiencia y productividad, una mayor percepción de calidad, además con su uso se amplía la satisfacción del cliente, se reducen las auditorias de calidad de los clientes y se agiliza el tiempo de desarrollo de un sistema.

Esta norma tiene definida entre sus secciones una dedicada a la inspección y pruebas, que guarda estrecha relación con el proceso de prueba que se desarrolla en determinados productos software a lo largo de su 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 cumplen con los requerimientos

específicos. Las verificaciones deben estar de acuerdo con el plan de calidad y los procedimientos documentados que ya estén establecidos. Para aquellos productos que son enviados a producción por situaciones de urgencia sin ser antes inspeccionados, éstos deben identificarse y registrarse para que en caso de no conformidad sean rápidamente reconocidos y reemplazados. La empresa debe establecer o mantener registros que contengan el criterio de aceptación del producto. (Referencia a la tesis de maestría de Michael González Jorrín, Diciembre del 2007).

1.3.3 ISO/IEC 15504.

Es un modelo con dos propósitos, documentar el conjunto de prácticas fundamentales para una buena ingeniería de software y describir los procesos que una organización puede realizar para adquirir, suministrar, desarrollar, explotar, evolucionar, soportar software y las prácticas genéricas que caracterizan la capacidad de esos procesos. (ISO/IEC TR 15504-1, 1998.) Consta de 6 niveles:

- Nivel 0: No Realizado.
- Nivel 1: Realizado informalmente.
- Nivel 2: Planificado y seguido.
- Nivel 3: Bien definido.
- Nivel 4: Controlado cuantitativamente
- Nivel 5: Mejorando continuamente.

SPICE enmarca las pruebas de software en la categoría de Procesos de Ingeniería. La cual tiene como propósito confirmar que el producto software integrado reúne los requisitos definidos. Para ello establece un grupo de prácticas base dentro de las que plantea: desarrollar las pruebas de integración del producto software y las pruebas de sistema. Realiza pruebas empíricas para evaluar si se están cumpliendo los objetivos antes de que la propuesta se vuelva estándar; ayudando a mejorar el contenido de los documentos, proporcionando una guía para su uso en la práctica.

1.3.4 CMMI.

CMMI es un modelo permite el alcance y la visibilidad, en las actividades del ciclo de vida y de la ingeniería de producto asegurándose de que las expectativas del cliente respecto al producto o al servicio serán óptimamente satisfechas. Determina áreas de procesos, brindando procedimientos para mejorar dichos procesos de forma incremental y permite el avance del proyecto sin tener que aplicar el modelo completo.

Recomienda para el proceso de pruebas la revisión por pares (peer review), que no es mas que la evaluación científica en la búsqueda y propósito para la competencia y originalidad por expertos

calificados que buscan crear productos de la misma forma. Propone que dos desarrolladores trabajen juntos escribiendo el mismo código. En general, el aseguramiento de la calidad en peer review recae en las pruebas de código en pareja y en la interacción del cliente (IEEE Std 1028-1997). Teniendo como beneficios el intercambio de conocimientos entre los expertos y desarrolladores del software, el aumento de la calidad y seguimiento del producto, la mejora de los procesos, tiene como finalidad eliminar los errores más rápido, fácil y reduce el costo del desarrollo del software.

CMMI exige además la preparación del ingeniero de manera individual, por lo que es necesario que haya practicado PSP (Personal Software Process), el cual es una guía para el planeamiento de módulos de software o pequeños programas. Implantando una determinada disciplina a las prácticas del ingeniero. Lo que proporciona la mejora de la calidad del producto, aumentando los costos y reduciendo el tiempo del ciclo de desarrollo del software.

Una vez dominada la practica de PSP para poder aplicar de manera correcta CMMI, es necesario además que los ingenieros estén preparados y sean capaces de integrar equipos, aquí es donde sale a relucir el TSP (Team Software Process), que tiene como objetivos: ayudar a los equipos de Ingeniería de Software a elaborar productos de calidad dentro de los costos y tiempos establecidos, tener equipos rápidos y confiables; y optimizar el performance del equipo durante todo el proyecto. Proporcionando ingenieros bien preparados y capaces de crear software de la manera más optima posible. (SEI)

La UCI ha hecho una importante inversión para la preparación de varios de sus especialistas en cursos de formación de CMMI, realizándose estudios del mismo para fundamentar su uso.

1.4 Soporte al proceso de pruebas.

Una práctica muy beneficiosa es realizar las pruebas de forma automatizada, para esto existen herramientas que dan soporte el proceso de pruebas, ejemplo de ellas:

Administración de Pruebas	Rational TestManager
Seguimientos de errores	Rational ClearQuest
Capture/Play back	Rational Robot
Administración de proyecto	Microsof Project
DBMS Tools	Motor de Base de Datos

Tabla 1: Herramientas de soporte al proceso de pruebas.

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, da seguimiento y maneja eficientemente los diferentes tipos de cambios. Permite manejar de forma eficiente la trazabilidad dentro del proyecto.

Capture/Play back: automatiza la ejecución de pruebas, evitando que las pruebas tengan que re-ejecutarse repetidamente en forma manual.

Rational Robot: permite automatizar las pruebas de regresión de aplicaciones java, .net, web y otras aplicaciones basadas en interfaz grafica de usuario.

Microsoft Project: software para la administración de proyecto fue diseñado para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.

Rational Manual Tester: mejora la auditoria y la ejecución de las pruebas manuales por medio de nuevas técnicas de diseños de pruebas.

Rational Funcional Tester: pruebas funcionales automatizadas y avanzadas para aplicaciones Java, Web y VS.Net.

Existen además otras vías que contribuyen a automatizar el proceso de pruebas, entre las mismas podemos encontrar las plantillas del expediente de proyecto y el estándar (IEEE 829-1998) que propone una serie de procesos para cada una de las pruebas a realizar en cada momento del ciclo de vida del software constituyendo en si una vía para el soporte al proceso de pruebas.

1.5 Conclusiones Parciales.

- Como se explicó anteriormente, el estándar IEEE 1012, propone que las pruebas se realicen de manera paralela al desarrollo de software, definiendo un proceso de V&V para el ciclo de vida del software establecido por la ISO/IEC 12207, con la intención de limar los posibles errores desde etapas muy tempranas, y ver si el software en cuestión, responde a los requerimientos especificados.
- Debido a las características que posee el modelo W y la detallada relación que propone entre el ciclo de vida del software y las pruebas, fue lo que permitió decir que el Manual y la Estrategia propuesta se basen sobre el mismo.

CAPÍTULO II: MANUAL DEL INGENIERO DE PRUEBA.

2. Introducción.

El trabajo de los ingenieros de prueba se ve afectado debido al poco tiempo que se le dedica a la planificación de esta etapa. En muchas ocasiones no se definen correctamente las actividades a realizar en cada una de las fases del ciclo de vida del proyecto. Provocando que las revisiones realizadas sobre dicho producto, no arriben a los resultados esperados de detectar la mayor cantidad de errores en el menor tiempo posible. Por estas razones se propone un Manual para los Ingenieros de Prueba, en el cual se definan un conjunto de actividades y tareas que permitan organizar el proceso personal de pruebas durante todo el desarrollo del software, basado en el modelo W, asegurando la calidad del software producido.

2.1 Introducción al Manual.

El Manual propuesto está dirigido a los ingenieros de prueba, presentando tablas donde se muestra la guía del contenido y mapas conceptuales a lo largo del mismo. Permitiendo un mejor entendimiento y ubicación del probador sobre las tareas que debe realizar en cada fase del ciclo de vida del software. Lo principal que se necesita saber es en qué momento del ciclo de vida del software se encuentra situado.

Según (IEEE Std. 610.12-1990) el ciclo de vida del software es el período de tiempo que comienza cuando el producto software es concebido y termina cuando el software no está disponible permanentemente para el usuario.

La ISO/IEC 12207 Procesos del ciclo de vida del software, establece un marco común para el ciclo de vida de un software, definiendo procesos primarios, de soporte y organizativos; con las actividades de cada uno de los procesos y sus tareas correspondientes, implementadas durante el desarrollo del software.

De todos los proceso definidos por la ISO/IEC 12207, los primarios son los seleccionados para u utilización en el Manual que se propone, debido a que el Estándar 1012-1998 para la Verificación y Validación del Software; propone actividades y tareas de V&V para cada uno de esos procesos. A continuación se muestra un mapa conceptual con cada proceso y la relación con el estándar 1012 que sirve para definir la estrategia de prueba a proponer.

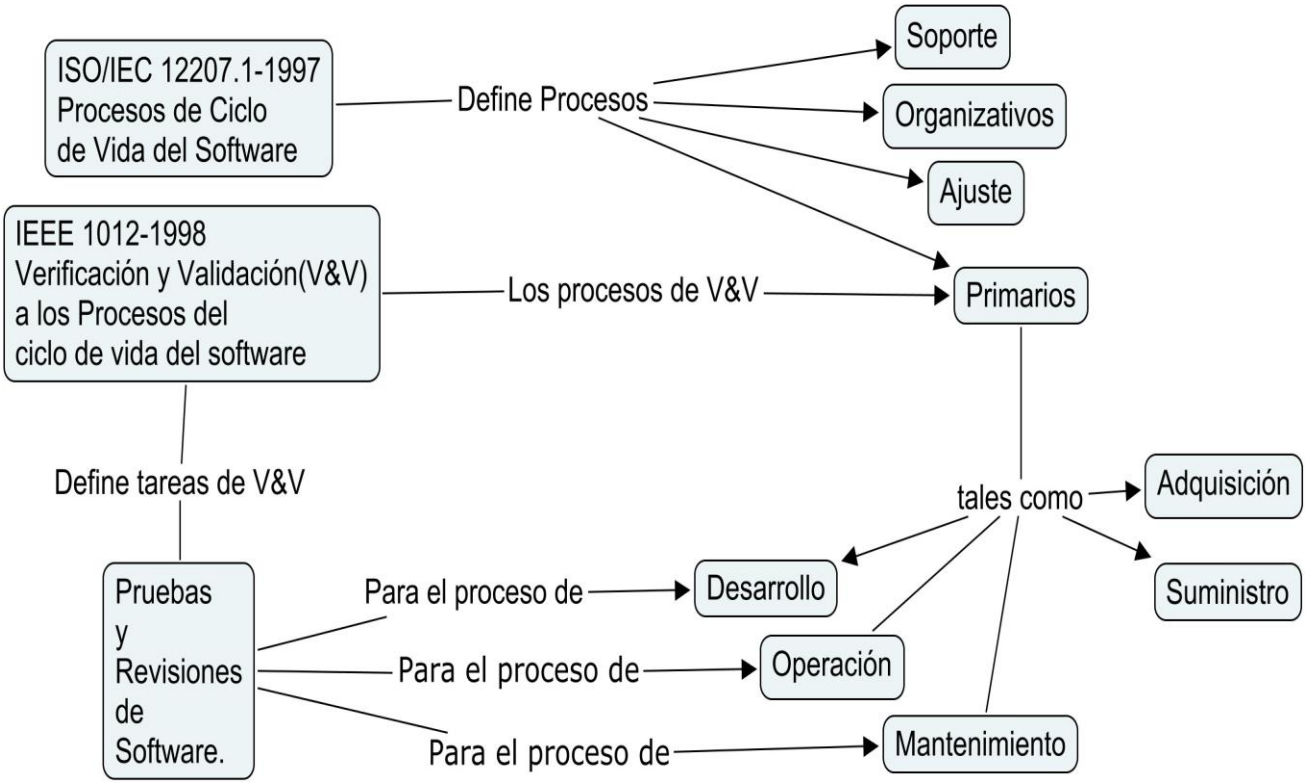


Figura 4: Relación ISO/IEC 12207 y IEEE 1012-1998.

La figura 4 muestra la relación que existe entre los procesos del ciclo de vida del software definidos según la ISO/IEC 12207 y las tareas generales de V&V a los procesos primarios determinadas por el estándar IEEE 1012.

En esta figura se resaltan los procesos primarios que son los que se analizan en el manual, ya que en el mismo se tratan específicamente las actividades relacionadas con las pruebas e inspecciones al software, durante estos procesos. Por esta razón se cree necesario explicar la concepción de cada uno de estos procesos.

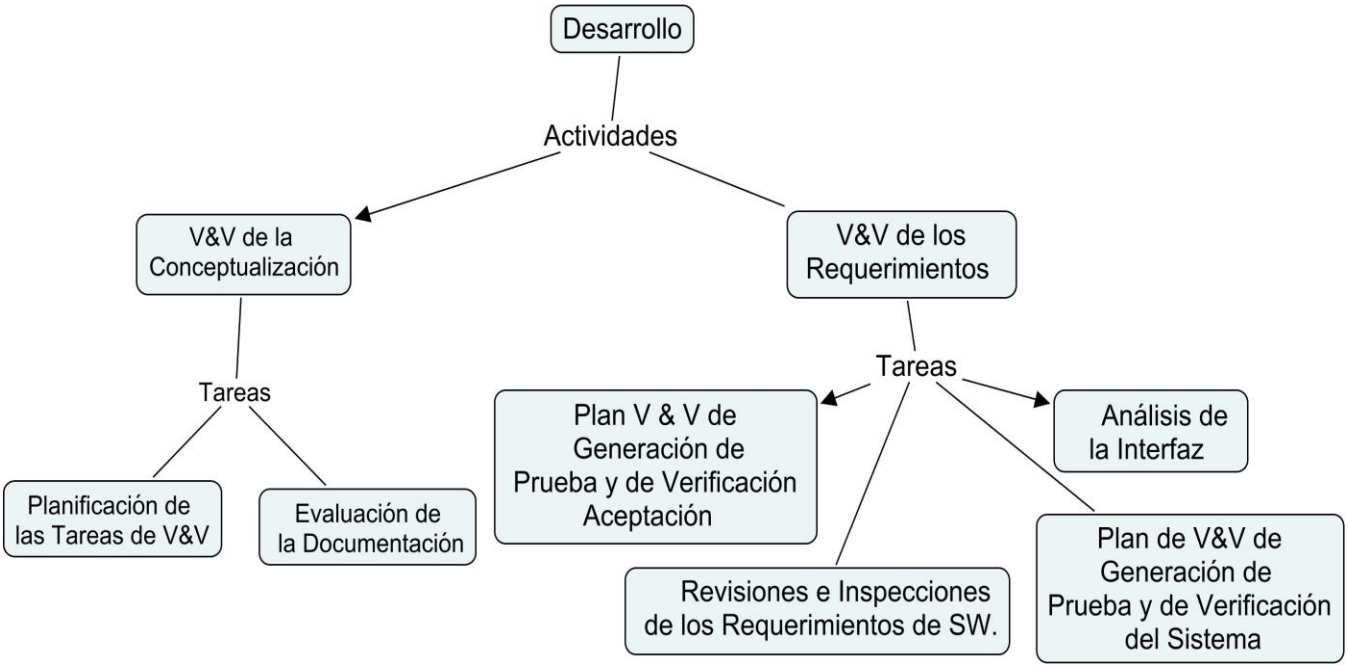
Los procesos primarios se descomponen en Adquisición, Suministro, Desarrollo, Operación y Mantenimiento. Debido a poca actividad relacionada con las pruebas en los procesos de Adquisición y Suministro se decide utilizar solamente los tres restantes.

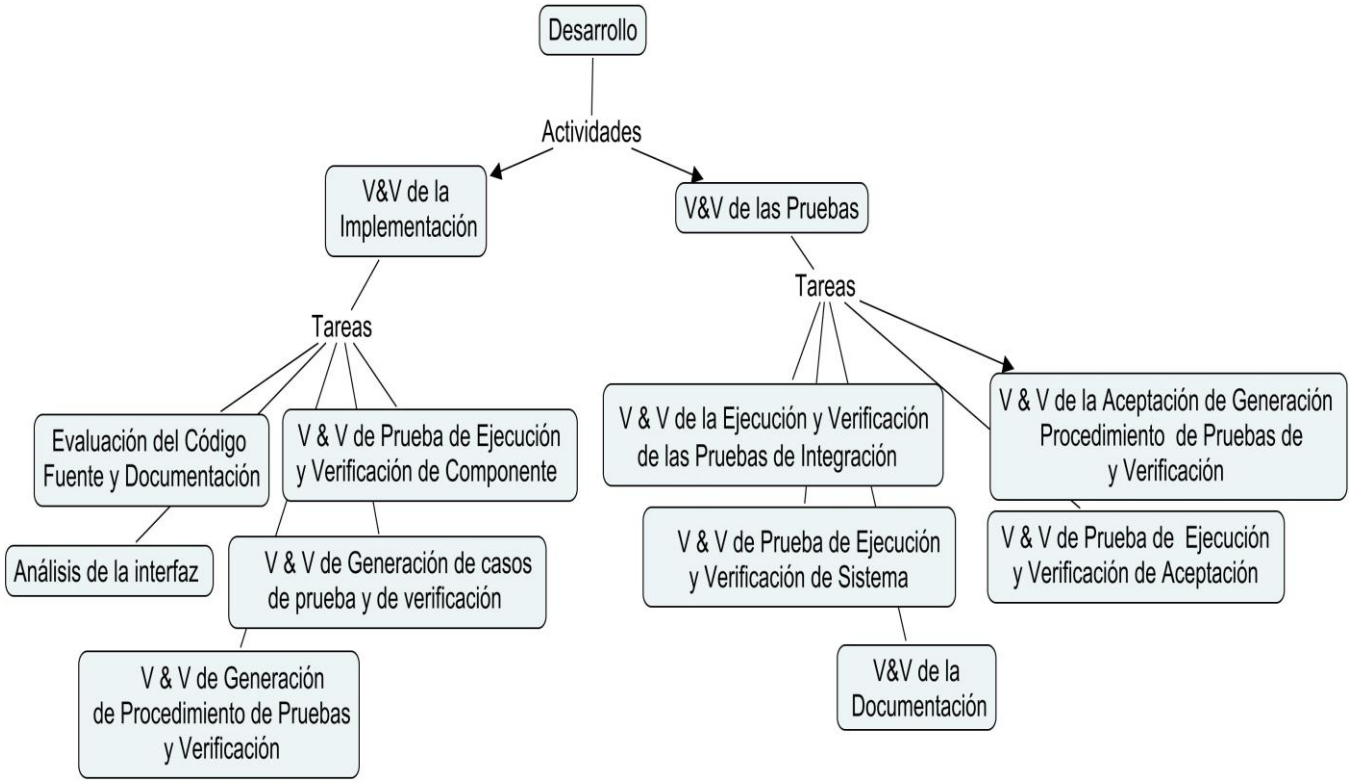
El proceso de Desarrollo: contiene las actividades y tareas para los requisitos, el análisis, el diseño, la codificación, la integración, las pruebas, la instalación y la aceptación relacionados con productos de software. Cubriendo desde que se comienza a desarrollar el software hasta casi terminado el producto. Es elegido este proceso debido a la importancia que tiene aplicar la V&V desde la iniciación del negocio, hasta el despliegue, para mejorar el funcionamiento del producto.

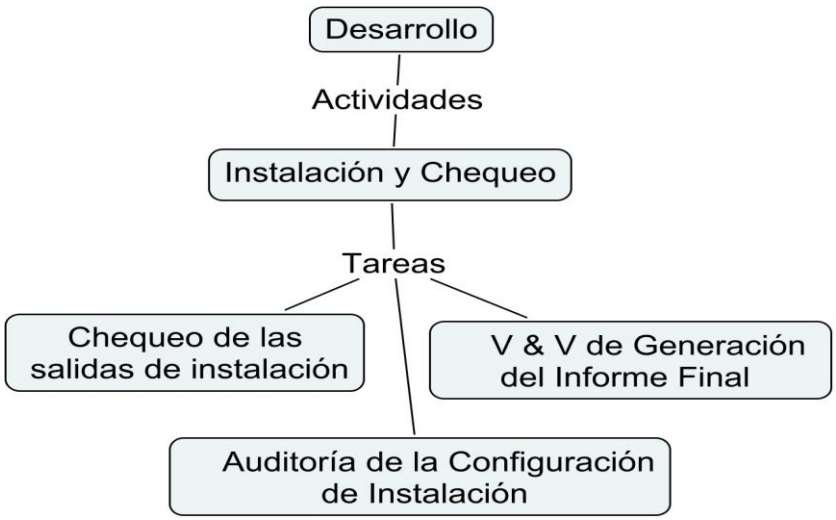
El proceso de Operación: comprende el funcionamiento del producto de software y el apoyo operacional a los usuarios. Es necesaria su utilización debido a que la actividad de V&V que es equivalente a este proceso aporta mejoras al producto creado, proporcionando la evaluación de los posibles cambios que puedan producirse, y de los procedimientos para saber si se cumple con el uso determinado, analizando también los riesgos que puedan afectar al usuario y al propio sistema.

El proceso de Mantenimiento: una vez culminado el proceso de operación, se pasa al de mantenimiento que es cuando el producto software sufre modificaciones en el código y/o la documentación asociada, causada por un problema, una necesidad de mejora o adaptación en a versión del producto.

Con la intención de mejorar el proceso de pruebas y ofrecer un orden a la hora de comenzar la V&V a lo largo del ciclo de vida del software, se realizo una esquematización de las actividades y tareas de V&V seleccionadas para cada proceso, que se tienen en cuenta en el Manual.




Desarrollo
Actividades
V&V de la Implementación
V&V de las Pruebas
Tareas
Tareas
Evaluación del Código Fuente y Documentación
V & V de Prueba de Ejecución y Verificación de Componente
V & V de la Ejecución y Verificación de las Pruebas de Integración
V & V de la Aceptación de Generación Procedimiento de Pruebas de y Verificación
Análisis de la interfaz
V & V de Generación de casos de prueba y de verificación
V & V de Prueba de Ejecución y Verificación de Sistema
V & V de Prueba de Ejecución y Verificación de Aceptación
V & V de Generación de Procedimiento de Pruebas y Verificación
V&V de la Documentación


Desarrollo
Actividades
Instalación y Chequeo
Tareas
Chequeo de las salidas de instalación
V & V de Generación del Informe Final
Auditoría de la Configuración de Instalación

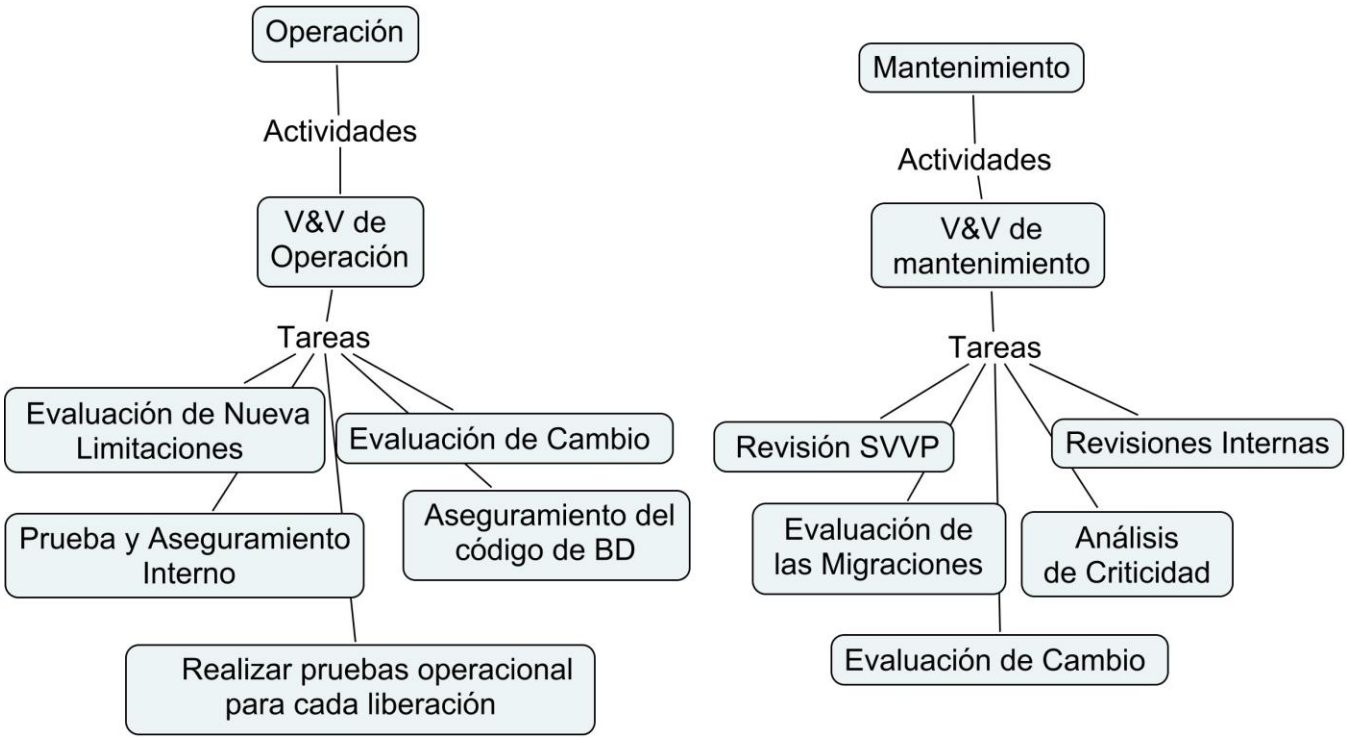


Figura 5: Actividades y tareas de V&V.

Una vez definido cada proceso dentro del ciclo de vida del software, será fácil la utilización del Manual, debido a que cada vez que se enfrente al mismo sabrá ubicarse donde se encuentra, posibilitando la aplicación de cada una de las actividades y tareas de V&V correspondiente a ese proceso, de igual manera el tipo de prueba en caso que sea necesario.

2.1.1 El PSP en las pruebas de software.

El PSP (Proceso Personal de Software) define un conjunto de actividades que guían a los desarrolladores de software a la hora de realizar su trabajo. (Marsha Pomeroy-Huff, Agosto 2005). El mismo se basa en la experiencia personal del ingeniero. Proporcionándole un uso constante de buenas prácticas de planificación y seguimiento de su trabajo. Además de mostrarle como establecer metas mesurables que puedan ser alcanzadas. Para lo cual, es obligado a registrar cada una de las actividades realizadas en el cuaderno de registro de tiempos; enfocándolos en la administración del tiempo y la calidad del software. Brindando así a cada ingeniero grandes habilidades, enriquecer sus conocimientos, y una mejor base a la hora de enfrentar un equipo eficaz.

El PSP organiza el proceso personal de software para un ingeniero, que se enfrenta al desarrollo de un software en su totalidad, sin definir las actividades según los roles que se establecen en un proyecto. Por esta razón es muy importante para el manual propuesto que el ingeniero de prueba conozca los métodos para mejorar la calidad de su trabajo, debido a que es el responsable de diseñar y desarrollar pruebas al software en construcción, logrando tener como resultado una estrategia de pruebas, la configuración del entorno de pruebas, un resumen de la evaluación de pruebas, el plan de pruebas, los casos de pruebas, la arquitectura de automatización de pruebas, los scripts de prueba, logs de pruebas, los datos de prueba y por ultimo los resultados de pruebas.

Llegando a la conclusión que el ingeniero de prueba puede utilizar el PSP para organizar y planificar su trabajo e identificar las tareas que realiza, estimando cuanto tiempo necesitará para cada tarea y medir el tiempo gastado.

2.1.2 Integridad del Software.

La integridad del software no es más que una descripción de la aplicación y uso pretendido de un sistema. Denotándose así un rango de valores de criticidad para mantener los riesgos dentro de los límites suficientemente aceptables. El software de integridad elevada precisa de una aplicación rigurosa de tareas de verificación y validación. (IEEE 1012-1998).

La integridad a los niveles del software puede ser asignada a los requisitos, grupos de funciones, componentes o subsistemas; y puede discrepar en dependencia del desarrollo del mismo. El diseño, la codificación, la implementación de la tecnología elegida puede proporcionar el aumento o descenso de la integridad; convirtiéndose la mitigación de riesgos en uno de los aspectos que puede usarse también para reducirla.

La asignación del nivel de integridad del software es continuamente actualizada y revisada, conduciendo las tareas del análisis de verificación y validación de la integridad en todo lo largo del proceso de auge del software. El estándar IEEE 1012, propone un esquema de integridad de cuatro niveles como método para definir las tareas mínimas de verificación y validación asignadas a cada nivel de integridad:

Integridad	Descripción	Nivel
Alto	Funciones que afectan el comportamiento critico del sistema.	4
Mejor	Funciones que afectan el comportamiento importante del sistema.	3
Moderado	Funciones que afectan el comportamiento del sistema pero pueden ser implementadas estrategias para compensar perdidas.	2
Bajo	Funciones seleccionadas que tienen efectos notables en el comportamiento del sistema, pero solo crea inconveniencia para el usuario si el comportamiento de esa función no es acorde a los requisitos.	1

Tabla 2: Niveles de integridad del software.

La integridad es usada para asegurar que los procesos y actividades que se lleva a cabo a la hora de desarrollar un software son lo suficientemente completos, y que una vez integrados los mismos en el sistema que conformarán como tal, no provocarán modificaciones de carácter negativo sobre las personas y el ambiente.

También es usada para eliminar costes innecesarios, de modo que el software de baja integridad no sea desarrollado con excesivos costos y sofisticados procesos. Y para que los procesos de software usados en distintos proyectos sean lo más uniformes posibles, para cada nivel de integridad. (Revista Española de Innovación, Calidad e Ingeniería del Software, 2005.)

2.1.2.1 Cómo se trata la integridad del software en la UCI.

El laboratorio de calidad, en la Universidad de Ciencias Informáticas, tiene definidos tres niveles de de calidad a la hora de liberar un software:

Liberados: cuando cumple con los niveles de calidad establecidos. Como constancia se emite un aval del estado del producto. El aval necesita de un valor mínimo de No Conformidades no significativas para su emisión positiva para ello la calidad y estabilidad del producto debe ser comprobada.

Rechazados: los que tienen errores solubles en próximas iteraciones del proceso de comprobación tienen altas probabilidades de terminar liberados. En este caso se emite solo un informe detallado de No Conformidades del producto respecto a los requisitos declarados, Normas y Principios establecidos.

Críticos: cuando están por debajo del mínimo de calidad establecida. Se establecen parámetros para declarar un producto del desarrollo software en estado crítico de terminación. Son aplicables en tres momentos fundamentales: Cuando se hace la revisión de la solicitud (RS); Durante el proceso de pruebas exploratorias (PE) y Durante las pruebas formales. (PF).

Por lo que la integridad del software es asignada a parámetos para declarar un producto de desarrollo software en estado crítico de terminación. Estos parámetros son aplicables en tres momentos fundamentales: Cuando se hace la revisión de la solicitud (RS); Durante el proceso de pruebas exploratorias (PE) y Durante las pruebas formales. (PF). Quedando establecidos los criterios de criticidad como los siguientes:

- No se presenta la última versión del producto de trabajo comprobada y avalada por la etapa anterior.
- No están presentes todos los elementos componentes del sistema, producto o entregable. (Hardware, Software, Partes y Documentación). (RS) (PE)
- No se corresponden totalmente los requisitos funcionales documentados con los implementados. (RS) (PE)
- Supera el producto la taza de: 1 defecto significativo por CU para (PF) ó ½ defecto significativo por CU para (PE) y en caso de otros artefactos al menos 2 errores significativos que estén relacionados con las pautas esenciales definidas por la metodología que sigue el proyecto. (PF)
- Excede el producto el número máximo de iteraciones establecidas por plan (entre 2 y 3 iteraciones). (PF)
- Se incumple más del 50% de las reglas para la documentación establecidas por el proyecto y/o expediente del proyecto. (RS) (PE) (PF)
- Existe incoherencia significativa entre los artefactos que tienen relación o dependencia entre sí. (RS) (PE) (PF)
- Existe incoherencia significativa entre lo documentado y lo implementado. (RS) (PE) (PF)
- La cantidad de faltas de ortografía excede la cantidad de páginas o pantallas que tiene el artefacto en cuestión. (RS) (PE) (PF)

- Durante las pruebas de regresión persisten al menos 2 errores significativos de iteraciones anteriores. (PE) (PF)
- Se observan los mismos tipos de errores, ya señalados en iteraciones anteriores, en otros o los mismos lugares donde fueron detectados, para (PE): ½ de la cantidad de errores tipo encontrados en etapa anterior, o para (PF): ¼ de la cantidad de errores tipo encontrados en etapa anterior.

Si se encuentra alguno de estos criterios de integridad, se considera la prueba fallida y se suspenden las pruebas hasta el día siguiente como mínimo.

En el manual se muestran las tareas fundamentales que debe desarrollar el ingeniero de pruebas, proponiendo que se deba dar un nivel de integridad a dichas tareas de acuerdo al nivel de funcionalidad y peso que las mismas tengan dentro del sitema que esta evaluando. Con la intencion de dar un nivel de pruebas proporcional a la integridad del software.

2.2 Estructura del Manual del Ingeniero de Prueba.

El Manual que se propone describe como llevar a cabo el proceso personal de pruebas de software, estableciendo un conjunto de actividades que guían al ingeniero de prueba en el desarrollo de las mismas. Permitiendo elevar el conocimiento de los ingenieros, organizar y estandarizar el comportamiento de los mismos; en las revisiones técnicas formales (RTF) y pruebas del producto dentro del proceso de verificación y validación.

Según (IEEE 1012-1998)el proceso de V&V se debe aplicar a todo el desarrollo del ciclo de vida del software, desde sus inicios hasta el mantenimiento después de la entrega al usuario final. Por esta razón en el manual que se propone se define una estrategia de pruebas; basada en un conjunto de actividades y tareas de V&V a realizar en cada etapa; describiendo detalladamente cuales son las revisiones y tipos de pruebas a seguir en cada tarea. Todo esto sustentado bajo una modificación del modelo W, proponiendo que se desarrolle de manera paralela la estrategia con el ciclo de vida del software propuesto.

Este manual consta de 4 partes:

1. *Punto de partida para la utilización del manual:* en esta primera parte se evalúan los conocimientos previos que puede tener el ingeniero de prueba, mediante un conjunto de preguntas a las que tendrá que darle respuesta para definir cual es su punto de partida a la hora de enfrentarse al manual; para lo cual se proponen las siguientes tablas, Tabla 3: NC,

temas generales del proceso de pruebas. Y Tabla 4: NC, tema métodos, niveles y tipos de prueba.

2. *Preparación inicial para la utilización del manual:* con el fin de asegurar una preparación previa sobre la concepción del manual, se describen las, actividades, tareas y tipos de pruebas a desarrollar en cada fase del desarrollo del software.
3. *Estrategia definida para la utilización del manual*: se explica la estrategia de prueba a seguir durante el desarrollo del ciclo de vida del software, definiendo en que consiste cada actividad que la compone, que tareas se realizan dentro de la misma, quien es el responsable de su realización y las entradas y salidas. Además de cómo representarla en el modelo propuesto.
4. *Glosario de término:* se brinda un glosario de término, con la finalidad de enriquecer y facilitar el entendimiento del manual propuesto.

Para un mayor entendimiento de la estructura del manual que se propone, se realizó el siguiente esquema mostrado en la *Figura 6: Estructura del Manual del Ingeniero de Prueba.* La cual representa la propuesta de la estructura elegida para el manual que se propone; con la intención de poder guiar a los ingenieros de pruebas durante todo el ciclo de vida del software.

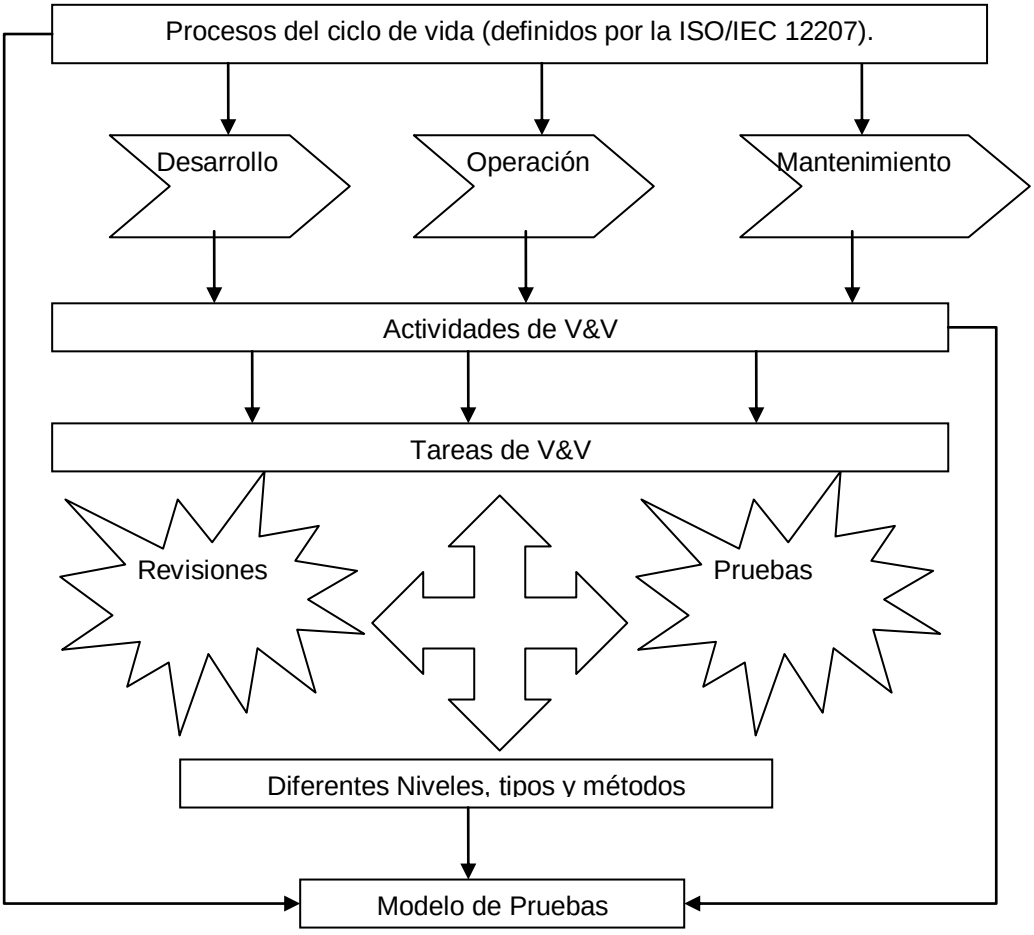


Figura 6: Estructura del Manual del Ingeniero de Prueba

Primeramente partimos de definir las fases del ciclo de vida del software, dentro de las cuales determinamos las actividades a desarrollar en cada una de ellas y a su vez se descomponen en tareas de V&V, incluyendo además la evaluación, el análisis, revisión, inspección, y las pruebas de los productos de software.

Estas actividades cumplen con las norma ISO / IEC Std 12207,1074-1997 y IEEE / EIA 12207.0-1996 Procesos del ciclo de vida del software y IEEE 1012-1998 Estándar para Verificación y Validación del Software, así como toda la familia de estándares IEEE de ingeniería de software.

Las fases definidas para el desarrollo del software en conjunto con las actividades de V&V, muestran el modelo de pruebas a seguir, después de realizadas las modificaciones pertinentes del moldeo W analizado.

2.3 Relación entre 12207, RUP y las actividades de V&V propuestas.

El flujo de trabajo de pruebas definido en RUP se encarga de evaluar la calidad del producto en desarrollo, pero no acepta o rechaza al producto final desarrollado, sino que debe ir integrándose en todo el ciclo de vida.

Los estudios realizados y bibliografías consultadas, permitieron conocer que en la Universidad, el proceso de pruebas que se lleva a cabo es el planteado por RUP. Es por eso que se realiza un análisis entre los procesos primarios definidos por la ISO/IEC 12207, para ver la correspondencia con los flujos de trabajo definidos por RUP.

A continuación se proporciona una relación entre ambos, y las actividades de V&V propuestas, demostrando que se pueden realizar estas actividades en ambos casos, con la intención de facilitar y mejorar el desarrollo de las pruebas en La UCI.

Flujos de trabajo de RUP.	**Procesos de la ISO/IEC 12207.**	**Actividades de V&V propuestas.**
Modelo de negocio	Desarrollo	❖ *V&V de la Conceptualización.*
Requerimientos		❖ *V&V de los Requerimientos*
Análisis y Diseño		❖ *V&V del Diseño.*
Implementación		❖ *V&V de la Implementación.*
Pruebas		❖ *V&V de las Pruebas.*
Despliegue	Desarrollo	❖ *Instalación y Chequeo.*
Admón. de cambios y configuración	Operación	❖ *V&V de la Operación.*
Admón. de cambios y configuración	Mantenimiento	❖ *V&V del Mantenimiento.*

Tabla 3: Relación RUP y tareas de V&V.

La V&V de la Conceptualización: es la primera de las actividades propuestas en el manual debido a que en la misma se verifica el diseño de la arquitectura del sistema, la asignación de los requisitos del sistema, se valida la solución seleccionada, y se garantiza que ninguna hipótesis falsa se han incorporado en la solución escogida.

La V&V de los Requerimientos: es la segunda actividad definida, se propone en el manual con la intención de se garantizar la exactitud, integridad, comprobabilidad, y coherencia de los requisitos definidos.

La V&V del Diseño: en esta actividad se demuestra que el diseño es correcto, preciso y completo con respecto a los requerimientos del software y que no han sido introducidas características no definidas al mismo. De ahí la suma importancia de llevar a cabo esta actividad de verificación y validación.

La V&V de la Implementación: es la actividad que verifica y valida que las transformaciones del diseño en el código, estructuras de base de datos, representaciones ejecutables, la codificación del software y las pruebas sean correctas, exactas y completas.

La V&V de las Pruebas: en esta actividad se vela porque los requisitos del software y del sistema sean correctos y correspondan con la ejecución de la integración, sistema, y las pruebas de aceptación.

La V&V Instalación y Chequeo: la intención de esta actividad es verificar y validar la corrección de la instalación de software en el ambiente objetivo.

La V&V de la Operación: es la actividad donde se evalúan las nuevas limitaciones en el sistema, los cambios propuestos y sus efectos en el software, además de los procedimientos de funcionamiento de la exactitud y facilidad de uso.

La V&V del Mantenimiento: se evalúan los cambios planteados y sus efectos en el software, la evaluación de las anomalías que se descubran durante el funcionamiento, así como las necesidades de jubilación, y volver a realizar tareas de V & V.

2.4 Estrategia de Prueba Propuesta.

Una estrategia de pruebas de software integra las técnicas de diseño de casos de pruebas en una serie de pasos bien planificados que llevan a una construcción correcta del software. (Telma R. y Adisleydis O., 2007)

La estrategia de prueba de software describe un mapa de como llevar a cabo las pruebas, como planificarlas, realizarlas, promediar el tiempo de duración, el esfuerzo y la cantidad de recursos requeridos. Siendo necesario que la estrategia sea flexible, para promover la creatividad y adaptabilidad en los grandes sistemas software y lo suficientemente rígida, para poder dar un seguimiento a la planificación. Debe incluir además, pruebas de bajo nivel que verifiquen todos los segmentos de código del programa, y pruebas de alto nivel con la intención de validar las funciones del sistema frente a los requerimientos del cliente. (Pressman, 2005.)

De acuerdo a lo planteado mencionamos algunas características generales que se deben tener en cuanta a la hora de realizar cualquier estrategia, y por las que se ha regido la propuesta de estrategia que se ha definido en el manual:

- Las pruebas se comienzan a nivel de módulo hasta llegar a la integración de todo el sistema.
- Según en el momento que nos encontremos en el ciclo de vida del software se aplican diferentes técnicas de pruebas.
- En grande proyectos las pruebas son llevadas a cabo por un grupo independiente.
- La depuración se debe incluir en toda estrategia de pruebas.

Siguiendo la idea de ir limando los posibles errores e incompatibilidades que puedan ir surgiendo en el camino entre los requisitos especificados y el producto creado, se propone una estrategia, con la intención de proporcionar una mejor aplicación de las actividades de V&V, a lo largo de todo el proceso de desarrollo del software; para un mejor entendimiento de su propósito se decidió realizar el siguiente esquema en el que se muestran los aspectos fundamentales que la componen:

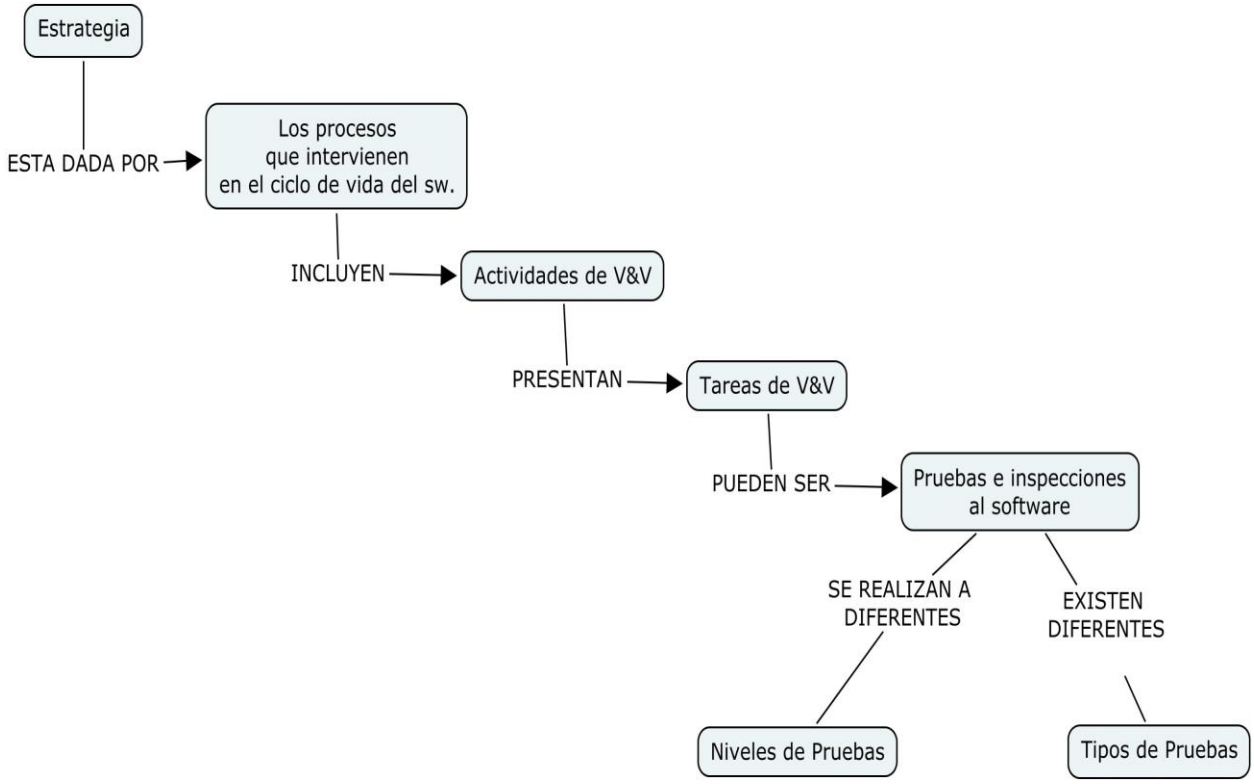


Figura 7: Estrategia de Prueba.

Quedando definida la estrategia propuesta de la siguiente manera:

- V&V de la Conceptualización.
- V&V de los Requerimientos
- V&V de Diseño.
- V&V de la Implementación.
- V&V de las Pruebas.
- V&V Instalación y Chequeo.
- V&V de la Operación.
- V&V del Mantenimiento.

A continuación se describen cada uno de los roles que pueden realizar las actividades de V&V, propuesta en la estrategia:

Probador: es el responsable de entender las reglas del negocio, requerimientos, casos de uso y todos los artefactos que ayuden a comprender el negocio. Debe investigar que determinada parte del programa tiene que llevar a cabo ciertas funciones, y que hacer para encontrar errores a la funcionalidad realizada con anterioridad.

El Diseñador de Pruebas: es el responsable de la integridad del modelo de pruebas, asegurando que este cumpla con su propósito. Además debe planear las pruebas, decidiendo los objetivos de pruebas apropiados y la planificación, seleccionando y describiendo los casos de uso de prueba, así como los procedimientos correspondientes que necesiten. Son responsables además de la evaluación de las pruebas de integración y de sistemas cuando se ejecutan. (No llevan a cabo las pruebas sino que las preparan y las evalúan, otros tres trabajadores son los que aplicaran las pruebas debido a que son los capacitados para hacerlo).

Los Ingeniero de Componentes: son los responsables de de los componentes que automatizan algunos de los procedimientos de prueba (no todos los procedimientos de prueba pueden ser automatizados, porque pueden necesitar de conocimientos de programación).

El Ingeniero de Pruebas de Integración: es el responsable de realizar las pruebas de integración que se necesitan para cada construcción producida en el flujo de trabajo de implementación. Se encarga además de documentar los errores obtenidos como resultados de las pruebas de integración.

El Ingeniero de Pruebas de Sistema: es el responsable de realizar las pruebas de sistemas necesarias, sobre una construcción que muestre el resultado de una iteración completa. Se encarga además de documentar los errores de los resultados de las pruebas de sistema, por estas razones necesitan tener mucho conocimientos del funcionamiento externo del mismo.

El Ingeniero de Pruebas Software: es quien ejecuta el manual de pruebas siendo el responsable de desarrollar ejecutar y mantener el plan de prueba de los módulos dentro de las que se encuentran las pruebas funcionales, de rendimiento, de carga y de estrés. Desarrolla e inspecciona los diseños e implementaciones de cada caso de prueba. Se encarga de la evaluación de los requisitos, casos de uso de aplicación para el diseño, desarrollo de pruebas, casos de prueba y preparar los informes de resultados de pruebas.

2.5 Modelo de Prueba propuesto.

La propuesta de modelo que se realiza en el manual es una modificación realizada sobre el modelo W, debido a que desde el principio de dicho modelo, se representa la iniciación de las actividades de verificación y validación definidas en la estrategia, reflejando así la relación que debe existir entre el ciclo de vida del software y el proceso de prueba; desde que se comienza a desarrollar el producto hasta una vez terminado. Siendo la siguiente figura el modelo que se propone que en conjunto con el manual y la estrategia potenciará la calidad del producto final y el proceso de pruebas que es llevado a cabo en cada elaboración de software.

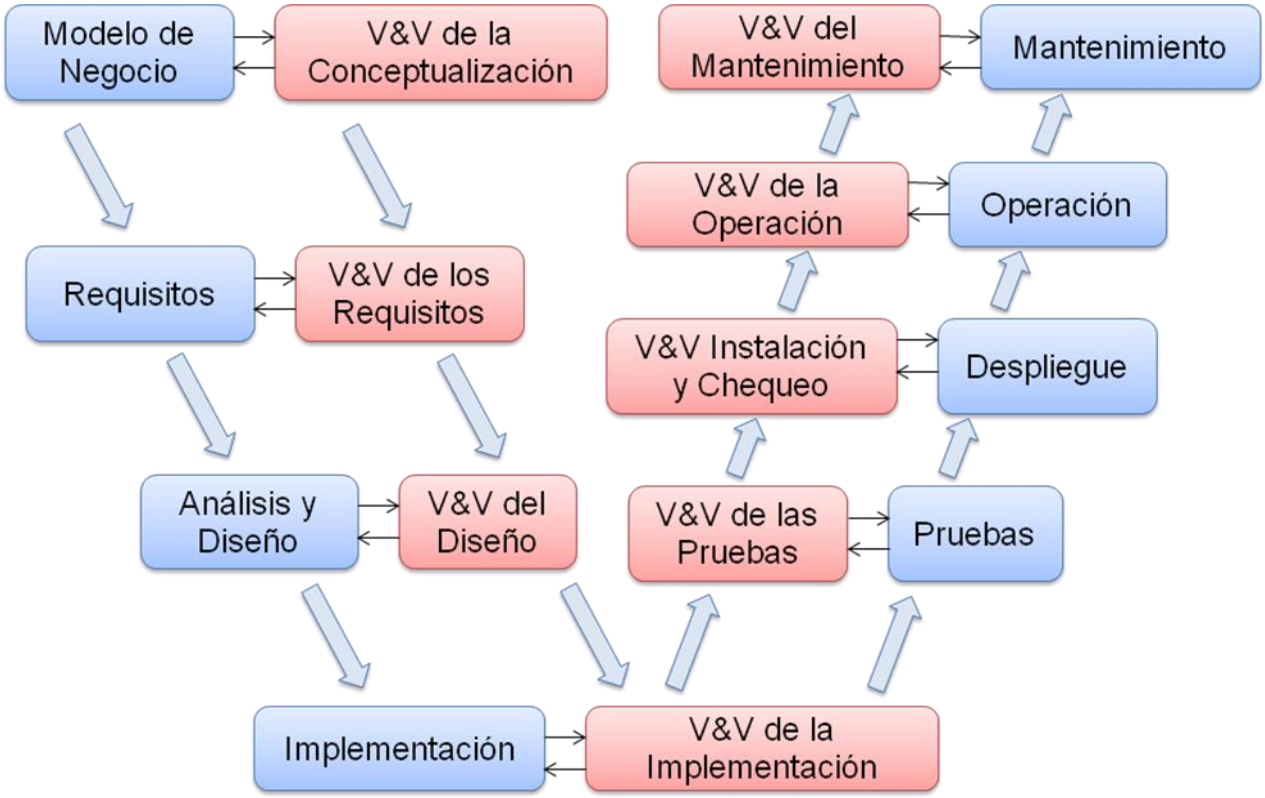


Figura 8: Modelo de Prueba Propuesto.

Otra posibilidad para el ingeniero de pruebas seria aplicar el modelo V sobre el software que este desarrollando. Debido a que puede desarrollar algunas de las actividades de V&V propuestas en el Manual; a continuación se muestra una representación del modelo V

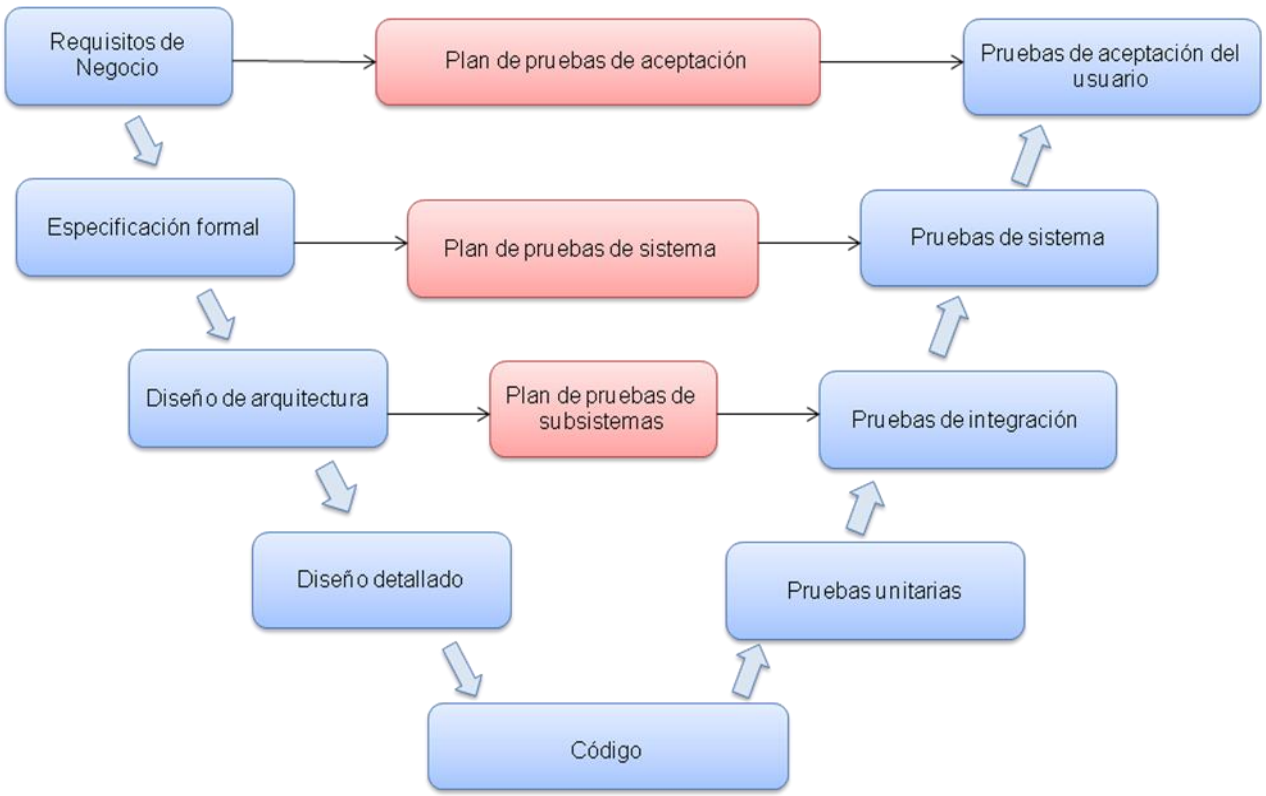


Figura 9: Modelo V.

2.6 Tipos de pruebas propuestas para cada actividad.

Las pruebas deben ocurrir durante todo el ciclo de vida del software, brindando la posibilidad de probar las funcionalidades de los prototipos de interfaz; la estabilidad del software, la cobertura y el rendimiento de la arquitectura; asegurando de esta forma la calidad del producto final.

Definida de una vez, la estrategia a seguir para las actividades de V&V propuestas, se pasará a determinar cuales son los tipos de pruebas a aplicar en cada una de ellas. Para esto se darán a conocer los objetivos de cada uno de los tipos de pruebas elegido.

Las pruebas de unidad: están dirigidas a los diferentes módulos que se desarrollan en cada proyecto. Para lo que se usa la descripción del diseño detallado como guía, se prueban los caminos de control más importantes, con el fin de descubrir errores dentro del ámbito del módulo. Haciendo un uso intensivo de las técnicas de prueba de caja blanca.

Las pruebas de integración: se realizan con el objetivo de tomar los módulos probados en las pruebas de unidad y construir una estructura para el programa, que esté acorde con lo especificado en el diseño. Prevaleciendo en este tipo de prueba las técnicas de caja negra.

Las pruebas de sistemas: Verifican que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total. La prueba del sistema está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora.

Con la intención de probar las funcionalidades del software, es necesario que se realicen un conjunto de pruebas de sistema como:

- *Las Pruebas de Funcionales:* tiene como objetivo verificar la función del sistema al fijar la tención en la validación de las funciones, métodos, servicios y caso de uso. Valida que el sistema cumpla con los requisitos funcionales y no funcionales especificados en el diseño de la solución.
- *Las Pruebas de Rendimiento o Carga:* prueban el rendimiento del software en tiempo de ejecución. Validan y evalúan la aceptabilidad de un elemento de un sistema sobre diferentes cargas de trabajo, mientras el sistema permanece constante. Generalmente se incluye la simulación de cargas de trabajo promedio o pico de desborde que puedan ocurrir dentro de la tolerancia operacional normal.
- *Las Pruebas de Seguridad:* verifican los mecanismos de protección.
- *Las Pruebas de Disponibilidad y Red:* verifican el comportamiento de la aplicación, cambiando la infraestructura de red al aplicar diferentes configuraciones y retardos. Valida que no se reduzca la disponibilidad de los sistemas dentro de la solución.
- *Las Pruebas de Compatibilidad:* Verifican el funcionamiento del sistema sobre diferentes componentes de software. Validando la aplicación con sistemas operativos y navegadores.
- *Las Pruebas de Estrés:* se refieren a cargas extremas, memoria insuficiente, no disponibilidad de servicios y hardware o recursos compartidos limitados. Este tipo de pruebas permite comprender mejor cómo y qué áreas del sistema colapsarán, de este modo es posible planificar contingencias, actualizar el mantenimiento, planear y asignar recursos de antemano.
- *Las Pruebas de Usabilidad:* Verifican la estética, la consistencia de la interfaz de usuario y documentación.
- *Las Pruebas de Fiabilidad:* Verifican la probabilidad de que el sistema funcione o desarrolle una determinada función, bajo condiciones prefijadas y durante un período de tiempo determinado.

- *Las pruebas de configuración:* se enfocan en evaluar las diferentes variaciones de una aplicación integrada, contra sus requerimientos de configuración.
- *Las pruebas de instalación:* se realizan para asegurar el funcionamiento correcto de opciones y funcionalidades de la instalación y para asegurar que todos los componentes necesarios sean realmente instalados.
- *Las pruebas de aceptación:* son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo del producto, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o sobre una versión del producto o una iteración funcional pactada previamente con el cliente. Son muy importantes, ya que definen el paso a nuevas fases del proyecto como el despliegue y mantenimiento.
- *Las Pruebas Alfa:* son llevadas a cabo, por un cliente, en el lugar de desarrollo, usando el software de forma natural con el desarrollador como observador del usuario. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.
- *Las Pruebas Beta:* son realizadas por los usuarios finales del software en los lugares de trabajo, sin el desarrollador. Es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas que encuentra durante las pruebas beta e informa a intervalos regulares al desarrollador.
- *Las 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.
- *Las 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.

Actividades de V&V / **Tipos de pruebas**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**

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						
Pruebas 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		
Prueba de Instalación						X		
Pruebas de aceptación Alfa, Beta						X		
Prueba de Regresión							X	
Depuración								X

Tabla 4: Actividades de V&V con los tipos de prueba.

Leyenda:

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

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

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

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

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

F: Actividad de V&V de Instalación y chequeo(Proceso Desarrollo)

G: Actividad V&V de Operación (Proceso de Operación)

H: Actividad de V&V de Mantenimiento (Proceso de Mantenimiento)

Según los estudios realizados, la norma ISO 9126, define una serie de atributos de calidad, los cuales se encuentran estrechamente relacionadas con los tipos de pruebas. Validando y verificando de esta

forma la calidad del producto desde la hora en punto donde se decidan aplicar los determinados tipos de pruebas. En la siguiente tabla se reflejan los atributos de calidad definidos por la ISO 9126 y la correspondencia de los mismos con los tipos de pruebas propuestos.

Atributos de Calidad	Tipos de Pruebas Propuestos.
Funcionalidad	Funcionales, Seguridad
Usabilidad	Usabilidad
Fiabilidad	Integridad
Rendimiento	Carga
Soporte	Configuración, Instalación

Tabla 5: Atributos de calidad y tipos de prueba.

En la figura que se muestra a continuación se define la relación de el proceso de V&V dentro del proceso de pruebas del software, aclarando cuando y donde utilizar los tipos de pruebas propuestos en el Manual. Los cuales corresponden con los atributos de calidad de la norma ISO 9126, lo cual permite que el software creado alcance determinado nivel de calidad.

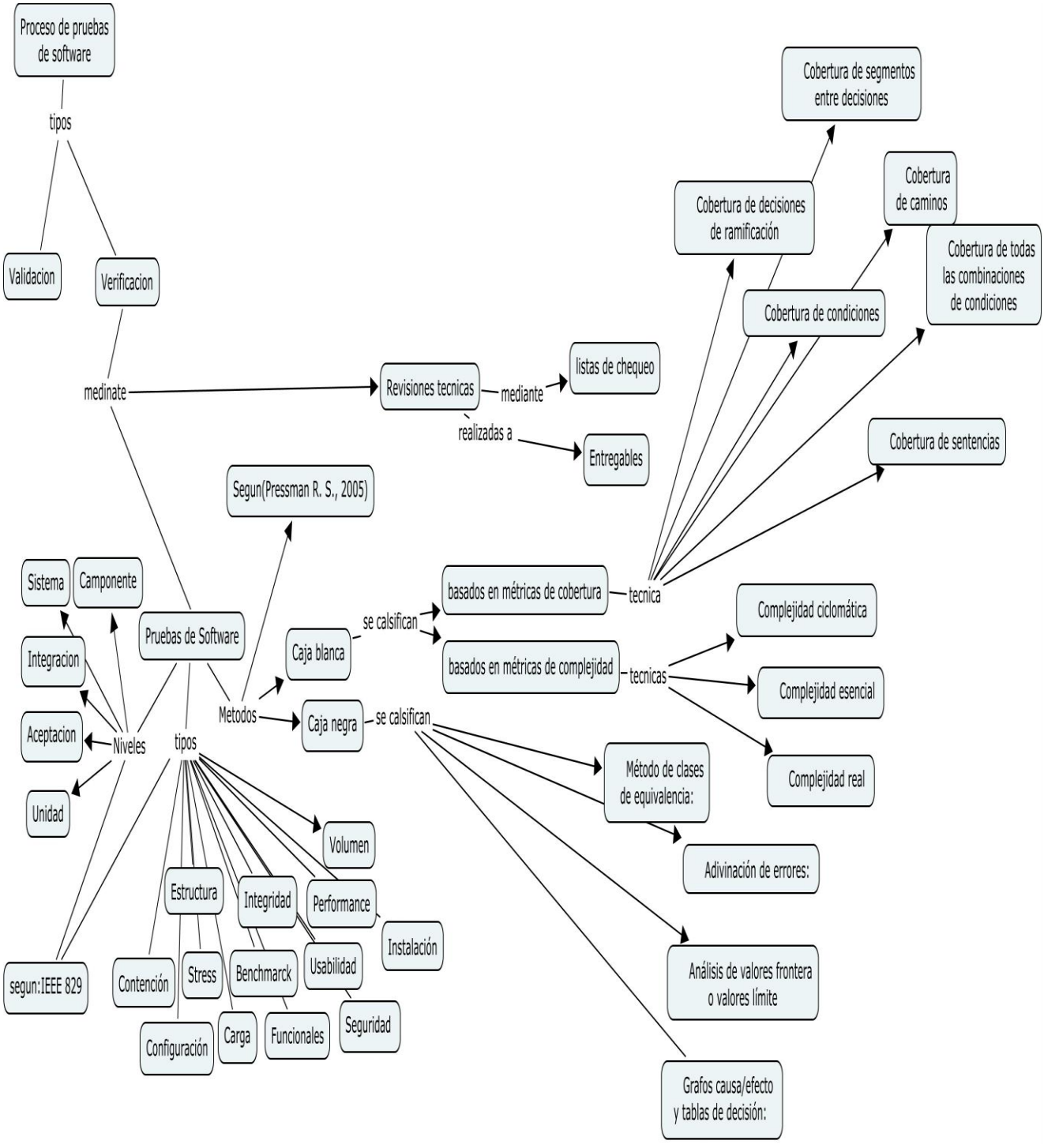


Figura 10: Tipos de prueba propuestos.

2.7 Conclusiones Parciales.

- Las fases definidas para el desarrollo del software en conjunto con las actividades de V&V, permitieron seleccionar como modelo a seguir, una modificación del modelo W.
- La propuesta de tener en cuenta el PSP, como proceso que organiza un uso constante de buenas prácticas de planificación y seguimiento del trabajo de un ingeniero en las etapas de pruebas, permite que este conozca los métodos para mejorar la calidad, organizar y planificar su trabajo a la hora de diseñar y desarrollar las pruebas al software en construcción, estimando el tiempo que demorarán.
- Con la estrategia propuesta para el Manual, se posibilitó definir los diferentes tipos de pruebas, actividades y tareas de V&V a utilizar, para proporcionar resultados satisfactorios en el proceso de pruebas de un producto.

CAPÍTULO III: VALIDACIÓN DEL MANUAL DEL INGENIERO DE PRUEBA.

3. Introducción.

Para aceptar y validad el Manual que se propone, se realizó una selección de expertos que emitieron su criterio, el mismo estuvo compuesto por especialistas en el tema de calidad y pruebas de software de la Universidad de las Ciencias Informáticas. El proceso de validación se llevó a cabo por el Método Delphi, que es una técnica de investigación social que tiene como objetivo, obtener de forma confiable un consenso de opiniones y valoraciones sobre un tema en específico, en este caso es utilizado para validar la calidad del Manual que se propone a utilizar en la UCI.

3.1 Delphi como método de validación de experto.

El nombre Delphi proviene de la antigua Grecia, Delphos fue la localidad donde estuvo el más famoso santuario panhelénico, centrado en el oráculo de Apolo, donde según la leyenda, el oráculo de Apolo manifestaba la voluntad de Zeus a través de una sacerdotisa ('la pitonisa'). Cuyas ambiguas palabras interpretaban los sacerdotes. Este oráculo alcanzó prestigio en los siglos V, VI y VII antes de J.C. El primer estudio Delphi fue realizado en 1950 por la Rand Corporation para la fuerza aérea de EE.UU. y se le dio el nombre de "Proyecto Delphi". El objetivo de este estudio fue obtener el mayor consenso posible, en la opinión de un grupo de expertos por medio de una serie de cuestionarios intensivos, a los cuales se les intercalaba una retroalimentación controlada. (Universidad del Salvador, 2005).

La técnica Delphi se ha convertido en una herramienta fundamental en el área de las proyecciones tecnológicas, incluso en el área de la Administración Clásica y operaciones de investigación.

Linston y Turoff[15] definen la técnica Delphi como un método de estructuración de un proceso de comunicación grupal que es efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema complejo.

En la actualidad, el método Delphi consiste en un proceso iterativo de rondas donde se le realizan encuestas a cada experto de forma individual evitando la interacción. La encuesta se lleva a cabo de una manera anónima (actualmente es habitual realizarla haciendo uso del correo electrónico o

[15] Linstone, H., Turoff, M.: « The Delphi Method. Techniques and Applications », Addison-Wesley, 1975, p.3

mediante cuestionarios web establecidos al efecto) para evitar los efectos de "líderes". (Astigarraga, 2003)

La calidad de los resultados depende, sobre todo, del cuidado que se ponga en la elaboración del cuestionario y en la elección de los expertos consultados. Siempre se comenzaría este proceso enviando un modelo a los posibles expertos con una explicación breve sobre los objetivos del trabajo y los resultados que se desean obtener. Para la aplicación práctica del método es necesario considerar dos cuestiones fundamentales:

- La selección de los expertos.
- La elaboración del cuestionario.

A continuación se explica cómo fueron aplicadas estas cuestiones en el presente trabajo.

3.2 El Proceso de Selección de Expertos.

Como bien se explica anteriormente, es muy importante después de haber decidido aplicar el método Delphi, seleccionar a los especialistas o expertos en el tema a evaluar. Estos, serán una persona o grupo de personas con un gran conocimiento sobre el tema tratado, que podrán emitir criterios irrebatibles de cualquier problema y valoraciones importantes con un alto nivel de conocimiento. (CRITERIO DE EXPERTOS: MÉTODO DELPHY., 2006.)

Para la selección de los expertos que realizarán la valoración del Manual propuesto, se tuvo en cuenta una serie de características, que posibilitaron obtener información curricular y actualizada sobre los posibles expertos. Basándose principalmente, en las participaciones de estos en eventos científicos, que contaran además con una ardua experiencia ocupacional y en el tema de interés. También se tuvo en cuenta los conocimientos que deben tener sobre el problema planteado. Estos conocimientos están basados en las siguientes áreas de trabajo:

- Proceso Unificado de Desarrollo (RUP).
- Proceso de Desarrollo del Software.
- Proceso de Pruebas de Software.
- Validación y Verificación del Software.
- Calidad del Software.
- Norma y Estándares de Calidad.

Basándose en las informaciones anteriores, la lista de expertos a consultar quedó conformada por 6 especialistas. Estos están integrados de manera directa a la producción en la UCI y de una forma u otra vinculados al área de calidad del software y constan con una ardua experiencia en los temas relacionados con el proceso de pruebas.

A este listado de experto, se le realizó una autoevaluación de los niveles de información y argumentación que tienen sobre el tema especificado. En la cual marcaron con una X, en una escala creciente del 1 al 10, el valor que se corresponde con el grado de conocimiento o información que tienen cada uno, sobre el tema de prueba; donde los resultados obtenidos se muestran a continuación en la siguiente tabla, en la cual se destaca que todos cuentan con un nivel de conocimiento mayor o igual a 8:

Expertos	1	2	3	4	5	6	7	8	9	10
1								X		
2										
3								X		
4								X		
5									X	
6								X		

Tabla 6: Grado de conocimiento de los expertos con respecto al tema.

Una vez analizado el grado de conocimiento con el que cuentan estos especialistas, se pasó a calcular el Coeficiente de Conocimiento o Información (Kc), a través de la siguiente fórmula.

Kc = n (0,1)

Donde:

Kc: Coeficiente de Conocimiento o Información.

n: Rango seleccionado por el experto.

En la tabla 7 se muestran los resultados obtenidos para cada especialista, en la que se puede apreciar que todos poseen un coeficiente de conocimiento elevado, superando el 0,8:

Expertos	Valor (Kc)
1	0,8
2	0,8
3	0,8

4	0,9
5	0,8
6	0,8

Tabla 7: Coeficiente de Conocimiento.

Seguidamente se valoran un grupo de aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar, donde se determinan los aspectos de mayor influencia. A partir de lo valores reflejados por cada experto en una tabla similar a esta, en la cual ellos podían marcar con una X el nivel que ellos tienen en estos aspectos. Posteriormente se contrastan con los valores de la siguiente tabla patrón:

Fuentes de argumentación o fundamentación	Alto	Medio	Bajo
Análisis teóricos realizados por usted	0.3	0.2	0.1
Su experiencia obtenida	0.5	0.4	0.2
Trabajos de autores nacionales	0.05	0.05	0.05
Trabajos de autores extranjeros	0.05	0.05	0.05
Su conocimiento del estado del problema en el extranjero	0.05	0.05	0.05
Su intuición	0.05	0.05	0.05

Tabla 8: Nivel de argumentación (tabla patrón).

Estos aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar, permitió calcular el Coeficiente de Argumentación (Ka) de cada experto, mediante la siguiente ecuación:

Ka =a ni = (n1 + n2 + n3 + n4 + n5 + n6)

Donde:

Ka: Coeficiente de Argumentación.

ni: Valor correspondiente a la fuente de argumentación i (1 hasta 6)

Los resultados obtenidos fueron que 3 de los especialistas poseen un nivel de coeficiente de argumentación igual a 1, y el resto poseen un nivel mayor o igual a 0.8.

Expertos	Valor (Ka)
1	0,9

2	0,8
3	1
4	1
5	1
6	0,9

Tabla 9: Coeficiente de Argumentación.

Una vez obtenido los valores del Coeficiente de Conocimiento (Kc) y el Coeficiente de Argumentación (Ka) se procede a obtener el valor del Coeficiente de Competencia (K) que finalmente es el coeficiente que determina en realidad que experto se toma en consideración para trabajar en esta investigación Este coeficiente (K) se calcula de la siguiente forma:

K = 0,5 (Kc + Ka)

Donde:

K: Coeficiente de Competencia

Kc: Coeficiente de Conocimiento

Ka: Coeficiente de Argumentación

Posteriormente obtenido los resultados se valoran de la manera siguiente:

0,8 < K < 1,0 Coeficiente de Competencia Alto

0,5 < K < 0,8 Coeficiente de Competencia Medio

K < 0,5 Coeficiente de Competencia Bajo

Expertos	Valor (K)	Alto	Medio	Bajo
1	0.85	X		
2	0.9	X		
3	0.9	X		
4	0.95	X		
5	0.85	X		
6	0.85	X		

Tabla 10: Coeficiente de Competencia.

En la tabla 10, se muestran los valores del coeficiente de competencia del listado de expertos, obteniendo resultados de Competencia Alto en cada uno de ellos, manteniéndose los 6 para la

validación de la propuesta. De ellos cuatro son especialistas de la Dirección de Calidad de software en la Universidad de la Ciencias Informáticas, uno es asesor de la calidad de la facultad en donde radica y el otro es profesor de Gestión de Software y BD. A continuación se presentan las fichas con los datos de cada experto seleccionado, manteniendo bajo confidencialidad la identidad de los mismos.

Fichas de los expertos.

Experto 1
Graduado de: Ingeniería en Ciencias Informáticas.
Años vinculados en la UCI: Recién graduado de la UCI.
Ocupación: Especialista de Calidad de Software.
Eventos científicos: • Fórum de ciencia y técnica • Séptima Semana Tecnológica • Talleres de informática de calidad 2003, • Cursos de calidad • Diplomados de V&V
Experiencia en el tema de pruebas: 3 años como Jefe de prueba en la liberación de software para Venezuela.

Experto 2
Graduado de: Ingeniería en Ciencias Informáticas.
Años vinculados en la UCI: Recién graduado de la UCI.
Ocupación: Asegurador de la calidad, Proyecto Mapeo Cerebral Humano.
Eventos científicos: • Participación en las Jornadas iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento 2008. • Una publicación internacional. • Diplomado de calidad • Diplomado de V&V
Experiencia en el tema de pruebas: 3 años como Jefe de prueba en la liberación de software para Venezuela.

Experto 3

Graduado de: Ingeniería en Ciencias Informáticas.

Años vinculados en la UCI: Recién graduada de la UCI.

Ocupación: Profesor Gestión de SW y BD facultad 6.

Eventos científicos:

- JIISIC Jornadas iberoamericanas de ingeniería de software e ingeniería del conocimiento, Guayaquil, Ecuador.
- 7 Semana Tecnológica, Fordes. La Habana, Cuba.

Experiencia en el tema de pruebas: Participación en ambos eventos con investigaciones relacionada con pruebas de software. Tres años trabajando con el tema de pruebas primeramente en el grupo de investigación de la dirección de calidad de la UCI, el laboratorio central de calidad de la UCI y como asegurador de calidad software en un proyecto.

Experto 4

Graduado de: Ingeniería en Ciencias Informáticas.

Años vinculados en la UCI: Recién Graduado4 años vinculados a la Dirección de Calidad.

Ocupación: Especialista de Calidad de Software.

Eventos científicos:

- Jornada Iberoamericana de la Ingeniería de Software y el Conocimiento
- 7ma Semana Tecnológica.
- Forma parte del comité organizador de UCIENCIA 2008
- Curso de técnicas avanzadas en desarrollo de software
- Está certificado en CMMI por el ESI Center de Monterrey México.
- Métricas de Software.
- Ha impartido todos los cursos del segundo perfil de calidad en la UCI Diplomados de V&V.
- Ha participado en pruebas de aceptación de productos en Venezuela y España.

Experiencia en el tema de pruebas: 3 Años de Jefe de Prueba. 1 Año de Especialista del Laboratorio de Certificación.

Experto 5

Graduado de: Ingeniería en Ciencias Informáticas.

Años vinculados en la UCI: Recién graduada de la UCI.

Ocupación: Especialista de la Dirección de Calidad de software.

Eventos científicos:

- Evento UCIENCIA
- Cursos del diplomado de calidad
- Cursos de Técnicas avanzadas de desarrollo de software
- Diplomado Métricas
- Diplomado Validación y Verificación
- Diplomado MDA
- Curso Gestión de proyecto.

Experiencia en el tema de pruebas: 3 años trabajando con el tema de pruebas primeramente en el grupo de investigación de la dirección de calidad de la UCI, el laboratorio central de calidad de la UCI. 1 Año de Especialista del Laboratorio de Certificación.

Experto 6

Graduado de: Ingeniería en Ciencias Informáticas.

Años vinculados en la UCI: 3 años.

Ocupación: Especialista General de la Dirección de Calidad.

Eventos científicos:

- Cuatro Publicaciones en UCIENCIA sobre el flujo de pruebas.
- Publicación en la revista cubana de informática
- Curso Técnicas avanzadas para el desarrollo de software impartida por el Dr. Juan Carlos Granja.

Experiencia en el tema de pruebas: 3 años trabajando con el tema de pruebas primeramente en el grupo de investigación de la dirección de calidad de la UCI, el laboratorio central de calidad de la UCI. 1 Año de Especialista del Laboratorio de Certificación.

Una vez confeccionado el listado de expertos, se invitó a cada uno de ello de forma personal para que participara en el proceso de Validación y aceptación de la propuesta. Se detalló de forma clara y

precisa los objetivos de la propuesta, así como la realización del cuestionario. Con este paso se termina el proceso de selección del personal que conformará el Panel de Experto.

Una vez dado el consentimiento de cada uno de los expertos se conforma el panel que no es mas conjunto de expertos que toma parte en el método Delphi.

3.3 Elaboración del Cuestionario.

El cuestionario es el documento que se envía a los expertos. No es sólo un documento que contiene una lista de preguntas, sino que es el documento con el que se consigue que los expertos interactúen, ya que en él se obtendrá la información para presentar los resultados.

Para la presentación del cuestionario se tuvieron en cuenta preguntas con enfoques investigativos y que se centran principalmente en los objetivos que debe cumplir la propuesta presentada, además de permitir que las respuestas fueran abiertas y en todos los casos con posibilidad de emitir su criterio personal, en cuanto a la propuesta que se presenta para el proceso de pruebas de la UCI, esta encuesta se puede apreciar en el (Anexo1: Cuestionario a Miembros del Panel de Expertos.).

En este proceso los Expertos que conforman el panel recibieron de forma personal un resumen del Manual del Ingeniero de Pruebas, unido al cuestionario que debían responder, en un mínimo de tiempo para analizar las respuestas y pedirles que realizaran preguntas en caso de que surgieran dudas durante el estudio de la misma.

3.4 Validación de la propuesta.

Para validad el Manual del Ingeniero de Prueba propuesto, se tuvieron en cuenta una serie de objetivos, que son reflejados en el cuestionario a través de las preguntas aplicadas en el mismo, la correspondencia de los objetivos con cada pregunta se muestra en la siguiente tabla:

Preguntas / **Objetivos**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Importancia de la propuesta.	X							
Satisfacción de las necesidades de los ingenieros de prueba.		X						
Necesidad de empleo de la propuesta.		X						
Posibilidad de aplicación.		X	X					
Adaptabilidad a proyectos de la propuesta.		X						
Aporte de la propuesta al proceso de pruebas en la UCI.		X						

Eficacia de la propuesta.						X		
Completitud de la propuesta.					X		X	
Recomendaciones.				X				X

Tabla 11: Objetivos por Preguntas.

El cuestionario presenta 8 preguntas, la primera responde al objetivo numero uno importancia de la propuesta, la segunda tributa a la satisfacción de las necesidades de los ingenieros de prueba, la necesidad de empleo de la propuesta, su posibilidad de aplicación, adaptabilidad a los proyectos y el aporte al proceso de pruebas en La UCI. En la pregunta número tres se retoma el objetivo de la posibilidad de aplicación, debido a que constituye uno de los principales para la investigación realizada. Determinándose la completitud del Manual del Ingeniero de Prueba, mediante las preguntas 5 y 7. Además los criterios emitidos en la pregunta 6 permiten determinar la eficacia de la propuesta realizada. Mientras que las preguntas 4 y 8 son opiniones y criterios que pueden tener los expertos de aporte al Manual evaluado. Cubriéndose así en cada una de las preguntas del cuestionario los objetivos trazados.

3.4.1 Importancia de la propuesta.

Para valorar este aspecto se tuvo en cuenta las respuestas de los expertos a la pregunta 1, referidos a la importancia de desarrollar un Manual para los ingenieros de prueba, para facilitar el desarrollo del proceso de pruebas en la Universidad de las Ciencias Informáticas. Las respuestas podían ser positivas, negativas o no se emite criterio, para mostrar los resultados estadísticamente se evalúan las preguntas en un rango de 1-3, se le asigna a las respuestas positivas un valor de 3, 1 a las negativas y 2 definido para la restante opción. El resultado se muestra a continuación:

Figura 11: Importancia de la propuesta.

Como se observa en la grafica todos los expertos coincidieron en su totalidad que el Manual propuesto tiene gran importancia para el proceso de pruebas en la Universidad. Argumentando su respuesta a través de los criterios que se emiten a continuación:

- Con el mismo se contará con una herramienta para los probadores.
- De esta forma las personas que desarrollan las pruebas tendrán una guía para la ejecución de las mismas fundamentalmente cuando no tienen casi experiencia por ser en su mayoría estudiantes.
- El equipo de prueba tendrá una guía de cómo llevar a cabo las pruebas de software, siguiendo una norma internacional e interrelacionándolo con la propuesta de RUP para la realización de este tipo de trabajos.
- En la universidad se cuenta con muy poca bibliografía y la que está disponible en internet es en ingles, por lo que el Manual propuesto mejoraría el acercamiento de las principales actividades que puede realizar el ingeniero de prueba.
- Esto ayudaría en gran medida a los que estén desempeñando este rol ya que tendría una serie de pasos y actividades a realizar para su trabajo. Además de agilizar el proceso de las pruebas.

- La estandarización y oficialización de cualquier proceso es un elemento importante, en nuestra universidad existen deficiencias en el proceso de pruebas, lograr describir un proceso y crear un Manual de cómo se pueden hacer las cosas, sencillamente sería una excelente solución.

3.4.2 Satisfacción de las necesidades de los ingenieros de prueba.

En la pregunta 2 del cuestionario, los expertos dieron una valoración en un rango del 1 al 5 siendo el 5 el mayor valor, del grado de satisfacción que proporciona el Manual a las necesidades de los ingenieros de prueba. El resultado se muestra a continuación:

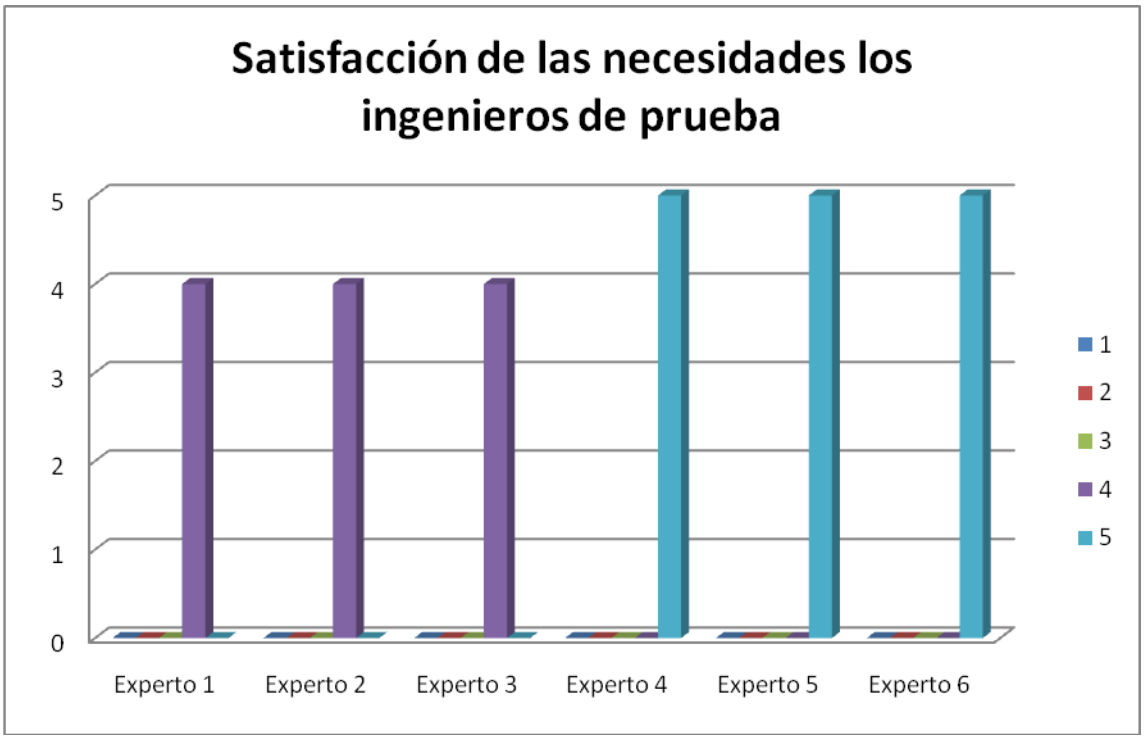


Figura 12: Satisfacción de las necesidades de los ingenieros de prueba.

En su totalidad los expertos concuerdan de manera general que el Manual puede satisfacer las necesidades de los ingenieros a la hora de realizar las pruebas de software, 3 expertos dan una puntación de 4 y los otros de 5 puntos, sin mostrar muchas divergencias en los criterios emitidos.

3.4.3 Necesidad de empleo de la propuesta.

Este objetivo se evalúa de la misma manera que el anterior, los expertos dieron una valoración en un rango del 1 al 5 siendo el 5 el mayor valor, al grado de necesidad que existe en la Universidad de emplearse esta propuesta. El resultado se muestra a continuación:

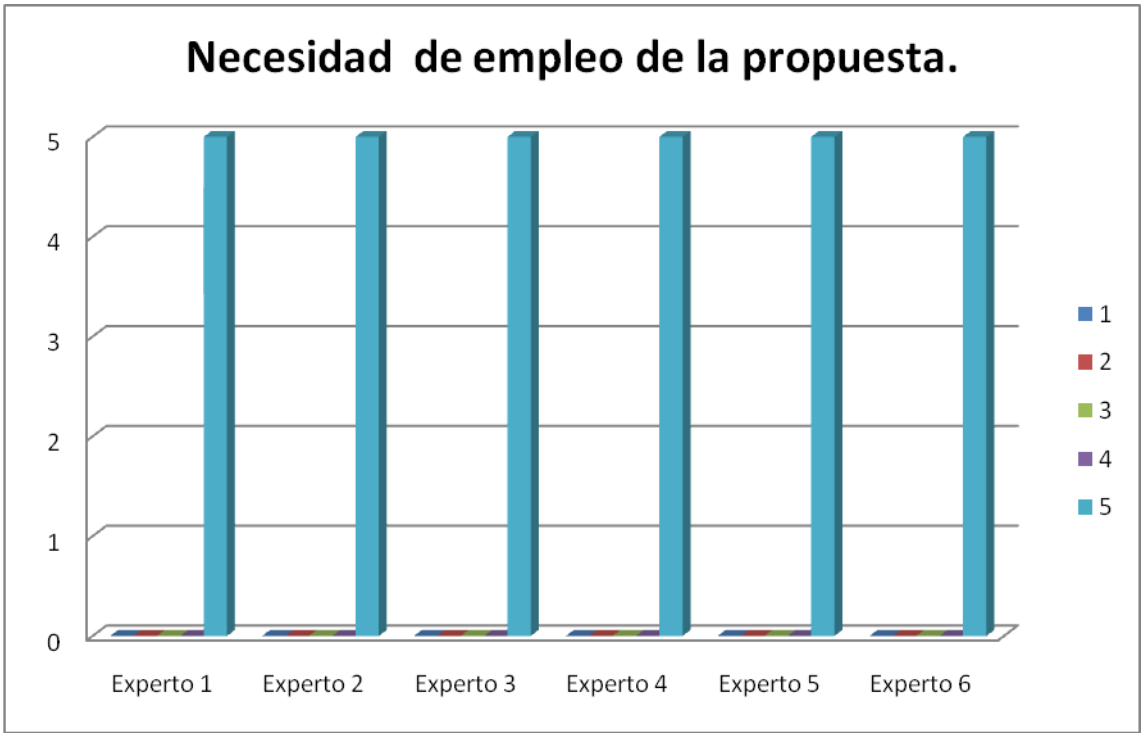


Figura 13: Necesidad de empleo de la propuesta.

En su totalidad los expertos concuerdan que es necesario utilizar el Manual dándole a este objetivo su mayor puntuación.

3.4.4 Posibilidad de aplicación.

En este objetivo los expertos dieron una valoración en un rango del 1 al 5 siendo el 5 el mayor valor. El resultado se muestra a continuación:

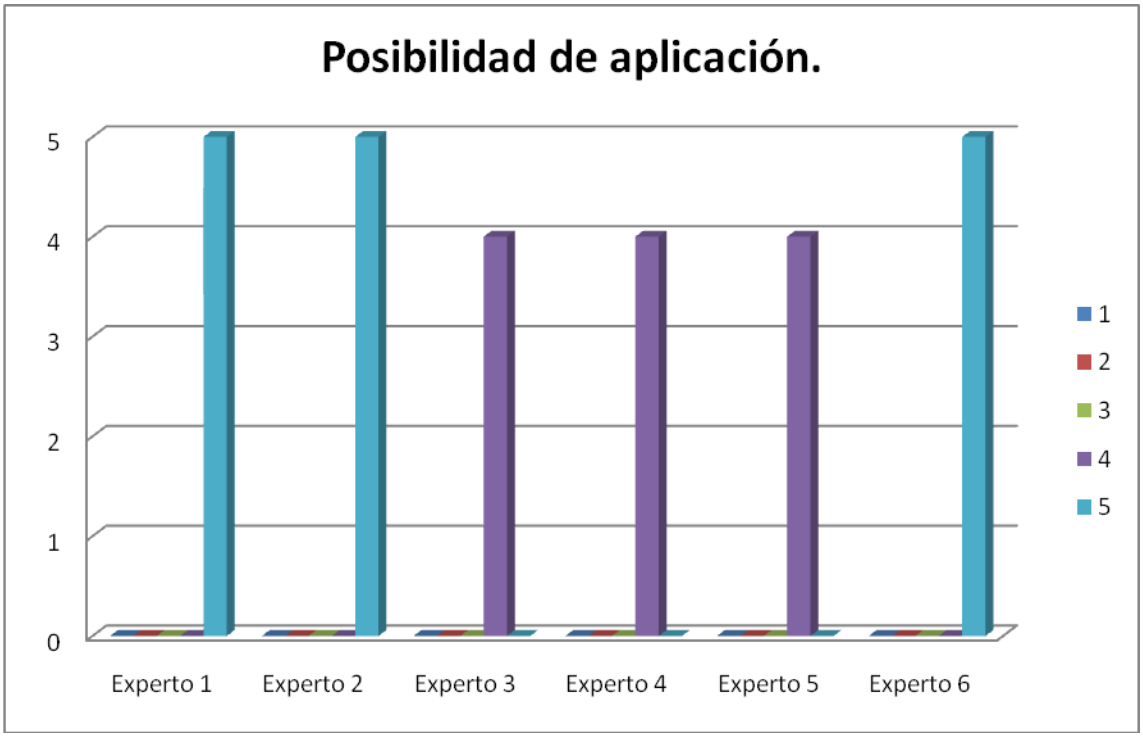


Figura 14: Posibilidad de aplicación.

Los expertos consideraron que el Manual tiene una gran posibilidad de ser aplicado, dando el 50% de ellos una evaluación de 4 y el resto de 5 puntos a este objetivo. De manera general se coincide que no existe gran diferencia de criterio por parte de los expertos. A continuación se citan algunos criterios emitidos por los mismos:

- Me parece muy interesante la idea pero primeramente no se debe hacer rechazo a la cultura de la calidad, aspecto muy común en nuestra universidad, su implementación en la universidad traerá consigo una mayor preparación de los profesionales y estudiantes vinculados a temas relacionados con las pruebas del software.
- El Manual propuesto podría aplicarse de manera exitosa en los proyectos de nuestra universidad, ya que recoge claramente las actividades a desarrollar por un ingeniero de prueba. Cumpliendo con todo lo que se propone en el Manual los productos de la UCI podrían salir con mayor calidad. Además se adecua al proceso de desarrollo que tenga el proyecto.
- Se puede aplicar porque el procedimiento que encierra es genérico y puede ser usado desde cualquier nivel del grupo de calidad interna hasta el laboratorio central.
- Se puede aplicar porque no hay una guía para el desarrollo de las pruebas en los proyectos.

- Se puede aplica porque todas las facultades se realizan pruebas al software y realmente les sería de gran ayuda un Manual para guiarse.

3.4.5 Adaptabilidad a proyectos productivos.

En este objetivo los expertos dieron una valoración en un rango del 1 al 5 siendo el 5 el mayor valor. El resultado se muestra a continuación:

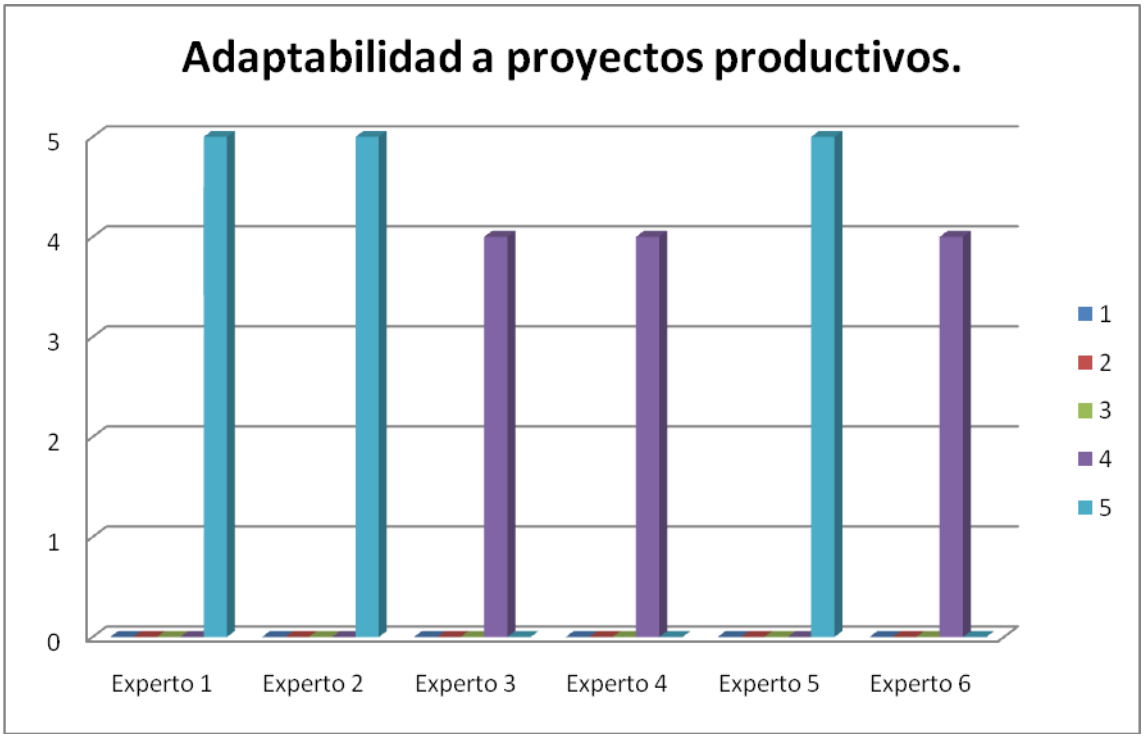


Figura 15: Adaptabilidad a proyectos productivos.

Los expertos coinciden de manera general en que el Manual se puede adaptar a proyectos productivos de la Universidad; para un 50% de la evaluación se obtuvo una puntación de 4 y el 50% restante proporcionó 5 puntos, sin mostrar grandes divergencias en los criterios emitidos.

3.4.6 Aporte al proceso de pruebas implementado en la UCI.

En este objetivo los expertos dieron una valoración en un rango del 1 al 5 siendo el 5 el mayor valor. El resultado se muestra a continuación:

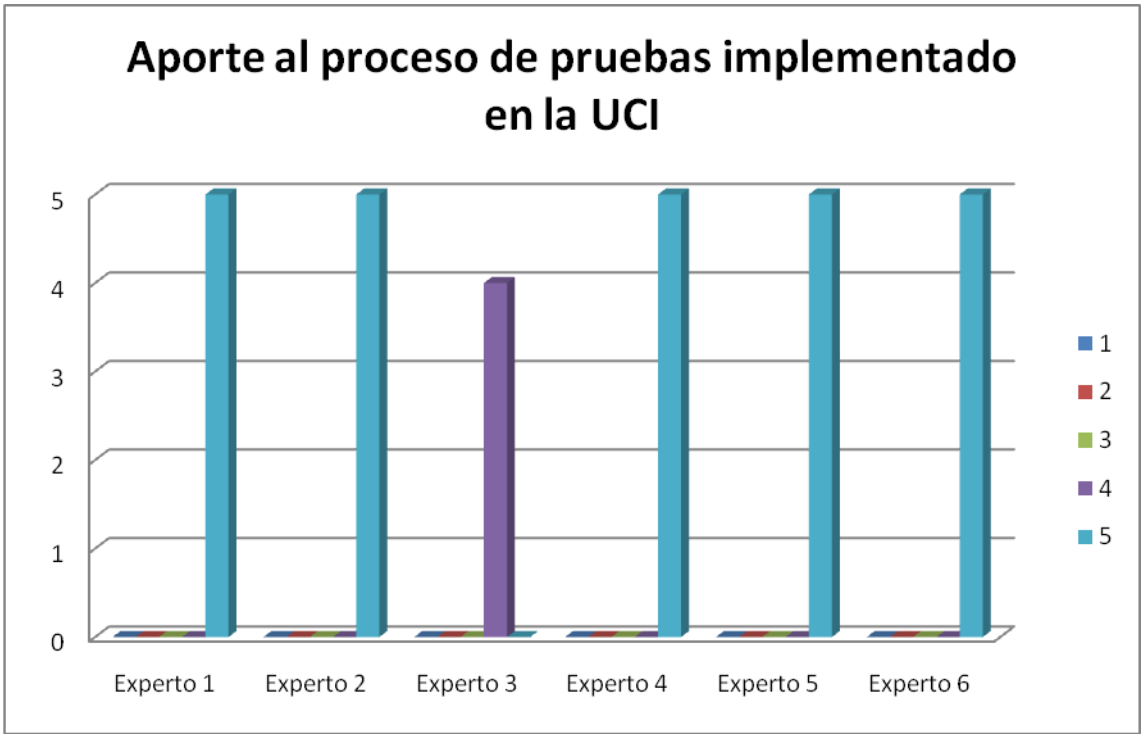


Figura 16: Aporte al proceso de pruebas implementado en la UCI.

El 80% de los expertos coinciden en que la propuesta serviría de aporte al proceso de prueba implementado en la UCI.

3.4.7 Eficacia de la propuesta.

Para valorar este aspecto se tuvo en cuenta las respuestas de los expertos a la pregunta 6, referidos a la eficacia del Manual. Las respuestas podían ser No sería eficaz, Pudiera serlo o no serlo y Sería eficaz, para mostrar los resultados estadísticamente se evalúan las preguntas en un rango de 1-3, se le asigna a las respuestas positivas un valor de 3, 1 a las negativas y 2 definido para la restante opción. El resultado se muestra a continuación:

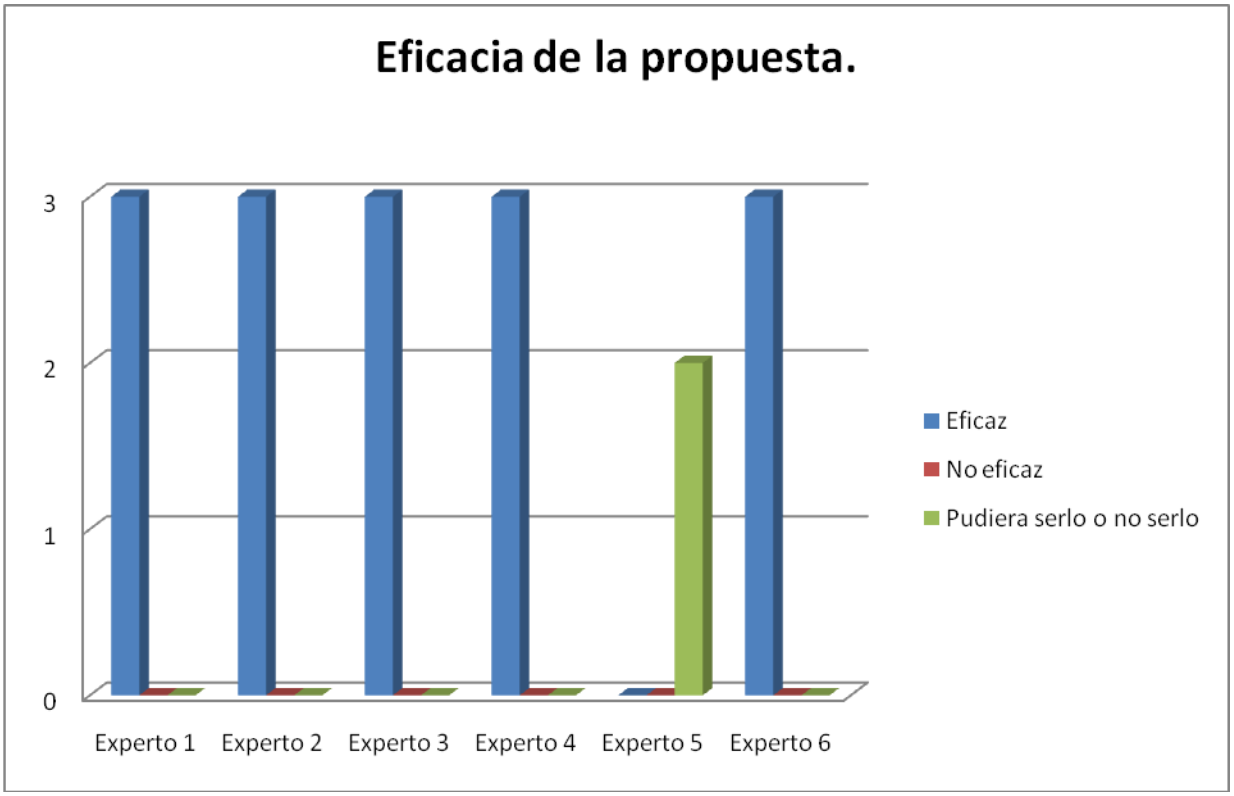


Figura17: Eficacia de la propuesta.

En este objetivo 4 de los expertos estuvieron de acuerdo con que el Manual propuesto seria eficaz, mientras que solo 1, no fue capaz de emitir un criterio certero de la eficacia que proporcionaría el mismo.

3.4.8 Completitud de la propuesta.

Para evaluar las respuestas emitidas por los expertos, las cuales podían ser, Si, No se y No, en este objetivo le fueron asignados a cada una de ellas un valor en un rango de 1- 3 donde tres era el mayor de los valores y coincidiendo con las mismas respectivamente.

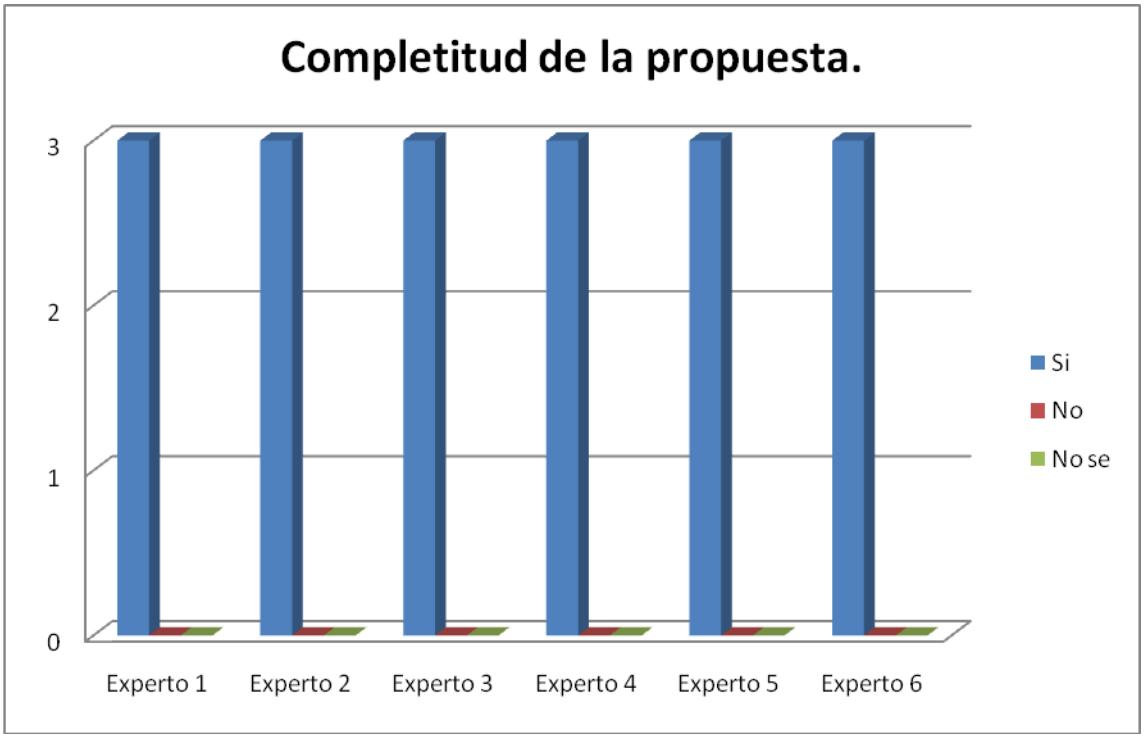


Figura18: Completitud de la propuesta.

El 100% de los expertos estuvieron de acuerdo con que el Manual propuesto, es bastante completo, por lo que no se evidencia ninguna divergencia de criterio entre los mismos.

3.4.9 Recomendaciones.

Por último en el punto número cuatro del cuestionario, se da la posibilidad a cada uno de los expertos de especificar los factores que creen que pueden estar a favor de la aplicación del Manual del Ingeniero de Pruebas y en contra del mismo. Mostrándose las siguientes opiniones:

En contra:

- Rechazo de las personas con respecto al tema de calidad, sobre todo al tema relacionado con las pruebas.
- Cronogramas retrasados, se sacrifica el tiempo de pruebas.
- El equipo de desarrollo le resta importancia a las pruebas, dedicándole un menor tiempo del que realmente se necesita, por lo que tiende a violar pasos de la propuesta.
- Ataría la resistencia al cambio.

- Como factor que podría atentar tenemos los apretados cronogramas de la mayorías de los proyectos, además el cambio cultural sobre la calidad de software que se está desarrollando en la uci, muy nuevos estos temas, y no aceptados por todos.
- Al tener que no estar abalado por el jefe de proyecto el Manual propuesto, se dificultaría su aplicación.
- Lo afecta el grado de conocimiento de los probadores.

A favor:

- La dirección de la producción esta identificada con la calidad y la exigencia es cada vez mayor.
- Favorecería la necesidad de la universidad.
- Y los principales factores que podrían facilitarlo esta que tenemos profesionales muy bien preparados en estos temas, y que muchos de los estudiantes egresados y que culminaran sus estudios este curso 2007-2008, están formados con esta cultura de la calidad.
- Este man0ual tendría que estar abalado por el jefe de proyecto para que se incluya el tiempo de las pruebas en la panificación inicial del proyecto.
- Grupos de calidad de cada facultad.

Los resultados obtenidos en la validación realizada por cada Experto al Manual propuesto, dieron resultados positivos en cada uno de los objetivos planteados y no hubo diversidad de criterios entre los expertos que formaron el panel, por lo que no fue necesaria la aplicación de otro cuestionario para validar la propuesta. A continuación se muestra una figura con porciento de concordancia de las respuestas de los expertos por objetivos.

Figura 19: Resultados finales de la validación.

Algunas de las opiniones aportadas por los expertos en el punto numero 8 del cuestionario:

- Pensar como realizar las pruebas de software a productos basados en arquitectura MDA.
- Seria muy interesante su aplicación práctica.
- De forma general el Manual esta muy bien explicado y tiene buen contenido referente a las diferentes pruebas en el software, pudiera tener un componente. Donde se pudiera poner de manifiesto su aplicación práctica del mismo.
- En general el Manual esta bastante completo ya que contiene un epígrafe explicando como usarlo, esto serviría para los ingenieros de pruebas que desempeñan este rol por primera vez. Otro tema importante del Manual que se propone son las pruebas en todo el proceso de desarrollo del software, esto ayuda en gran medida que el producto final salga con más calidad.
- El Manual debe ir evolucionando a medida que se perfeccionen los procedimientos de calidad en la Universidad de Ciencias Informáticas.

3.5 Conclusiones Parciales.

- El análisis de los resultados anteriores permite afirmar de modo general que el Manual del Ingeniero de Pruebas, fue evaluado por los expertos como efectivo, correcto y necesario para desarrollar en el proceso de pruebas de los proyecto de la Universidad de Ciencias informáticas.
- Las recomendaciones estuvieron alrededor de pensar como realizar las pruebas de software a productos basados en arquitectura MDA y realizar las mejoras necesarias según se vaya perfeccionando los procedimientos de calidad en la UCI. Además seria muy interesante que se pusiera en práctica lo más pronto posible.

CONCLUSIONES.

Al culminar esta Investigación se ha llegado a las siguientes Conclusiones Generales:

- El empleo de la norma ISO 12207 unido al estándar IEEE 1012 ha sido conveniente y productivo, porque logró definir primeramente las fases del ciclo de vida del software y para cada una de ellas se introdujeron las actividades de verificación y validación del software como guía al proceso personal de pruebas llevado a cabo en La UCI.
- Seleccionar el modelo W como base para la estrategia, garantizó las pautas para obtener un modelo sobre modificaciones del anterior, que propone la relación entre el ciclo de vida del software y las actividades de V&V definidas.
- La utilización del PSP, como proceso que organiza un uso constante de buenas prácticas de planificación y seguimiento del trabajo de un ingeniero en las etapas de pruebas, permitió que el ingeniero de pruebas conozca los métodos para mejorar la calidad, organizar y planificar su trabajo a la hora de diseñar y desarrollar las pruebas al software en construcción.
- Mediante la Estrategia propuesta dentro del Manual, se organizó los diferentes tipos de pruebas, actividades y tareas de V&V a utilizar en el proceso personal de pruebas de un producto.
- Las herramientas de la suite de Rational para la administración de las pruebas, Rational test manager, Rational robot, constituyen una propuesta dentro de dicho Manual como instrumento para automatizar el proceso personal de pruebas en La UCI.
- Con la obtención del Manual del Ingeniero de Prueba, quedó plasmado un comportamiento organizado y con calidad, en las actividades a realizar por el ingeniero de pruebas, lo que contribuirá a la calidad del desarrollo del proceso personal de pruebas de La UCI.

RECOMENDACIONES.

En aras de lograr una adecuada puesta en práctica y la obtención de buenos resultados en el proceso de pruebas llevado a cabo en la UCI se recomienda:

- Realizar nuevamente un estudio minucioso sobre el proceso de pruebas a lo largo del desarrollo del software.
- Valorar la necesidad de la incorporación de nuevas actividades a la estrategia propuesta.
- Analizar como realizar el proceso de pruebas con respecto a productos desarrollados bajo arquitectura SOA y MDA.
- La automatización del Manual del Ingeniero de Pruebas.

BIBLIOGRAFÍA.

1. 1008-1987. *Standard for Software Unit Testing.* New York: The Institute of Electrical and Electronics Engineers,.
2. 12207-0-1996. *Software life cycle processes (86 págs).pdf.*
3. 12207-2-1997, 1. *Software life cycle processes—Implementation considerations (110 págs).pdf .*
4. 610-12-1990, I. *Standard Glossary of Software Engineering Terminology (93 págs).pdf.*
5. 9000-3, I. *Laboratorio de Sistemas de Información Facultad de Informática - Universidad Politécnica de Valencia.* Valencia.
6. Astigarraga, E. (2003). *Método Delphi.Universidad de Deusto.* E-20.080 Donostia - San Sebastian.
7. BIBLIOGRAPHY España, M. V. (s.f.). Prueba de Software y Seguridad en entornos distribuidos .
8. Expo:QA. (2008). *Jornadas Profecionales de Calidad y Pruebas de software*. (Madrid, España) Recuperado el 23 de febrero de 2008, de http://www.expoqa.com/
9. G.Myers. (2005). *"Fase de Elaboración. FT Prueba (procedimientos genéricos y aplicación de algunos tipos de pruebas simples).".*
10. Glossary of Testing Terms. (2005). *Standard glossary of terms used in Software Testing.Version 1.1 (dd. September, 29th 2005).* Editor : Erik van Veenendaal (The Netherlands).
11. GreenSQA. (2008). *GreenSQA Software Quality Assurance*. (ParqueSoft) Recuperado el 23 de febrero de 2008, de http://greensqa.com/
12. Ian Sommerville. (2005). *Ingeniería de software. Séptima edición,p. 712.* Madrid: PEARSON EDUCACIÓN, S.A.,.
13. IEEE 1012-1986. *Estandar IEEE para la Planificacion de Validación y Verificación de Software.*
14. IEEE 1012-1998. *Standard for Software Verification and Validation.* Sponsored by the Software Engineering Standards Committee.
15. IEEE 1074-1997. *Standard for Developing Software Life Cycle Processes (96 págs).*
16. IEEE 829-1998. *Standard for Software Test Documentation. 59 pág.* New York, NY: Software Engineering Technical Committee.
17. IEEE, S. (2004). Guide to the Software Engineering Body of Knowledge.

BIBLIOGRAFÍA.

18. ISO 9000-3. (s.f.). *Normas de gestión de la calidad y garantía de la calidad,*. Recuperado el marzo de 2008, de http://campus.fortunecity.com/defiant/114/iso9000.htm
19. ISO 9001:2000. *Sistemas de gestión de la calidad.* Ginebra, Suiza,: Impreso en la Secretaría Central de ISO.
20. ISO/IEC 12207.1-1997. *Software life cycle processes—Life cycle data (37 págs),*. New York.: The Institute of Electrical and Electronics Engineers,.
21. ISO/IEC TR 15504-1. (1998.). *Information Technology – Software Process Assesment – Part 1: Concepts and Introductory Guide.*
22. JTS. (2008). *JTS 2008: Jornadas sobre el Testeo de Software*. (Valencia,España) Recuperado el 23 de febrero de 2008, de http://www.iti.upv.es/groups/squac/events/JTS2008
23. Leticia Davila N. (Diciembre del 2003). *Evaluacion de la Calidad en Sistemas deInformacion en Internet.* Mexico D.F.
24. M.B.CH.M.K.S.SHRUMI. (2003.). *CMMI Guidelines for Process Integration and Product.* Addison Wesley,.
25. Michaell G.J. (2007.). *PROCESO DE PRUEBAS PARA LA LIBERACIÓN DE PRODUCTOS SOFTWARE.* Ciudad Habana.
26. Miguel Ángel M.A. (11-12 de mayo de 2006). *Técnicas de Mejora de la Calidad y la Productividad del Software en el Ámbito de la Ingeniería de Requisitos y la V&V de Modelos.* Recuperado el 14 de marzo de 2008, de http://alarcos.inf-cr.uclm.es/calipso/data/presentacionesCR2006/NodoUM.pdf
27. MONTERO, F. (2005). *Integración de calidad y experiencia en el desarrollo de interfaces de usuario dirigidos por modelos,*. Universidad de Castilla-La Mancha, España,.
28. Myers, G. (2005.). *"Fase de Elaboración. FT Prueba (procedimientos genéricos y aplicación de algunos tipos de pruebas simples).".*
29. NC ISO/IEC 9126-1. (2003). *TECNOLOGÍA DE LA INFORMACIÓN.CARACTERÍSTICAS DE CALIDAD Y MÉTRICAS DEL SOFTWARE.,*.
30. Nemury S. M. (2007). *Pruebas de Caja Negra al Módulo de Administración Financiera del Sistema Registros y Notarías.* Caracas, Distrito Federal, Venezuela.
31. Pressman, R. (2002). *Ingeniería del Software. Un enfoque práctico.*

BIBLIOGRAFÍA.

32. Pressman, R. (2002.). *“Ingeniería del software: Un enfoque práctico. Quinta edición”.* McGraw-Hill,.
33. Pressman, R. S. (2005.). *Ingenieria del software. Un enfoque práctico. Parte I.* La Habana.Cuba:: Felix Varela,2005.
34. Pressman, R. S. (2006.). *"Ingeniería de Software, un enfoque practico. DISEÑO DE CASOS DE PRUEBA.".*
35. Pressman, R. S. (1995.). *"Ingeniería del software, un enfoque práctico, Mcgraw Hill.”.*
36. Pressman, R. S. (2005.). *Ingeniería de Software, Un enfoque Práctico.*
37. Pressman., R. S. (1998). *"Ingeniería del software. Un enfoque práctico. McGrawHill”.*
38. QA&TEST. (2008). *QA&TEST 2008 Conferencia Internacional sobre Testing y Calidad del Software*. (Bilbao, España) Recuperado el 23 de febrero de 2008, de http://www.qatest.org/es/
39. Rguez, P. (2005.). Revista Española de Innovación, Calidad e Ingeniería del Software. *Vol.1*, (No. 2,).
40. RUP. (2006). *Rational Unifield Process.* IBM.
41. RUZ, F. C. (2002). *Tropa de Futuro.*
42. RUZ, F. C. (2002.). *Tropa de Futuro.*
43. Sanchez, M. A. (Junio 7 del 2004). *"Tesis: Metodologías De Desarrollo De Software Ing. Informático".* Peru: TeamSoft.
44. Sandra Hurtado de Mendoza Fernández. (s.f.). *HISTODIDACTICA: Enseñanza de la Historia/ Didáctica de las Ciencias Sociales* . Recuperado el 2008, de http://www.ub.edu/histodidactica/webquests.htm
45. SCALONE, F. (2006.). *ESTUDIO COMPARATIVO DE LOS MODELOS Y ESTANDARES DE CALIDAD DEL SOFTWARE.* . FACULTAD REGIONAL BUENOS AIRES, UNIVERSIDAD TECNOLOGICA NACIONAL.
46. SEI. (2008). *Software Engineering Institute.Carnegie Mellon*. Recuperado el marzo de 2008, de http://www.sei.cmu.edu/cmmi/
47. SHRUM, M. B. (Marzo 2006.). *Guidelines for Process Integration and Product.*
48. SOMMERVJLLE, J. (2005). *Ingeniería del software,septima edición* . Madrid Espeña.

49. *Standard glossary of terms used in Software Testing: Glosario de calidad del software.* (s.f.). Recuperado el 12 de abril de 2008, de http://squac.iti.upv.es/glosario_calidad/task,showpart/part,W/catid,30/
50. Std.1028-1997. *IEEE Standard for Software Reviews.* New York: Software Engineering Standards Committee.
51. SWEBOX. (2004). *Guide to the Software Engineering.* Los Alamitos, California.
52. TestAbil. (2007). *Kit de herramientas para el testing de software.* Recuperado el 23 de febrero de 2008, de http://www.testabil.com
53. Timothy G Olson. (25 de octubre, de 2005.). *Exitosa V&V basada en el proceso de CMMI.* Recuperado el 14 de marzo de 2008, de http://www.dtic.mil/ndia/2005systems/tuesday/olson.pdf
54. Universidad del Salvador. (2005). *Pronóstico Delphi.* Salvador.

ANEXOS.

Anexo 1: Cuestionario a Miembros del Panel de Experto

Nombre del Encuestado: _____

Ocupación: _____

1. ¿Considera usted que es importante el desarrollo de un Manual que presente las actividades a realizar por el ingeniero de pruebas, para facilitar el desarrollo del proceso de pruebas en la Universidad de las Ciencias Informáticas?

____Si __No _No Sé

¿Por Qué?

_____

_____

_____

_____

_____

2. De una valoración de 1 a 5(siendo 5 el mayor valor) de los criterios expuestos a continuación según la propuesta a validar.

_____ Satisfacción de las necesidades de los ingenieros de prueba
_____ Necesidad del empleo de la propuesta
_____ Posibilidad de aplicación
_____ Adaptabilidad a proyectos productivos
_____ Aporte al proceso de pruebas implementado en la UCI
_____ Impacto en el proceso de liberación de software.

3. ¿Considera usted que se pueda aplicar exitosamente en la universidad el Manual propuesto, de manera que permita mejorar la calidad de los productos que se desarrollan en la UCI dando el cumplimiento a su objetivo?

Si No No Sé
____ ____ ____

¿Por Qué?

_____

_____

4. ¿Que factores cree usted que podría atentar contra la correcta aplicación de este Manual en la UCI y cuales lo facilitarían?

5. ¿Considera usted que se logra demostrar mediante las Actividades de V&V que el Manual propone, que el software desarrollado tiene alta calidad?

   ------ Si ----- No ----- No Sé

   ¿Por Que?

6. Luego de conocer a profundidad el Manual para el ingeniero de prueba. Valore la eficacia que este pudiera tener.

   ___No sería eficaz.

   ___Pudiera serlo o no serlo.

   ___Sería eficaz.

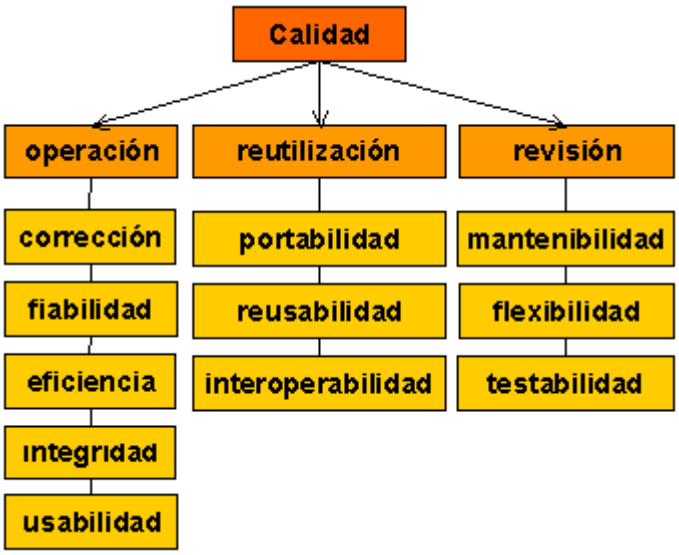
7. ¿Cree usted que contiene los elementos necesarios acorde a las características de la UCI?

   ___Si ___No ___No se

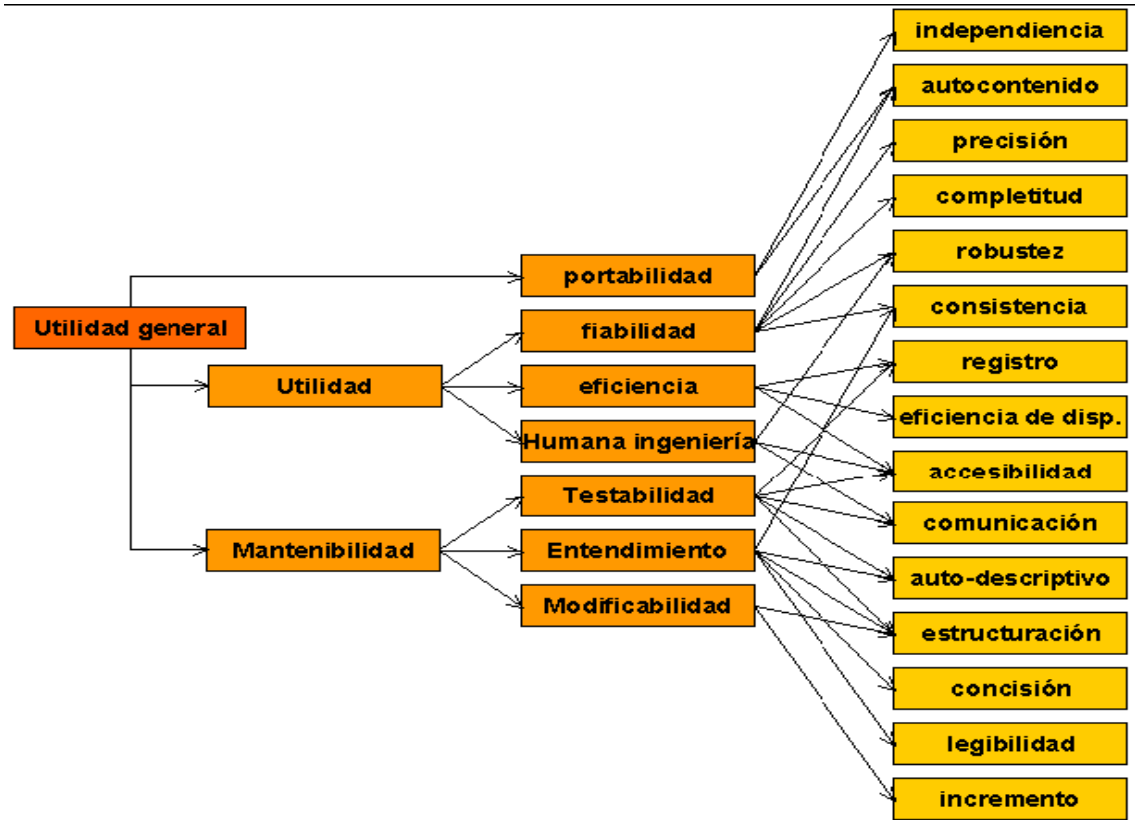
   ¿Por qué?

8. Haga un comentario o aporte sobre el Manual que usted esta evaluando. (El comentario es libre y debe reflejar algún elemento de interés que aporte elementos a la mejora del mismo):

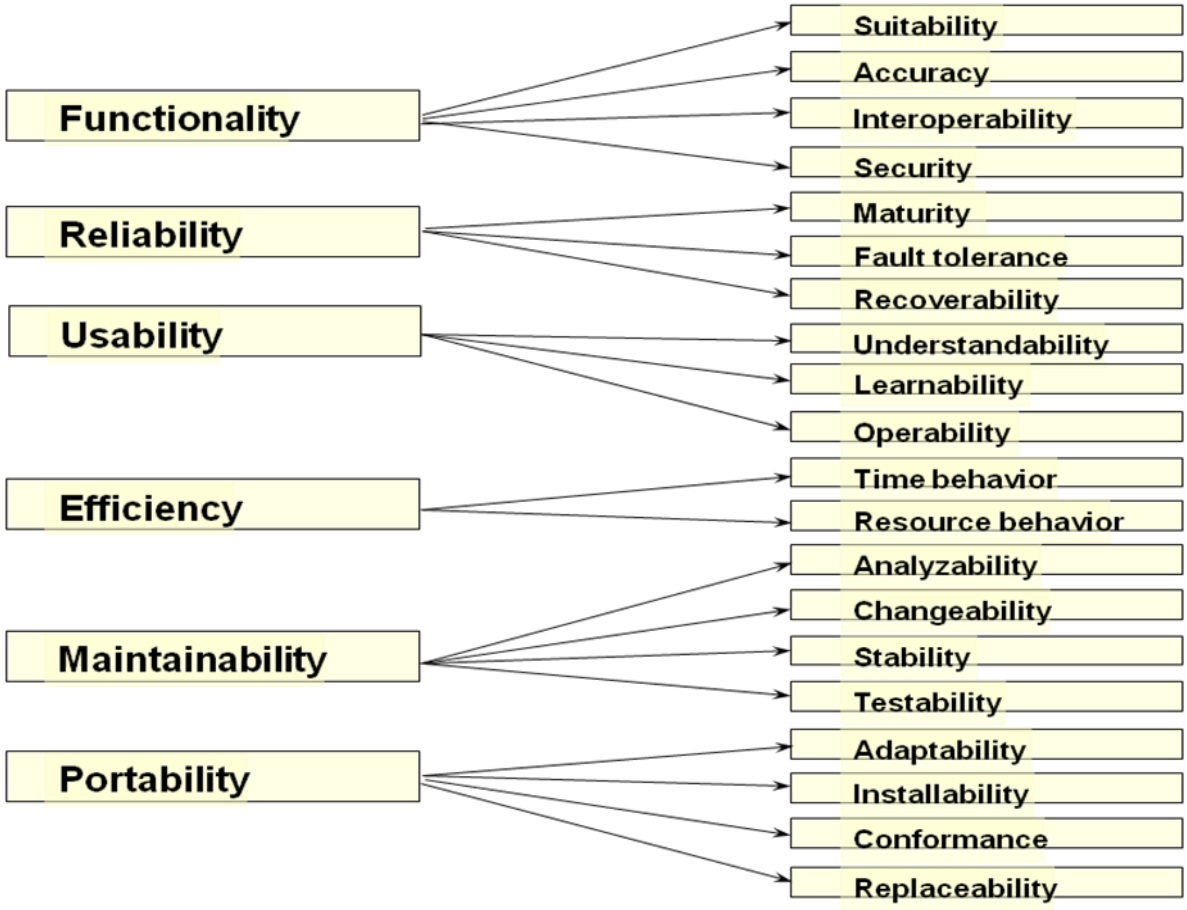
Anexo 2: Modelo de McCall.



Anexo 3: Modelo de Bohem.



Anexo 4: Modelo de ISO 9126.



.

Anexo 5: Manual del Ingeniero de Prueba. (Ver documento del manual).

GLOSARIO.

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar las actividades; representa un área de responsabilidad y es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo, o un documento. Véase trabajador, actividad.

Bugs: Es el resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador o computadora (software). Dicho fallo puede presentarse en cualquiera de las etapas del ciclo de vida del software aunque los más evidentes se dan en la etapa de desarrollo y programación. Los errores pueden suceder en cualquier etapa de la creación de software

Casos de prueba: Conjunto de entradas, precondiciones para la ejecución y salidas esperadas desarrolladas con el objetivo de testear un aspecto concreto del software (ejecutar un camino del programa en particular, verificar la conformidad de un requisito concreto, detectar tipos de errores específicos).

CMMI: Capability Maturity Model Integration. Modelo para la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Fue desarrollado por el Instituto de Ingeniería del Software de la Universidad Carnegie Mellon (SEI) y publicado en su primera versión en enero de 2002.

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

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 subcaracterísticas comportamiento temporal, utilización de recursos.

Error: Una acción humana que puede producir resultados incorrectos.

Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.

El diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.

Evolución: El software debe evolucionar, para adaptarse a las necesidades del cliente.

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

Verificación: Comprobación de que se está construyendo el producto correctamente.

Validación: Comprobación de que se está construyendo el producto correcto.

Proceso: Un conjunto de actividades interrelacionadas, que transforman los insumos en productos.

Modelo de proceso: dirige el orden de las actividades del proyecto y representa el ciclo de vida de dicho proyecto.

Tester: Probador.

Grafo causa efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Pruebas Exhaustivas: El conjunto de pruebas que se compone de todas las combinaciones de valores de entrada y de condiciones previas.

Estrés: Testeo del comportamiento del sistema bajo cargas muy altas con el objetivo de romper el sistema y encontrar los límites del sistema.

Gestión de la calidad: actividades coordinadas para dirigir y controlar una organización en lo relativo A la calidad. Dirección y control con respecto a la calidad en general, incluye la creación De la política de calidad y objetivos de calidad, la planificación de la calidad, control de calidad, la calidad Aseguramiento y mejoramiento de la calidad.

ISO/IEC 9126: Estándar que define un modelo de calidad de producto software.

Logs de pruebas: Colección de salida cruda capturada durante una ejecución única de una o más pruebas, representando generalmente las salidas, resultado de la ejecución de una prueba para un solo funcionamiento del ciclo de prueba.

Modelo: Es una abstracción del sistema, especificando el sistema desde un punto de vista y un determinando el nivel de abstracción.

MIP: Manual del Ingeniero de Prueba.

Proyectos: Un proyecto es esencialmente un conjunto de actividades interrelacionadas, con un inicio y una finalización definida, que utiliza recursos limitados para lograr un objetivo deseado.

Software: Los programas de ordenador, procedimientos y posiblemente, la documentación y los daros asociados relacionadas con el funcionamiento de un sistema informático.