Universidad de las Ciencias Informáticas

"Facultad 3"

Título: Propuesta de una guía de buenas prácticas para la gestión de la calidad de los proyectos productivos del Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD).

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor(a): Odaimy Caballero Padrón
Tutor(a): Ing. Marielis Izquierdo Matías
Tutor: DrC. Pedro Yobanis Piñero Pérez
Tutor: MsC. Michael González Jorrín

"Ciudad de la Habana, Junio 2009"

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 _____.

Odaimy Caballero Padrón

_____

Firma del Autor

DrC. Pedro Y. Piñero Pérez

_____

Firma del Tutor

"La calidad como resultado de la interacción de dos dimensiones: dimensión subjetiva (lo que el cliente quiere) y dimensión objetiva (lo que se ofrece)".

Walter A. Shewhart

DEDICATORIA

A mi familia, porque sin su ayuda y apoyo incondicional en estos años de mi vida no hubiese llegado hasta aquí, por sus sabias palabras, por indicarme siempre el camino correcto, por sus rezos y sus desvelos en cada examen, por su sacrificio, por su confianza y su amor, gracias por todo.

AGRADECIMIENTOS

A mi mama primeramente por darme la vida, por su confianza y apoyo incondicional durante todos estos años, por estar siempre ahí y ser mi orgullo.

A mis papas por ser tan lindos conmigo y darme los gustos siempre que se podía, en especial a Yoti que se esforzó muchísimo en mi educación y siempre estuvo presente cuando lo he necesitado y mi papito que lo extraño mucho y sé que se siente muy orgulloso de mi.

A mis hermanos que son la luz de mis ojos, especialmente a mi flaca linda que siempre me ha seguido los pasos y le he servido de ejemplo.

A mis abuelitos por estar ahí desde que existo, por sus mimos y su presencia divina en mi crianza.

Al amor de mi vida Félix por llegar en el momento justo en que necesitaba de su amor, por ser ante todo amigo y apoyarme en todo lo que necesitaba. Gracias a ti conocí lo que es el amor. Te amo

A mis compañeros de aula de todos estos años con los cuales compartí muchas alegrías y a los cuales recordare siempre.

A todos mis amigos y compañeras de apartamento, especialmente a Malenilla que estuvo ahí en todo momento, los quiero mucho a todos.

A mis tutores Marielis, Pedro y Michael por brindarme todo su apoyo, conocimiento y depositar su confianza en mí para llevar a cabo esta tesis.

Y en general a todos los que han contribuido de una manera u otra a mi formación.

RESUMEN

El presente trabajo se realizo en el Centro de Tecnologías de Almacenamiento y Análisis de Datos de la Universidad de las Ciencias Informáticas con el objetivo de elaborar una guía que contenga buenas prácticas de Calidad Total para los procesos de gestión de la calidad de los proyectos productivos que se desarrollan en dicho Centro y contribuir con esto al aumento en la detección de No Conformidades favoreciendo así a la reducción de errores en el producto final.

En la investigación se recolectan conceptos e ideas de diferentes autores y bibliografías referentes al tema de calidad de software que ayudaron a enriquecer el conocimiento acerca del tema y así poder cuantificar los problemas existentes con respecto a la calidad de software en la institución. Se realizó un estudio de los diferentes modelos de desarrollo de software para decidir cual de ellos sería la base para la guía propuesta La calidad es un proceso que debe aplicarse a lo largo de todo el proceso de desarrollo del software, esto es lo que se quiere lograr con dicha guía y para esto se estudiaron diferentes modelos de pruebas que sirvieron de apoyo a la misma.

El trabajo tiene un valor teórico-práctico y metodológico pues se generaliza el concepto de calidad como una necesidad inmediata para la institución, proponiéndose diferentes aspectos que se deben cumplir una vez que se aplique la investigación durante la producción.

PALABRAS CLAVES: Calidad Total, Gestión de la Calidad, Calidad de Software, Proyectos Productivos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Según la definición del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) “La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”

Hablar de calidad es algo que hace la mayoría de las organizaciones, pero dar calidad a los clientes es algo que muy pocas se aseguran de hacer. Los sistemas de calidad son aplicables y necesarios para todo tipo de organizaciones, todas tienen como principal objetivo darle al cliente lo que solicita para que de esta forma la rentabilidad de la empresa esté garantizada. Por otro lado, existe confusión de lo que significa calidad de las empresas.

La calidad es un concepto ligero, fácil de visualizar y sin embargo difícil de medir, algunos autores la definen como un término subjetivo para el cual cada persona tiene su propia definición la cual puede ser a la vez, absoluta y relativa.

El término calidad ha ido evolucionando a lo largo de la historia adaptándose a diferentes interpretaciones y manifestando en todo momento su carácter subjetivo. El nacimiento del concepto actual de calidad se puede situar en la adopción de la “Calidad Total” propuesta por Joseph M. Juran y W. Edwards Deming. La calidad daba así un gran salto, pasando de la mera inspección a la mejora de los procesos de una organización incluyendo a las personas involucradas en dichos procesos. El objetivo de este ya no es la ausencia de defectos sino la adecuación a los requisitos de negocio y del cliente.

Así la calidad está presente en todas las actividades del ciclo de vida de un proyecto de software. Para ello, se hace imprescindible disponer de una serie de pilares sobre los que sustentar las actividades de calidad: infraestructura apropiada de soporte, un grupo de personas especializada en esta disciplina y procesos alineados con los objetivos de negocio que permitan una mayor industrialización en el desarrollo y mantenimiento. La calidad del software puede gestionarse a diferentes niveles:

- A nivel de producto: Cuando nos centramos en el proceso de desarrollo de software y hacemos una serie de pruebas en paralelo con cada etapa, para detectar y corregir los posibles defectos que puedan surgir.

- A nivel de proyecto: Cuando nos centramos en controlar todas las fases y áreas de gestión de proyecto, implantando metodologías y mejores prácticas que aseguren la correcta gestión de las mismas.

- A nivel de proceso: Cuando nos centramos en gestionar todas las áreas de proceso de una organización, mediante la implantación de una metodología. Así se consigue tener mayor información de los procesos de modo que puedan controlarse y mejorarse, y produzcan así un aumento de la calidad de los productos y servicios relacionados con ellos.

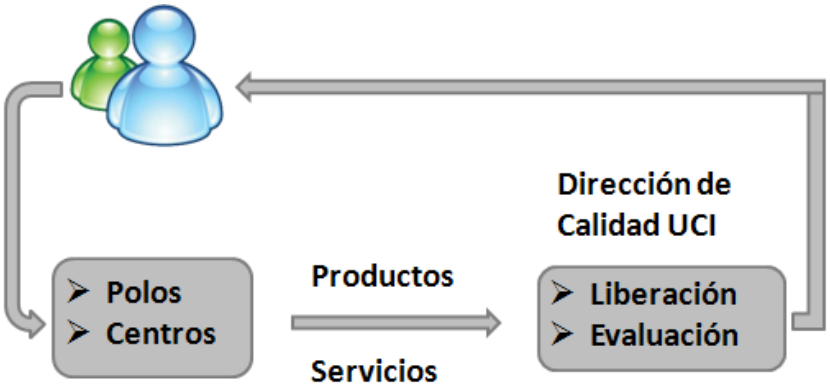
Estudios realizados demuestran que un alto porcentaje del éxito o fracaso del proyecto no solamente está en la tecnología disponible y en el conocimiento que se tenga de ella, sino también en la forma en que el proyecto no lleva un control de calidad durante su desarrollo.

La calidad de software es uno de los problemas que se afrontan actualmente en la rama de la informática. Desde la década de los 70 este tema ha sido motivo de preocupación para ingenieros, investigadores y comercializadores de software, los cuales han realizado gran cantidad de investigaciones con dos objetivos fundamentales: ¿Cómo obtener un software con calidad? ¿Cómo evaluar la calidad del software?

Cuba es uno de los países donde el desarrollo del software es aún incipiente, por ello una de las principales tareas del Gobierno Cubano es desarrollar nuestra propia Industria del Software, no solo con el fin de desarrollar sistemas para la informatización de la sociedad sino también por los beneficios de insertarnos en el mercado a nivel mundial y así mejorar la economía del país. Para obtener esto es necesario hacer productos con calidad que sean capaces de mantener los siguientes criterios: usabilidad, mantenimiento, eficiencia, seguridad, funcionalidad y estabilidad, la equivalencia de estos criterios hacen que un producto tenga calidad.

En la Universidad de las Ciencias Informáticas (UCI) se desarrollan software que son de gran ayuda a la vida económica, política y social del país. Todos estos productos son desarrollados por equipos de trabajo que están compuestos por distintos roles cada uno con sus tareas definidas, pero el objetivo principal es uno**:** lograr un producto de alta calidad.

La **estructura productiva** que se lleva a cabo en la UCI está conformada de la siguiente manera:



De acuerdo a dicha estructura para ir de lo más amplio a lo específico, en el caso que estamos enmarcados viene siendo el Centro y dentro de el los productos que son el punto de partida de la investigación.

Desde que se comienza a desarrollar un producto se quiere lograr que el proceso de calidad sea el más óptimo y asegurar en gran medida el mínimo de errores, para esto es importante mantener una organización del proceso de pruebas que cubra en todo momento las fases del ciclo de vida y así lograr poco a poco que el producto cuente con un proceso de gestión de la calidad lo mas eficaz posible.

La Gestión de la Calidad de Software es aun un tema novedoso para la UCI y principalmente para el Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD).

El Centro fue creado a finales de septiembre del año 2008 con la siguiente misión:

- Proveer soluciones integrales y consultorías relacionadas con tecnologías de bases de datos y análisis de información.
- Para desarrollar nuevas tecnologías de bases de datos, de procesamiento y representación de la información a partir del desarrollo de proyectos de I+D (Investigación y Desarrollo)
- Contribuir con su trabajo al cumplimiento de las misiones fundamentales de la universidad: la formación y la producción de software. Con profesionales integrales comprometidos con un alto nivel científico y productivo.

Su visión está enmarcada en:

- Alcanzar prestigio internacional a mediano y largo plazo, su visibilidad se refleja en los proyectos y contratos que establece, la calidad de sus producciones y las publicaciones científicas de alto nivel que desarrolla.
- Mantener estrecha colaboración con importantes empresas, centros de investigación y desarrollo de alto nivel así como con comunidades internacionales de desarrollo.

La dirección a nivel del Centro está organizada de la siguiente manera

- Director
- Subdirector Producción
- Subdirector Formación
- Subdirector Negocio

Durante la investigación se detectaron muchos de los problemas existentes con respecto a la calidad en los proyectos que se desarrollan en CENTALAD.
Actualmente no se cuenta con una estructura organizativa que permita de forma estándar la aplicación de los procesos de gestión de la calidad en los proyectos productivos que se desarrollan en Centro. El proceso de pruebas presenta insuficiencias ya que no se realizan desde que comienza las primeras fases del desarrollo del producto contribuyendo así al atraso en la entrega de los documentos por la existencia de errores que no fueron vistos durante el proceso de desarrollo y al final provocan una demora en la liberación del producto. No se documenta correctamente la estrategia de pruebas que le será aplicada al software lo que provoca un desorden a la hora de efectuar las pruebas. Existía un desconocimiento acerca de los roles y responsabilidades del grupo que lleva a cabo la calidad provocando así desorden en las pruebas recayendo sobre pocas personas el gran peso de las pruebas.

Dada esta problemática el **problema a resolver** es el siguiente: ¿Cómo introducir buenas prácticas de Calidad Total en los procesos de gestión de la calidad de software de los proyectos productivos del CENTALAD para contribuir al aumento en la detección de No Conformidades?

A partir del problema se enmarca como **objeto de estudio** la Gestión de Proyecto de Software

Según lo mencionado anteriormente el **objetivo general** de la investigación es:
Establecer una guía que disponga de buenas prácticas de Calidad Total para los procesos de gestión de la calidad de los proyectos productivos que se desarrollan en el CENTALAD.

Los **objetivos específicos** son los siguientes**:**

- Realizar un estudio del estado del arte acerca de los diferentes enfoques sobre la gestión de la calidad de los procesos de desarrollo de software y hacer valoraciones de los mismos.
- Elaborar una estructura organizativa que contenga buenas prácticas de Calidad Total para los procesos de gestión de la calidad de los proyectos productivos de CENTALAD.
- Implantar parcialmente la propuesta en los proyectos productivos de CENTALAD.

El **campo de acción** es la Gestión de la Calidad de Software.

Hipótesis: Si se establece una guía que disponga de buenas prácticas de Calidad Total en los proyectos productivos que se desarrollan en CENTALAD, se contribuiría a la organización de los procesos de gestión de la calidad de los proyectos y al aumentando en gran medida de la detección de No Conformidades logrando así un producto con el mínimo de errores.

Para dar cumplimiento al objetivo de este trabajo se trazaron las siguientes **tareas:**

- Sistematización en el estudio del estado del arte acerca de las tendencias actuales de la gestión de la calidad existentes en el mundo que puedan aportar al desarrollo de la investigación.
- Análisis del enfoque del Project Management Institute acerca de la gestión de la calidad basada en los principios de calidad total.
- Análisis del enfoque de la IEEE para la gestión de la calidad.
- Análisis del enfoque de Pressman acerca de la gestión de la calidad.
- Definición de las listas de chequeo de los documentos para la correcta aplicación de las pruebas en CENTALAD.
- Definición del Plan de Aseguramiento de la calidad de los proyectos que se llevan a cabo en CENTALAD.
- Definición de las plantillas y de la estrategia de organización de las revisiones técnicas formales del proceso de desarrollo de software de CENTALAD.
- Adaptación del proceso de pruebas establecido en el Laboratorio Industrial de Prueba de la Dirección de Calidad UCI en el desarrollo de los proyectos de CENTALAD.
- Implantación parcial de la propuesta en algunos proyectos de CENTALAD.
- Análisis de los resultados de la aplicación de la propuesta en los proyectos de CENTALAD.

Este trabajo tiene una estructura de tres capítulos. En el capítulo 1 se realiza una fundamentación teórica, con el objetivo de estudiar el estado actual de la calidad de software, analizar los principales aspectos referentes a la gestión de la calidad, su evolución en el mundo del software, descripción de diferentes modelos de procesos de desarrollo de software, modelos de pruebas y algunos conceptos emitidos por autores acerca de la calidad de software que sirvieron de apoyo para dicha investigación. El capítulo 2 describe la estructura de la guía propuesta para los procesos de gestión de la calidad de los proyectos productivos de CENTALAD con su representación gráfica y la del proceso de desarrollo a seguir. El capítulo 3 describe la parte práctica con la aplicación de la guía propuesta en algunos de los proyectos, donde se realizan comparaciones en cuanto a las técnicas utilizadas durante la revisión de los documentos obteniéndose resultados que certifican la factibilidad de la propuesta establecida.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo está dirigido a recoger los aspectos más significativos relacionados con algunas temáticas abordadas en las fuentes bibliográficas que fueron analizadas y sirvieron de apoyo durante la investigación, y algunos criterios emitidos por autores acerca del estado actual del tema estudiado. Se analizaron diferentes modelos de procesos de desarrollo de software para precisar cuál de ellos sería la base de la propuesta. Se revisaron algunos conceptos tales como calidad de software, calidad total, gestión de la calidad, aseguramiento de la calidad como un elemento dentro de la gestión, control de la calidad y mejora de la calidad

1.2 La Industria del Software a nivel mundial

Las industrias de software son organizaciones capaces de diseñar y desarrollar productos con calidad aceptada en el ámbito mundial y bajo criterios de rentabilidad, planificación, diseño y organización, es decir con una fuerte tendencia a la efectividad. Operan tomando como base metodologías y certificaciones de calidad, conocidas internacionalmente: CMMI (Capability Maturity Model Integrated), Serie ISO 9000-3, ISO/IEC 15504. Todo ello basado en el capital humano, siendo este la columna vertebral de nuestro proyecto.

La industria de software es capaz de involucrar la investigación, el desarrollo, la comercialización y la distribución de software. Actualmente vivimos en una revolución informática y los paradigmas que se conocían hace años no son los mismos.

El desarrollo de software constituye un sector de gran importancia mundial, se encuentra en el centro de todas las grandes transformaciones. La industria de software es considerada una industria que no contamina y que genera fuentes de trabajo bien remuneradas. Actualmente el software representa una oportunidad para la industria, no requiere grandes inversiones y la materia prima es el capital humano. Existen países que han alcanzado grandes éxitos como son: la India e Israel, algunos países de América Latina como México, Colombia, Brasil, y otros como China, Canadá, Estados Unidos y Costa Rica.

El crecimiento de la ISW en la India ha dejado boquiabiertos a muchos con su plan de alcanzar la escalofriante suma de $ 50.000 millones en exportaciones de software para 2008, objetivo que representaría un valor cercano al 8% de su Producto Interno Bruto. El éxito ha sido tan acelerado que, sólo por poner un ejemplo, quince años atrás el mercado TI (Tecnologías de la Información) local

alcanzaba los $ 20 millones anuales; mientras que ya en 2003 alcanzaba los $ 16.494 millones, de los cuales el país exportó nada menos que $ 10.760 millones, es decir algo más del 65%. Además, este valor representó el 17,4% del volumen total de exportaciones del país asiático. Es innegable el hecho de que la India se ha consolidado como uno de los países líderes en exportación de software y servicios informáticos en todo el mundo. La India cuenta con alrededor de 300.000 profesionales calificados trabajando en la industria TI y afines. Además, casi 100.000 ingenieros en software se reciben al año, aportando una fuente de conocimientos inagotable para la industria local. Los jóvenes se han inclinado hacia una especialización informática debido a un fuerte nivel de conocimiento previo en matemáticas, idiomas y lógica. [1]

Con un valor de mercado cercano a 10000 millones de dólares en 2006, equivalente a 50% del total latinoamericano, la industria de software y servicios de información brasileña enfrenta el doble reto de admitir a las empresas transnacionales en su mercado interno, al tiempo que busca explotar sus ventajas competitivas para ampliar su presencia en los mercados internacionales.

La industria costarricense de software alcanzó $70 millones de dólares en exportaciones contribuyendo así al desarrollo económico de Costa Rica. Para asegurar que esta industria siga brindando un impacto positivo es importante asegurar su crecimiento. Los EUA y Canadá, son países posicionados internacionalmente y que la India, Israel e Irlanda son fuertes proveedores de software manteniendo cifras de exportación entre uno y ocho billones de dólares. Existen otros países como Rusia y China que están logrando participaciones interesantes exportando entre $350 y $600 millones respectivamente. [2]

Una de las condiciones que deben cumplir las empresas para insertarse en los mercados internacionales y atraer inversión extranjera es la generación de confianza en el abastecimiento de los productos y servicios de software. Esta confianza se demuestra con la implementación de estándares reconocidos mundialmente tal es el caso de ISO 9001:2000, o el modelo de madurez (CMM) del SEI (Software Engineering Institute) [2]. La calidad del software debe ser demostrable y esto se puede lograr a través de certificaciones que es una herramienta de credibilidad de cara a los mercados internacionales.

1.3 La industria del software en Cuba

La Industria Cubana del Software (ICSW) está llamada a convertirse en una significativa fuente de ingresos para el país, como resultado del correcto aprovechamiento de las ventajas del alto capital humano disponible. La promoción de la industria cubana del software en el ámbito internacional ha tenido como línea estratégica aprovechar la enorme credibilidad que tiene Cuba en sectores tales

como la salud, la educación y el deporte. Continuar la producción sostenida de software de alta calidad en prestaciones, imagen y soporte, para satisfacer las necesidades nacionales en estos sectores, tendrá una positiva repercusión en el incremento de la exportación.

Numerosos son los logros que se han alcanzado hasta el momento como resultado de los grandes esfuerzos realizados por el gobierno cubano, al punto que se puede decir que las TICs se han insertado en casi todas las ramas de nuestra sociedad. Actualmente se sigue perfeccionando el trabajo y ampliando el radio de acción de las nuevas tecnologías en beneficio de todas las personas. Se habla y planifican metas ambiciosas que están a la altura de los países del primer mundo y que ya hoy no estamos muy lejos de poderlas alcanzar, ejemplo de ello es la Industria Cubana del Software, que con la participación de la Universidad de las Ciencias Informáticas (UCI) y otras empresas productoras de software del país se están dando grandes pasos.

Cuba en este momento está en pleno desarrollo de la industria software y como parte importante de ese desarrollo se encuentra la UCI con dos misiones fundamentales: formar profesionales comprometidos con su Patria, 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.

Como parte de los esfuerzos por mejorar la calidad de los productos desarrollados en Cuba se creó el Centro Nacional de Calidad de Software (CALISOFT), el cual se encarga, de controlar la calidad y la realización de las pruebas para la liberación de los productos de software. CALISOFT ha participado en las Pruebas de Aceptación a importantes productos desarrollados en la UCI, viabilizando, organizando y desarrollando los procesos de pruebas. Otro ejemplo de la importancia que actualmente se le da a las pruebas, es la creación en la UCI del Laboratorio Industrial de Prueba de la Dirección de Calidad, para el control a los productos, antes que sean entregados o presentados a sus clientes y usuarios finales.

La UCI pretende ser la vanguardia del desarrollo de las empresas de software en Cuba y de llevar la informatización a todos los sectores de la sociedad: Salud, Educación, Cultura, Deporte, Turismo, Prensa, etc. Regir y propiciar un avance tecnológico y de la industria del software en Cuba, convertir la industria del software en un renglón fundamental de la economía e insertarnos en el mercado internacional, por lo que el reto de nuestra universidad es producir software de alta calidad. La Universidad tiene más de 160 proyectos productivos en ejecución, está dotada de una sólida plataforma tecnológica para la docencia, una red interna integrada por más de 7 000 computadoras, la mayor de su tipo existente en el archipiélago cubano.

Se puede asegurar que en los últimos años, la Informática viene creciendo en los segmentos del quehacer económico y social de nuestro país y ejemplo de ello son los siguientes datos: [3]

- Cinco años atrás contábamos con poco más de 150 mil computadoras. Hoy se calculan más de 300 mil y su incremento no se detiene.

- Los dominios, que ya suman más de 1 200 solo en .cu, con alrededor de 1 500 sitios en Internet, 790 mil usuarios de correo electrónico y 150 mil de Internet.

- La Red Telemática de la Salud, INFOMED, cuenta con servicios de Universidad Virtual, Biblioteca Virtual, Red de Telemedicina, Acceso a Bases de Datos Especializadas y correo electrónico.

- Se han equipado con computadoras al 100% de las escuelas primarias, secundarias y preuniversitarias con lo que se benefician más de 2 millones de estudiantes y más de 12 mil centros educacionales.

- Se crean las condiciones de infraestructura necesarias para conectar a la red en una primera etapa los 16 Institutos Superiores Pedagógicos del país, y a éstos con sus 199 Sedes Universitarias Municipales.

- La Universidad de las Ciencias Informáticas (UCI), que prepara fuerza de trabajo altamente calificada para desarrollar esencialmente la informatización de la salud, la educación cubana, y la producción de software.

- Los estudiantes universitarios cubanos cuentan como promedio con 1 máquina por cada 12 estudiantes y disponen de correo electrónico, acceso a cursos, materiales de estudio y bases de datos por la vía de las redes locales y el acceso a Internet en laboratorios de computación.

- Los Joven Club de Computación y Electrónica en sus 16 años de trabajo han preparado de forma gratuita a más de medio millón de cubanos y prestado importantes servicios a centros de la salud, escuelas, instituciones estatales y otras organizaciones comunitarias en diferentes grados de utilización de las TIC.

- Existen 301 instalaciones, ubicadas en los 169 municipios del país, y cuatro laboratorios móviles para llevar los conocimientos asociados a las tecnologías de la información a zonas de difícil acceso.

1.4 ¿Qué es la calidad de software?

En toda empresa grande o pequeña, o en cualesquiera de sus áreas se manejan entornos que tienen diferentes niveles de actuación, desde la parte administrativa hasta las áreas donde se desarrolla o proporciona el producto o servicio, procesos que van enfocados a ser parte de una comunidad que también exige resultados que sean al menos igual a sus expectativas. Vivimos rodeados de esa mágica palabra llamada calidad y la exigimos, pero en definitiva, ¿sabemos de verdad lo que significa calidad?

De acuerdo a la definición del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) “La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”.

Esto en otras palabras significa darle al cliente lo que quiere, cuando lo necesita, es darle una buena razón para que vuelva a comprar nuestros productos. Calidad es en definitiva, no solo la calidad específica del producto, es también la rapidez y la forma de satisfacer las necesidades del cliente, entregar pedidos en forma correcta, facturar correctamente y una amplia respuesta de nuestros servicios post-venta.

Cuando se habla de calidad no quiere decir mejor, sino lo mejor para el cliente en servicio, es ser preciso, reducir al mínimo la variabilidad y hacer las cosas bien desde el primer momento, teniendo siempre en cuenta los requisitos planteados. La calidad de un producto o de un sistema es en su mayor parte consecuencia de la calidad de los procesos empleados en su desarrollo y mantenimiento.

1.4.1 Definiciones de calidad por categorías de enfoque [4]

Basadas en la fabricación:

"Calidad (significa) conformidad con los requisitos". Philip B. Crosby.

"Calidad es la medida en que un producto específico se ajusta a un diseño o especificación".
Harold L. Gilmore.

Basadas en el cliente:

"Calidad es aptitud para el uso". J.M.Juran.

"Calidad total es liderazgo de la marca en sus resultados al satisfacer los requisitos del cliente

haciendo la primera vez bien lo que haya que hacer". Westinghouse.
"Calidad es satisfacer las expectativas del cliente. El Proceso de Mejora de la Calidad es un conjunto de principios, políticas, estructuras de apoyo y prácticas destinadas a mejorar continuamente la eficiencia y la eficacia de nuestro estilo de vida". AT & T
"Se logra la satisfacción del cliente al vender mercancías que no se devuelven a un cliente que sí vuelve". Stanley Marcus.

Basado en el producto:
"Las diferencias en calidad son equivalentes a las diferencias en la cantidad de algún ingrediente o atributo deseado". Lawrence Abbott.
"La calidad se refiere a la cantidad del atributo no apreciado contenido en cada unidad del atributo apreciado". Keith B. Leffler.

Basado en el valor:
"Calidad es el grado de excelencia a un precio aceptable y el control de la variabilidad a un costo aceptable". Robert A. Broh.
"Calidad significa lo mejor para ciertas condiciones del cliente. Estas condiciones son: a) el uso actual y b) el precio de venta del producto". Armand V. Feigenbaum.

1.4.2 Definiciones de organizaciones reconocidas y expertos [5]

- Definición de la norma ISO 9000: “Calidad: grado en el que un conjunto de características inherentes cumple con los requisitos”
- Real Academia de la Lengua Española: “Propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla como igual, mejor o peor que las restantes de su especie”
- Edwards Deming: "la calidad no es otra cosa más que "Una serie de cuestionamiento hacia una mejora continua".
- Dr. J. Juran la calidad es: "La adecuación para el uso satisfaciendo las necesidades del cliente".
- Kaoru Ishikawa define a la calidad como: "Desarrollar, diseñar, manufacturar y mantener un producto de calidad que sea el más económico, el útil y siempre satisfactorio para el consumidor".
- Rafael Picolo, Director General de Hewlett Packard define: "La calidad, no como un concepto aislado, ni que se logra de un día para otro, descansa en fuertes valores que se presentan en el medio ambiente, así como en otros que se adquieren con esfuerzos y disciplina".

- Philip Crosby:”Calidad es cumplimiento de requisitos”
- Genichi Taguchi: “Calidad es la menor pérdida posible para la sociedad”.
- Walter A. Shewhart:”La calidad como resultado de la interacción de dos dimensiones: dimensión subjetiva (lo que el cliente quiere) y dimensión objetiva (lo que se ofrece)”.

1.4.3 Evolución de la calidad

La evolución de la calidad debe analizarse en diferentes periodos como muestra la Tabla # 1. [6]

Etapa	Concepto	Finalidad	Periodo y ejemplos
Artesanal	Hacer las cosas bien independientemente del coste o esfuerzo necesario para ello. Control individual de cada tarea.	Satisfacer al cliente. Satisfacer al artesano, por el trabajo bien hecho Crear un producto único.	1918: Ford Motor Company. (Primera cadena de montaje). 1930: Laboratorios Bel.
Revolución Industrial	Hacer muchas cosas no importando que sean de calidad (Se identifica Producción con Calidad).	Satisfacer una gran demanda de bienes. Obtener beneficios.	
Segunda Guerra Mundial	Asegurar la eficacia del armamento sin importar el costo, con la mayor y más rápida producción (Eficacia + Plazo = Calidad)	Garantizar la disponibilidad de un armamento eficaz en la cantidad y el momento preciso.	
Posguerra (Japón)	Hacer las cosas bien a la primera	Minimizar costes mediante la Calidad Satisfacer al cliente Ser competitivo	Entre 40 y 70 Deming y Ishikawa
Postguerra (Resto del	Producir, cuanto más	Satisfacer la gran	

mundo)	mejor	demanda de bienes causada por la guerra	
Control de Calidad	Técnicas de inspección en Producción para evitar la salida de bienes defectuosos.	Satisfacer las necesidades técnicas del producto.	Hasta los años 80
Aseguramiento de la Calidad	Sistemas y Procedimientos de la organización para evitar que se produzcan bienes defectuosos.	Satisfacer al cliente. Prevenir errores. Reducir costes. Ser competitivo.	A partir de los 80 · 1980. Interés por la calidad en los EEUU. · 1987. Premio *Malcom Baldrige Quality Award* · *1987.* ISO 9000. A partir de las normas británicas · 1992. Premio Europeo a la calidad de la EFQM
Calidad Total	Teoría de la administración empresarial centrada en la permanente satisfacción de las expectativas del cliente.	Satisfacer tanto al cliente externo como interno. Ser altamente competitivo. Mejora Continua	

Tabla # 1 Evolución de la calidad

1.5 ¿Qué es Gestión de la Calidad del Software?

Es un conjunto de actividades de la función de gestión que determina la calidad, los objetivos y las responsabilidades. Se basa en la determinación y aplicación de las políticas de calidad de la empresa. La gestión de la calidad se aplica normalmente a nivel de empresa o dentro de la gestión de cada proyecto. El propósito de la misma es entender las expectativas del cliente en términos de calidad, y poner en práctica un plan proactivo para satisfacer esas expectativas.

Desde el punto de vista de la calidad, la Gestión de la Calidad del Software está formada por 4 partes, las cuales son: Planificación de la calidad, Control de la calidad, Aseguramiento de la calidad y Mejora de la calidad. La gestión de la calidad es responsabilidad de todos los niveles ejecutivos, pero debe estar guiada por la alta dirección. Su realización involucra a todos los miembros de la organización. La gestión de la calidad garantiza el éxito de la organización en general.

La gestión de la calidad, según la norma internacional ISO 9004:2000, se basa en ocho principios, desarrollados con la intención de que la alta dirección de las empresas los emplee para liderar la organización hacia un mejor desempeño: [7]

- Enfoque al cliente
- Liderazgo
- Compromiso de las personas
- Enfoque del proceso
- Enfoque del sistema de administración
- Mejoras continuas
- Enfoque de toma de decisiones basado en hechos
- Relaciones con proveedores de beneficio mutuo

Enfoque en el cliente:

Dado que las organizaciones dependen de sus clientes, deben comprender sus necesidades actuales y futuras, satisfacer las mismas y luchar para superar las expectativas de los clientes. La organización debe ver al cliente no sólo como un consumidor sino, particularmente, como el usuario de los productos y servicios producidos por la organización y debe asegurarse de que esto se ajuste a los objetivos de la compañía. Por lo tanto, debe establecerse un sistema centrado en el cliente, de manera que la organización pueda tener un mejor panorama de las necesidades y expectativas del cliente para poder satisfacerlas de la mejor manera posible. Además, debe evaluarse regularmente la satisfacción del cliente, de modo que se puedan identificar las oportunidades y/o riesgos tan pronto como sea posible.

Liderazgo:

Los directores de la organización definen coherentemente los objetivos y la orientación de la organización. Deben crear y mantener la atmósfera interna adecuada para que los empleados se sientan totalmente comprometidos en el logro de los objetivos de la organización. El objetivo de este principio es asegurar que se tomen en cuenta las necesidades de todos los participantes en el momento de definir y formalizar una visión clara para el futuro de la organización estableciendo objetivos que satisfagan a todos.

Deben crearse valores compartidos, para reemplazar los posibles temores por una relación de confianza.

Compromiso de las personas:

Los empleados de todos los niveles constituyen la esencia de una organización. Al involucrar a todos, las aptitudes de cada individuo trabajan para la organización. Los empleados deben comprender que juegan un rol importante en la organización y deben involucrase en el proceso de establecer objetivos motivadores para sí mismos. Las habilidades del empleado deben evaluarse regularmente y deben implementarse planes de capacitación para ayudar a los empleados a evolucionar en su trabajo. A la inversa, también puede resultar útil permitir que los empleados evalúen el estilo de gestión de sus superiores y su relación de trabajo.

En ese contexto, cada empleado se inclinará cada vez más a mejorar sus habilidades para lograr sus objetivos personales y, por lo tanto, para compartir su experiencia y conocimientos.

Enfoque del proceso:

Un resultado esperado se logra con más eficacia cuando se administran las acciones y los recursos correspondientes como procesos. Por lo tanto, las actividades necesarias para lograr un resultado deben identificarse claramente como procesos y cada persona debe hacerse responsable de uno de estos procesos. La identificación de actividades puede resultar más fácil si se compromete a las partes involucradas.

Sobre esta base, es posible evaluar el desempeño de cada proceso y analizar el modo en que puede ser mejorado para satisfacer mejor los objetivos estratégicos de la compañía.

Enfoque del sistema de administración:

Identificar, comprender y administrar un sistema de procesos interdependientes para objetivos específicos permite que las organizaciones mejoren su efectividad y su eficacia. La idea de este principio es considerar que el acto de estructurar y documentar claramente las acciones que contribuyen a los objetivos de la organización permite mejorar su efectividad y eficacia. Para esto, la organización primero debe identificar las dependencias existentes para reducir los conflictos entre procesos y el trabajo repetido. Esto debe llevar a la formalización de un sistema de gestión de calidad claramente documentado.

Capacitar o informar a los participantes necesarios puede ser esencial para asegurar que todos cumplan con este paso.

Mejoras continuas:

Mejorar de manera continua debe ser uno de los objetivos permanentes de la organización. Se deben controlar los diferentes procesos y analizar su rendimiento cíclicamente para sugerir e implementar mejoras. Esto se puede llevar a cabo a través de una revisión de administración regular y con auditorías internas y externas. Es muy importante saber cómo detectar mejoras y hacer que todos las conozcan.

Enfoque de toma de decisiones basado en hechos:

Las decisiones eficaces se basan en el análisis de datos e información tangible. Este principio consiste en tomar decisiones basadas en el análisis fáctico de información corroborada por la experiencia y la intuición. Según este enfoque, después del hecho es más fácil argumentar sobre una decisión bien fundada refiriéndose a documentos disponibles.

Esto brinda los medios para que todas las personas involucradas entiendan el modo en que se toman las decisiones.

Relaciones con proveedores de beneficio mutuo:
Una organización y sus proveedores son interdependientes y una relación de beneficio mutuo mejora su capacidad para crear valores. La relación con los proveedores se debe pensar de modo de conciliar logros a corto plazo con consideraciones futuras. Para lograr esto, una organización debe comprender los intereses de sus socios, definir claramente sus obligaciones en un contrato y evaluar regularmente su desempeño. Cuando este principio se aplica correctamente, una organización puede mejorar notablemente su relación con los proveedores con respecto al tiempo de respuesta y, por lo tanto, al costo total.

1.5.1 Planificación de la Calidad del Software

Es la parte de la Gestión de la Calidad encargada de realizar el proceso administrativo de desarrollar y mantener una relación entre los objetivos y recursos de la organización, se debería realizar de forma paralela a los demás procesos de planificación del proyecto.
Los aspectos a considerar en la planificación de la calidad del software son: organización del proceso de desarrollo, costos de la calidad, recursos humanos, modelos y estándares de calidad a utilizar, etc. El plan de calidad define los atributos de calidad más importantes del producto a ser desarrollado y define el proceso de evaluación de la calidad. Los factores para determinar qué modelo o estándar de calidad se va a utilizar es la complejidad del proceso de diseño, la madurez del diseño, la complejidad del proceso de producción, las características del producto o servicio y el aspecto económico.
Una vez que una organización aprende a planificar la calidad, el tiempo total transcurrido entre el concepto inicial y las operaciones efectivas es mucho menor.

Según la Norma ISO 9000:2000, la planificación de la calidad es la parte de la gestión de la calidad enfocada al establecimiento de los objetivos de la calidad y a la especificación de los procesos operativos necesarios y de los recursos relacionados para cumplir los objetivos de calidad.

1.5.2 Aseguramiento de la Calidad del Software

Es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza que el software satisfará los requisitos dados de calidad. Este aseguramiento se diseña para cada aplicación antes de comenzar a desarrollarla y no después. El aseguramiento de la calidad del software engloba: un enfoque de gestión de calidad, métodos y herramientas de Ingeniería del Software, revisiones

técnicas formales aplicables en el proceso de software, una estrategia de prueba multiescala, el control de la documentación del software y de los cambios realizados, procedimientos para ajustarse a los estándares de desarrollo del software y mecanismos de medición y de generación de informes.

Según el PMBok el aseguramiento de calidad (QA) es la aplicación de actividades planificadas y sistemáticas relativas a la calidad, para asegurar que el proyecto emplee todos los procesos necesarios para cumplir con los requisitos.
Según la Norma ISO 9000:2000, el aseguramiento de la calidad es la parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de calidad.
El Aseguramiento de la Calidad del Software se puede definir como el esfuerzo total para plantear, organizar, dirigir y controlar la calidad en un sistema de producción con el objetivo de dar al cliente productos con la calidad adecuada, además de proporcionar personas y gestión con el objetivo de que los procesos y los elementos de trabajo cumplan los procesos estipulados, los cuales son validados a la hora de crear un producto.
Aspectos relacionados con el aseguramiento de la calidad del software: [8]

- ✓ La calidad no se puede probar, se construye.
- ✓ El aseguramiento de la calidad del software no es una tarea que se realiza en una fase particular del ciclo de vida de desarrollo.
- ✓ Las actividades asociadas con el aseguramiento de la calidad del software deben ser realizadas por personas que no estén directamente involucradas en el esfuerzo de desarrollo.

Estas acciones no son suficientes para un correcto aseguramiento de la calidad, sin embargo, reducen notablemente el riesgo de que ocurran.
El aseguramiento de la calidad incluye el involucramiento en cada etapa del ciclo de vida de desarrollo del software y se encarga de establecer procedimientos de revisión y entrena a las personas en mejores formas de diseño y desarrollo de productos. Mediante el aseguramiento de la calidad se ayuda a la compañía a prevenir defectos.

1.5.3 Control de la Calidad del Software

Son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en 2 objetivos fundamentales: mantener bajo control un proceso y eliminar las causas de los defectos en las diferentes fases del ciclo de vida.

Según J. M. Juran es el proceso de regulación a través del cual se puede medir la calidad real, compararla con las normas o las especificaciones y actuar sobre la diferencia.

Esta etapa se caracteriza por la realización de inspecciones y ensayos para comprobar si un determinado producto cumple con las especificaciones establecidas previamente. El aspecto a considerar en el Control de la calidad es la "Prueba del Software".

El control de la calidad garantiza que las actividades de un programa ocurran según fueron planeadas. Las actividades para el control de la calidad también pueden identificar fallas en el diseño y, por ende, señalar cambios que podrían mejorar la calidad. Esta actividad incluye la ejecución de un conjunto de pruebas que se establecieron siguiendo estándares y procedimientos y buscan la detección de errores.

- **Pruebas de Software**

La prueba es el proceso de ejecutar un programa con intención de encontrar defectos. Es un proceso destructivo que determina el diseño de los casos de prueba y la asignación de responsabilidades. La prueba exitosa es aquella que descubre defectos. El "caso de prueba bueno" es aquel que tiene alta probabilidad de detectar un defecto aún no descubierto. El "caso de prueba exitoso" es aquel que detecta un defecto aún no descubierto.

La prueba no es: demostración que no hay errores, demostración que el software desempeña correctamente sus funciones y establecimiento de confianza que un programa hace lo que debe hacer.

La prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la confiabilidad del software e indican la calidad del software como un todo. Pero, la prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

La IEEE, 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.
La prueba del software es un elemento importante para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado.

La etapa de pruebas implica: [8]

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

Según Pressman las pruebas del software son el proceso de ejecución de un programa con la intención de descubrir un error y tiene éxito si descubre un error no detectado hasta entonces. Se debe tener en cuenta que la prueba no puede asegurar la ausencia de defectos, sólo puede demostrar que existen defectos en el software.
Esa afirmación es de gran relevancia pues aunque no se encuentren defectos en la prueba no quiere decir que el sistema esté libre de ellos, por lo tanto la presencia de defectos no puede tomarse como responsabilidad del equipo de prueba, la prueba debe ser vista como una oportunidad del equipo de desarrollo para demostrar que el software desarrollado cumple con las especificaciones planteadas por el cliente.

Para poder desarrollar un buen proceso de pruebas es muy útil conocer algunos de sus principios: [9]

- A todas las pruebas se le debería poder hacer un seguimiento hasta los requisitos del cliente. Como se ha visto el principal objetivo es encontrar errores. Para el cliente los errores más graves son los que le impiden al sistema cumplir sus requisitos.
- Las pruebas deberían planificarse mucho antes de que empiecen. La planificación de las pruebas puede comenzar tan pronto como esté completo el modelo de requisitos. La definición detallada de los casos de prueba puede empezar una vez que se haya aprobado el modelo de diseño. Por tanto, se puede planificar y diseñar todas las pruebas antes de generar ningún código.

- El principio de Pareto es aplicable a la prueba del software. El principio de Pareto implica que al 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. El problema está en aislar estos módulos sospechosos y probarlos.
- Las pruebas deberían empezar por lo pequeño y progresar hacia lo más grande. Las primeras pruebas planeadas y ejecutadas en general se centran 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. Esto se plantea porque incluso en un programa pequeño la cantidad de permutaciones de caminos es muy grande, por lo que es imposible cubrir todas las combinaciones de caminos. Sin embargo es posible cubrir adecuadamente la lógica del programa y asegurarse de que se han aplicado todas las condiciones del diseño procedimental.
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente al desarrollo. Se ha demostrado que el ingeniero de software que creó el sistema no es el más indicado para realizar las pruebas al sistema.

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida, podemos probar la funcionalidad de los primeros prototipos; probar la estabilidad, rendimiento de la arquitectura, probar el producto final. Lo que conduce al principal beneficio de la prueba es proporcionar reacción mientras hay todavía tiempo y recursos para hacerlo.

- **Estrategia de pruebas del software**

Inicialmente, la prueba se centra en cada módulo individualmente, asegurando que funciona adecuadamente como una unidad. La prueba de unidad hace un uso intensivo de las técnicas de prueba de caja blanca, ejercitando caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores. La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: el componente de software o módulo. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada. Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de ejecución del algoritmo. Se prueban las condiciones límite para asegurar que el

módulo funciona correctamente en los límites establecidos. Se ejercitan todos los caminos independientes de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y, finalmente, se prueban todos los caminos de manejo de errores. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. Además de las estructuras de datos locales, durante la prueba de unidad se debe comprobar el impacto de los datos globales sobre el módulo.

A continuación, se deben ensamblar o integrar los módulos para formar el paquete de software completo. La prueba de integración es una técnica sistemática que permite construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es juntar los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. Se combinan todos los módulos por anticipado. Se prueba todo el programa en conjunto. Se encuentra un gran conjunto de errores. Una vez que se corrigen esos errores aparecen otros nuevos y el proceso continúa en lo que parece ser un ciclo sin fin.

Después que el software se ha integrado, se dirigen un conjunto de pruebas de alto nivel. Se deben comprobar los criterios de validación establecidos durante el análisis de requisitos. La prueba de validación proporciona una seguridad final que el software satisface todos los requisitos funcionales, de comportamiento y de rendimiento. Durante la validación se usan exclusivamente técnicas de prueba de caja negra.

El software, una vez validado, se debe combinar con otros elementos del sistema. La prueba del sistema verifica que cada elemento se ajusta 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. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se ha integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

Cuando se construye un software a medida para un cliente, se llevan a cabo una serie de pruebas de aceptación para permitir que el cliente valide todos los requisitos. Estas pruebas las realiza el usuario final en lugar del responsable del desarrollo de sistema. Una prueba de aceptación puede ir desde un informal paso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas.

El diseño de casos de prueba para el software o para otros productos de ingeniería puede requerir tanto esfuerzo como el propio diseño inicial del producto. Sin embargo, los ingenieros de software tratan las pruebas como algo sin importancia, desarrollando casos de prueba que “parezcan

adecuados", pero que tienen poca garantía de ser completos. Se deben diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo posible. Cualquier producto de ingeniería puede probarse de una de estas 2 formas: prueba de caja negra y prueba de caja blanca.

Cuando se considera el software de computadora, la prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

La prueba de caja blanca se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. Para este tipo de prueba, se deben definir todos los caminos lógicos y desarrollar casos de prueba que ejerciten la lógica del programa.

- **Prueba de caja blanca**

Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que: [9]

a) Se ejercitan todos los caminos independientes de cada módulo.
b) Se ejercitan todas las decisiones lógicas.
c) Se ejecutan todos los bucles.
d) Se ejecutan las estructuras de datos internas.

La prueba de caja blanca se apoya en técnicas como: [9]

- La prueba del camino básico o cubrimiento: a partir de la estructura del código calcula una medida que sirve como guía para la definición de un conjunto básico de caminos de ejecución que garanticen en principio se cumpla con el inciso "a)".
- La prueba de condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
- La prueba de flujo de datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- La prueba de bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

- **Prueba de caja negra**

Son pruebas que se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa.

Los casos de prueba de caja negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, producen una salida correcta y la integridad de la información externa se mantiene.

Se derivan conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del programa.

Las pruebas de caja negra intentan encontrar errores de las siguientes categorías:

- 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.

Se basan en técnicas como: [9]

- Método de prueba basado en grafos: Este tipo de prueba se basa en el análisis de las relaciones entre objetos del programa y su comportamiento. Se comienza con la creación de un grafo que modele estas relaciones definiéndose los nodos (objetos) y sus pesos (atributos), después se precisan los enlaces entre los nodos (relaciones) y sus pesos (describe alguna característica de un enlace), luego se diseñan casos de prueba con el fin de encontrar errores en las relaciones.
- Partición equivalente: Esta técnica utiliza las clases de equivalencia para el diseño de los casos de prueba, es decir, la entrada de una serie de datos válidos y no válidos, cubriendo en cada caso el máximo número de entradas.
- Análisis de valores límites: Esta técnica complementa la partición equivalente pero no solo utiliza las condiciones de entrada sino también las de salida, selecciona, además, los casos de prueba en los extremos de la clase.
- Prueba mano a mano o de comparación: Esta técnica se utiliza en situaciones críticas en las que el software debe ser sumamente fiable. En estos casos se desarrolla una serie de versiones del programa siguiendo las mismas especificaciones a las cuales se les realiza pruebas con los mismos datos de entrada para asegurar la misma salida, en caso que se obtenga la misma salida, significa que todas las implementaciones son correctas, en caso contrario, es necesario revisar todas las versiones para encontrar la causa de las diferencias.

- Prueba de la tabla ortogonal: Esta técnica se aplica en pequeños dominios de entrada.

❖ **Tipos de pruebas**

Pruebas de Verificación

- *Prueba de unidad:* La prueba de unidad se centra en el módulo. Usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca.
- *Prueba de integración:* El objetivo es coger los módulos probados en la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Hay dos formas de integración:

- Integración no incremental (big bang): Se combinan todos los módulos por anticipado y se prueba todo el programa en conjunto.
- Integración incremental: El programa se construye y se prueba en pequeños segmentos.
  - ✓ Integración ascendente: Comienza la integración de los módulos por los niveles más bajos de la estructura del programa hasta llegar al módulo principal del mismo. Primeramente se combinan los módulos en grupos que cumplan una función determinada, se crea para cada grupo, un módulo impulsor que es un programa que simula la llamada a los módulos, la entrada de datos de prueba y la recogida de los resultados. Luego de haber probado cada impulsor de los grupos, se eliminan y se sustituyen por los módulos de nivel superior de la jerarquía.
  - ✓ Integración descendente: Es el proceso inverso al de integración ascendente, se inicia con el módulo principal siguiendo una jerarquía de control hacia abajo de la forma “primero en profundidad” o “primero a lo ancho”, hasta llegar a la integración con los módulos de más bajo nivel.
  - ✓ Integración mixta o sándwich: Este tipo de integración combina la integración ascendente con la descendente, utilizando la primera para los niveles más bajos del programa y la segunda para los niveles superiores.

En la prueba de integración el foco de atención es el diseño y la construcción de la arquitectura del software. Las técnicas que más prevalecen son las de diseño de casos de prueba de caja negra, aunque se pueden llevar a cabo unas pocas pruebas de caja blanca.

Pruebas de sistema: Verifica 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.

Algunas de estas pruebas son:

- *Pruebas de Funcionalidad:* tiene como objetivo verificar la función del sistema al fijar la resistencia 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.
- *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.
- *Pruebas de Seguridad:* verifican los mecanismos de protección.
- *Pruebas de Volumen:* Verifica que la aplicación funcione adecuadamente bajo los siguientes escenarios de volumen:
    - Máximo (actual o físicamente posible) número de clientes conectados (o simulados), todos ejecutando la misma función (peor caso de desempeño) por un período extendido.
    - Máximo tamaño de base de datos (actual o escalado) y múltiples consultas ejecutadas simultáneamente.
- *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.
- *Pruebas de Compatibilidad:* Verifican el funcionamiento del sistema sobre diferentes componentes de software. Validando la aplicación con sistemas operativos y navegadores.
- *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.
- *Pruebas de Usabilidad:* Verifican la estética, la consistencia de la interfaz de usuario y documentación.

- *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.
- *Pruebas de Configuración:* se enfocan en evaluar las diferentes variaciones de una aplicación integrada, contra sus requerimientos de configuración.
- *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.

Pruebas de Validación

- *Pruebas de aceptación:* Las pruebas de aceptación tienen como fin validar que el sistema cumple los requisitos básicos de funcionamiento esperado y permitir que el usuario determine la aceptación del sistema. Estas pruebas son realizadas por el usuario final que, durante este periodo de tiempo, debe plantear todas las deficiencias o errores que encuentre antes de dar por aprobado el sistema. 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.
- *Pruebas de regresión:* 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. El propósito de estas pruebas es asegurar que:
    - Los defectos identificados en la ejecución anterior de la prueba se ha corregido.

- Los cambios realizados no han introducido nuevos defectos o reintroducido defectos anteriores.

La prueba de regresión puede implicar la re-ejecución de cualquier tipo de prueba. Normalmente, las pruebas de regresión se llevan a cabo durante cada iteración, ejecutando otra vez las pruebas de la iteración anterior.

1.5.4 Mejora de la Calidad del Software

Es la parte de la gestión de la calidad que contribuye, por medio de las mediciones, a los análisis de los datos y auditorias, a efectuar mejoras en la calidad del software.

Una auditoria de calidad tiene como objetivo mostrar la situación real para aportar confianza y destacar las áreas que pueden afectar adversamente esa confianza. Otro objetivo consiste en suministrar una evaluación objetiva de los productos y procesos para corroborar la conformidad con los estándares, las guías, las especificaciones y los procedimientos.

Para implementar un programa de mejoras es necesario definir procesos, decidir qué se quiere mejorar, definir qué medidas serán necesarias recoger, cómo y dónde tomarlas, gestionarlas mediante herramientas, utilizarlas para la toma de decisiones y reconocer las mejoras. Cuando el proceso a mejorar es el de desarrollo del software, es importante definir qué objetivos se quieren alcanzar, para reducir el número de medidas y, en consecuencia, el coste de recopilarlas y el impacto sobre la actividad de producción de software.

La calidad ha dejado de ser un tópico y es necesario que forme parte de los productos o servicios que comercializamos para nuestros clientes. El cliente es el mejor auditor de la calidad, él exige el nivel que está dispuesto a pagar por ella. La calidad de software es resultado del movimiento global dentro del proceso de mejoramiento continuo de los modelos y estándares de producción en todos los sectores industriales, en particular, cuando éste se concentra en la producción de sistemas de información y software especializado.

Según la Norma ISO 9000:2000, la mejora de la calidad es la parte de la gestión de la calidad orientada a aumentar la capacidad de cumplir con los requisitos de la calidad. Los requisitos pueden estar relacionados con cualquier aspecto tal como la eficacia, la eficiencia o la trazabilidad.

La mejora de la calidad del software logra que se desarrollen procesos de manera más productiva y eficiente, reduciendo los costos y ofreciendo un producto o servicio de calidad. Está encaminada a efectuar mejoras en la calidad del software, como su nombre lo indica.

El mejoramiento continuo de la calidad se ha convertido en una parte necesaria e integral de la estrategia de negocios de las organizaciones. Está basado en el cambio, describe dos tipos de cambio:

gradual y abrupto. El cambio gradual resulta de pequeñas mejoras al status mediante esfuerzos continuos que incluyen a todo mundo. El cambio abrupto proviene de la innovación una mejora drástica al estado actual.

1.6 ¿Qué es Calidad Total?

La calidad total es una alusión a la mejora continua, con el objetivo de lograr la calidad óptima en la totalidad de las áreas, es un concepto que explica como ofrecer el mayor grado de satisfacción a un cliente por medio de un bien o servicio. La calidad total es un sistema de gestión empresarial íntimamente relacionado con el concepto de Mejora Continua, que incluye las dos fases anteriores control y aseguramiento de la calidad. Su metodología de implantación garantiza en primera instancia un mejoramiento continuo del proceso; no existe un final o meta máxima para las instituciones que implanten esta metodología debido a que siempre habrá posibilidades de mejorar el proceso y obtener unos niveles más elevados de calidad.

1.6.1 Definiciones de diferentes autores [10]

- Armand Feigenbaum: Calidad Total es un sistema efectivo de los esfuerzos de varios grupos en una organización para la integración del desarrollo, del mantenimiento y la mejora de la calidad con el objetivo de hacer posibles marketing, ingeniería, producción, y servicio a satisfacción total del consumidor y al nivel más económico.
- Philip Crosby: Calidad Total es el cumplimiento de los requerimientos, donde el sistema es la prevención, el estándar es cero defectos y la medida es el precio del incumplimiento.
- Joseph Juran: Calidad Total es estar en forma para el uso (fitness for use), desde los puntos de vista estructurales, sensoriales, orientados en el tiempo, comerciales y éticos en base a parámetros de calidad de diseño, calidad de cumplimiento, de habilidad, seguridad del producto y servicio en el campo.
- Genichi Taguchi: Calidad Total es la pérdida mínima impartida a la sociedad por el producto desde el momento en que se despacha (considerando reproceso, mantenimiento, desechos, tiempo sin ser usado a causa de fallas, reclamos por garantías, y bajo rendimiento del producto).
- Kaoro Ishikawa: Calidad Total es cuando se "encompass after" servicio de ventas, administración, la compañía en sí misma y el ser humano.

Con lo anterior se puede concluir que la calidad total se define como: un proceso de mejoramiento continuo, donde todas las áreas de la empresa participan activamente en el desarrollo de productos y servicios, que satisfagan las necesidades del cliente, logrando con ello mayor productividad.

1.6.2 Características de la calidad total [11]

Visión de largo plazo: Calidad total implica transformaciones y, sobre todo, trabajo de cada persona involucrada en la firma y orientada hacia el consumidor. Usualmente sus resultados no son inmediatos. Es necesario persistir en el tiempo, para lograr el éxito esperado a través de la corrección aplicación del proceso de calidad y los estudios de retorno que medirán la satisfacción de nuestros consumidores.

Compromiso de la alta gerencia: Esta es una necesidad evidente, ya que la iniciativa envuelve a toda la compañía, por ello no se puede llevar a cabo sin el apoyo de la Gerencia General. La alta Gerencia no sólo no puede estar ausente, si no que es necesario que establezca liderazgo en los programas tendientes a lograr satisfacción a través de la calidad, predicando con el ejemplo.

Administración participativa: Dicha participación se expresa en recolección y análisis de datos, generación y discusión de ideas, entre muchos otros aspectos. Requiere de la participación de todos sus integrantes.

Trabajo en equipo: Como la satisfacción depende de muchos factores, es necesario enfrentar los problemas y el desarrollo de los procesos en equipo. En la empresa, el trabajo coordinado permitirá descubrir fuentes de errores y fallas y, consecuentemente tomar medidas correctivas para ir mejorando.

La calidad total tiene tres dimensiones:

- **Gestión:** el cuerpo directivo está totalmente comprometido.
- **Calidad:** los requerimientos del cliente son comprendidos y asumidos exactamente.
- **Total:** todo miembro de la organización está involucrado, incluso el cliente y el proveedor, cuando esto sea posible.

1.6.3 Principios de la Calidad Total [12]

- ✓ Consecución de la plena satisfacción de las necesidades y expectativas del cliente (interno y externo).
- ✓ Desarrollo de un proceso de mejora continua en todas las actividades y procesos llevados a cabo en la empresa (implantar la mejora continua tiene un principio pero no un fin).
- ✓ Total compromiso de la Dirección y un liderazgo activo de todo el equipo directivo.
- ✓ Participación de todos los miembros de la organización y fomento del trabajo en equipo hacia una Gestión de Calidad Total.
- ✓ Involucración del proveedor en el sistema de Calidad Total de la empresa, dado el fundamental papel de éste en la consecución de la Calidad en la empresa.
- ✓ Identificación y Gestión de los Procesos Clave de la organización, superando las barreras departamentales y estructurales que esconden dichos procesos.
- ✓ Toma de decisiones de gestión basada en datos y hechos objetivos sobre gestión basada en la intuición. Dominio del manejo de la información.

1.7 ¿Qué es un proceso de desarrollo de software?

El proceso de desarrollo de software es una descripción de la construcción del software que contiene actividades organizadas y orientadas a ese fin. No existe una definición estándar de estas actividades y muchos autores le dan importancia a algunas más que a otras. Este proceso no es único, no existe un proceso de desarrollo universal que sea efectivo para todos los proyectos. A pesar de la variedad de propuestas de procesos, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos: [13]

- **Especificación de software**: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
- **Diseño e Implementación**: Se diseña y construye el software de acuerdo a la especificación.
- **Validación**: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
- **Evolución**: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Pressman caracteriza un proceso de desarrollo de software como se muestra a continuación: [13]

- **Un marco común del proceso**, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia del tamaño o complejidad.
- **Un conjunto de tareas**, cada uno es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y los requisitos del equipo del proyecto.
- **Las actividades de protección**, tales como garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.

1.7.1 Modelos de proceso de desarrollo de software

Estos modelos son definidos como una representación simplificada de un proceso de desarrollo de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de proceso de desarrollo es una abstracción de un proceso real. Existen varios modelos de proceso de desarrollo de software como son Modelo en espiral, Modelo en cascada, Modelo de desarrollo por prototipo, Modelo de 4ta generación (RAD), Modelo basado en componentes, Modelo en V, los cuales fueron analizados según sus características.

- **Modelo en cascada** [14]

Fue el primer modelo de desarrollo de software que se publicó, fue propuesto por Royce en 1970 y fue adaptado para el software a partir de ciclos de vida de otras ramas de la ingeniería. Este utiliza las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso.

El modelo en cascada consta de las siguientes fases:

1. Definición de los requisitos: Los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.

2. Diseño de software: Se particiona el sistema en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.

3. Implementación y pruebas unitarias: Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.

4. Integración y pruebas del sistema: Se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.

5. Operación y mantenimiento: Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

La interacción entre fases puede observarse en la Figura # 1. Cada fase tiene como resultado documentos que deben ser aprobados por el usuario.
Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas. Este modelo admite la posibilidad de hacer iteraciones, es decir, durante las modificaciones que se hacen durante el mantenimiento se puede ver por ejemplo la necesidad de cambiar algo en el diseño, lo cual significa que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo las pruebas si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrer de nuevo el resto de las etapas.

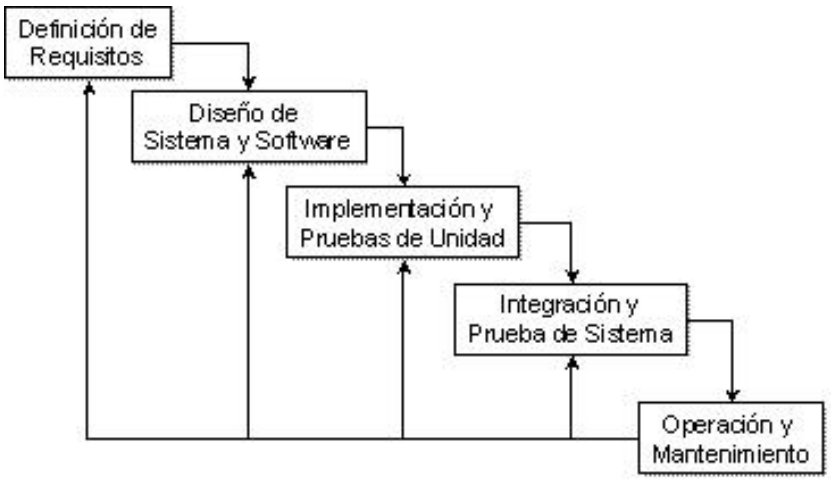


Figura # 1: Modelo en cascada.

Algunos problemas que se observan en el modelo cascada son:

- Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
- Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
- Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos. Es difícil responder a cambios en los requisitos.
- Es comparativamente más lento que los demás y el coste es mayor también.

Los tipos de proyectos para los cuales es adecuado utilizar este modelo son aquellos para los que se dispone de todas las especificaciones desde el principio.

Ventajas del modelo:

- Fácil de manejar los planes de proyectos
- Alto nivel de seguridad y fiabilidad.

Desventajas del modelo de cascada

- Usuarios no ven el sistema hasta las pruebas operacionales.
- Periodos de desarrollo largos.
- Dificultades en la respuesta a cambios en los requerimientos de usuario.

❖ **Modelo de desarrollo por prototipo** [15]

Este modelo se utiliza para dar al usuario una vista preliminar de parte del software. Este modelo es básicamente prueba y error ya que si al usuario no le gusta una parte del prototipo significa que la prueba falló por lo cual se debe corregir el error que se tenga hasta que el usuario quede satisfecho.

Ventajas del modelo:

- Buena comunicación con los clientes.
- Recomendado para proyectos de pequeño mediano alcance.

Desventajas del modelo:

- Planificaciones pudieran sufrir cambios.
- Dificultades para mantener el sistema en proyectos grandes.
- Periodos de desarrollo largos.
- Dificultades en la respuesta a cambios en los requerimientos de usuario.

Como otras desventajas tenemos que debido a que el usuario ve que el prototipo funciona piensa que este es el producto terminado y no entienden que recién se va a desarrollar el software. Otro problema es que el prototipo deber ir acompañado de otro modelo pasa su desarrollo

Hay dos clases de prototipos el **desechable y el evolucionario.**

El desechable nos sirve para eliminar dudas sobre lo que realmente quiere el cliente además para desarrollar la interfaz que más le convenga al cliente

El evolucionario es un modelo parcialmente construido que puede pasar de ser prototipo a ser software pero no tiene una buena documentación y calidad. Ver Figura # 2

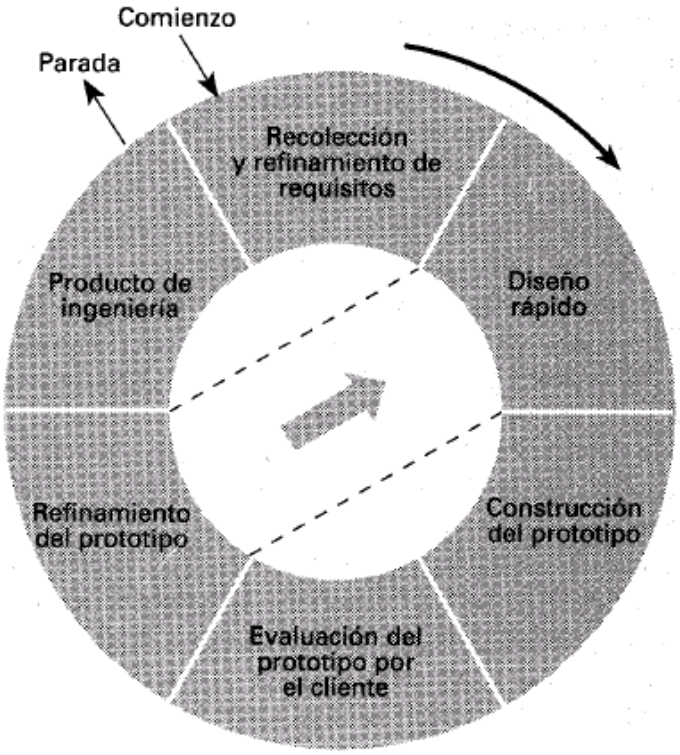


Figura # 2 Modelo prototipo

- **Modelo en espiral** [15] [16]

El modelo de desarrollo en espiral es actualmente uno de los más conocidos y fue propuesto por Boehm. El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra.

Cada ciclo de desarrollo se divide en cuatro fases:

1. Definición de objetivos: Se definen los objetivos. Se definen las restricciones del proceso y del producto. Se realiza un diseño detallado del plan administrativo. Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.
2. Evaluación y reducción de riesgos: Se realiza un análisis detallado de cada riesgo identificado. Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos. Se llevan a cabo los pasos para reducir los riesgos.
3. Desarrollo y validación: Se escoge el modelo de desarrollo después de la evaluación del riesgo. El modelo que se utilizará (cascada, sistemas formales, evolutivo, etc.) depende del riesgo identificado para esa fase.
4. Planificación: Se determina si continuar con otro ciclo. Se planea la siguiente fase del proyecto.

Este modelo a diferencia de los otros toma en consideración explícitamente el riesgo, esta es una actividad importante en la administración del proyecto.

El ciclo de vida inicia con la definición de los objetivos. De acuerdo a las restricciones se determinan distintas alternativas. Se evalúan los riesgos con actividades como análisis detallado, simulación, prototipos, etc. Se desarrolla un poco el sistema. Se planifica la siguiente fase. Ver Figura # 3

Fortalezas del modelo:

- El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora.
- Como el software evoluciona a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los nivele evolutivos.
- El modelo en espiral permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.
- El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas.
- En la utilización de grandes sistemas a doblado la productividad.

Debilidades del modelo:

- Es nuevo (1988) y no se ha utilizado tanto como otros.
- Debido a su elevada complejidad no se aconseja vida en donde los riesgos de diseño, implementación e utilizarlo en pequeños sistemas.
- Resulta difícil convencer a grandes clientes de que enfoque evolutivo es controlable.

Amenazas que presenta el modelo:

- Resolución tardía de riesgos: Uno de los principales problemas es la falta de previsión temprana de riesgos, esto se debe al enfoque en el diseño en papel al inicio del ciclo de vida en donde los riesgos de diseño, implementación e integración del sistema todavía son relativamente impredecibles.
- Inicio de la codificación muy tarde en el ciclo de vida del proyecto: Al enfocarse inicialmente en el diseño del papel (en ocasiones demasiado) se inicia la etapa de codificación tarde dando poco tiempo para solucionar problemas imprevistos en el diseño, lo que nos lleva a la entrega tardía de un sistema sin las pruebas óptimas.

Ventajas del modelo:

- Buena comunicación con los clientes a partir de que combina las ventajas de prototipo y cascada.
- Aproximación sistemática por pasos.
- Usuarios pueden revisar el sistema sistemáticamente.
- Recomendado para proyectos de gran alcance.

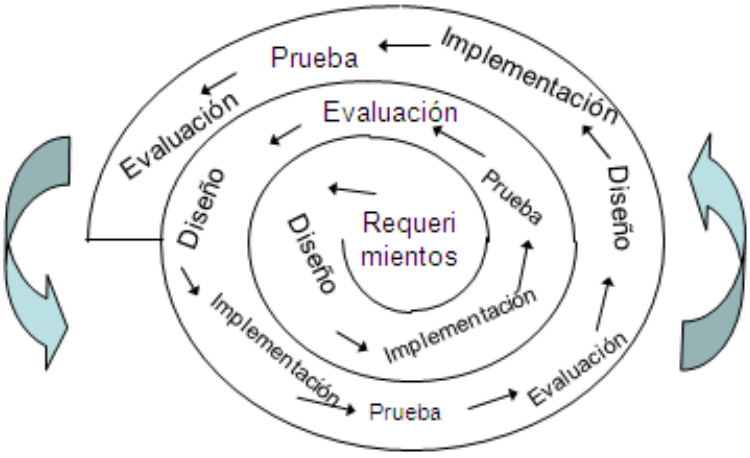


Figura # 3 Modelo en espiral

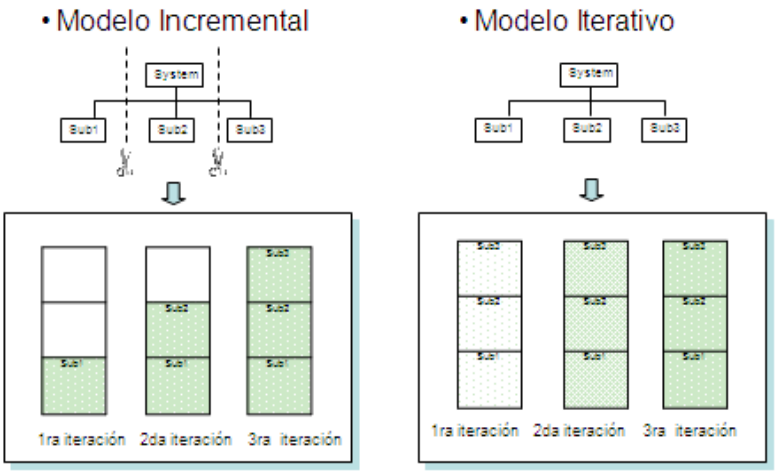


Figura # 4 Variaciones del desarrollo en espiral

- **Modelo de desarrollo basado en componentes** [19]

Incorpora características del Modelo Espiral. Se crean aplicaciones desde componentes separados del Software (clases). Esto permite reutilizar los componentes en futuras nuevas aplicaciones (reutilización de código). Este sistema es modular, lo que permite desarrollar futuras aplicaciones integrando módulos hechos anteriormente, junto con algunos nuevos construidos. Es evolutivo por naturaleza y exige un enfoque interactivo para la creación del software. Ver Figura # 5

El modelo de desarrollo basado en componentes conduce a la reutilización del software, y la reutilización proporciona beneficios a los ingenieros de software. La reutilización de software es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la especificación, análisis, diseño, implementación y pruebas de una aplicación o sistema de software.

Varios estudios han demostrado la efectividad de la reutilización del software, algunos de estos indicadores:

- Entre el 40 y 60% del código fuente de una aplicación es reutilizable en otra similar.
- Aproximadamente el 60% del diseño y del código de aplicaciones administrativas es reutilizable.
- Aproximadamente el 75% de las funciones son comunes a más de un programa.
- Sólo el 15% del código encontrado en muchos sistemas es único y novedoso a una aplicación específica. El rango general de uso recurrente potencial está entre el 15% y el 85%.

Beneficios del modelo: [15]

- Reutilización del software. Nos lleva a alcanzar un mayor nivel de reutilización de software.

- Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre componentes, el desabollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.
- Versiones fácilmente actualizables.

Desventajas del modelo: [15]

- Alto costo si la adaptación de nuevos paquetes es costosa.
- Dificultades para asegurar las versiones si los suministradores de componentes no mejoran continuamente sus productos.
- Genera mucho tiempo en el desarrollo del sistema.
- Modelo costoso.
- Requiere experiencia en la identificación de riesgos.
- Genera mucho trabajo adicional.
- Cuando un sistema falla se pierde tiempo y coste dentro de la empresa.
- Exige una cierta habilidad en los analistas (es bastante difícil).

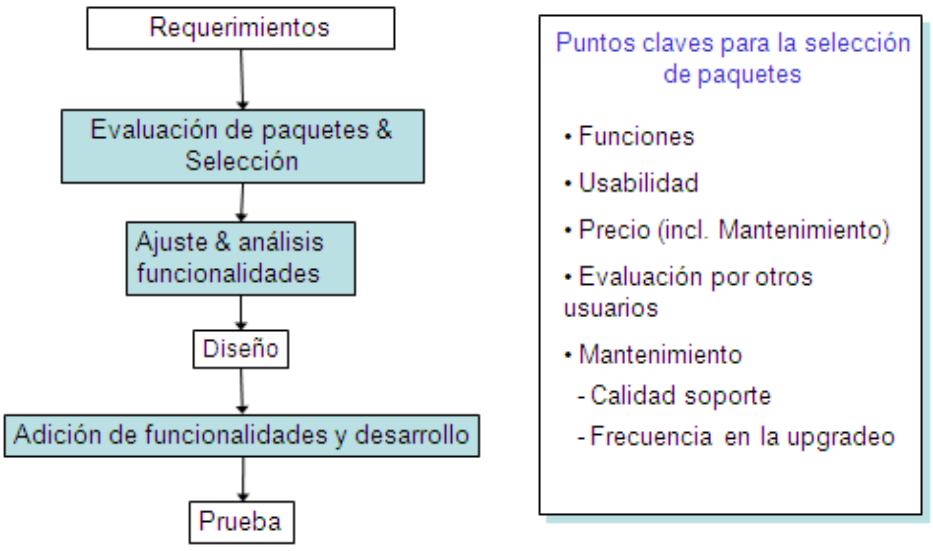


Figura # 5 Modelo de desarrollo basado en componentes

❖ **Modelo de 4ta generación RAD (Desarrollo Rápido de Aplicación)** [20]

El Desarrollo rápido de aplicaciones (o RAD) definido por James Martin a principios de la década de 1980, consiste en un ciclo de desarrollo corto basado en tres fases (Requisitos, Diseño y Construcción) con un plazo de entrega ideal de 90 a 120 días como máximo. En lugar de crear Software, el RAD reutiliza componentes de programas ya existentes o crea componentes reutilizables. Ver Figura # 6

Ventajas del modelo: [15]

- Buena comunicación con los clientes.
- Usuarios pueden revisar el sistema sistemáticamente.

Desventajas del modelo: [15]

- Dificultades para que los usuarios participen en todos las fases de desarrollo.
- Dificultades para asegurar equipo de alto nivel profesional.

Este modelo en parte puede ser muy bueno con respecto al ciclo de desarrollo que es bastante rápido pero no evidencia ningún proceso de calidad que asegure que ese producto tiene la calidad requerida.

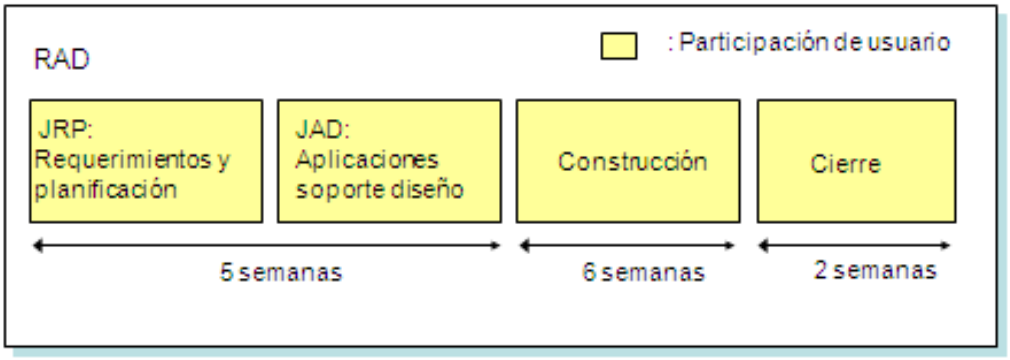


Figura # 6 Modelo RAD

1.7.2 Modelos de prueba para el soporte a la gestión de la calidad durante el ciclo de vida de un proyecto.

Existen varios modelos de prueba que de una manera u otra permiten organizar el proceso de gestión de la calidad del ciclo de vida de un proyecto utilizándolo como base y a partir de él desarrollar una estrategia de prueba que cubra todas las fases del proyecto asegurando en gran medida la calidad del mismo.

Un modelo de prueba es una vista de las actividades que ocurren durante el desarrollo de software referentes a la calidad para establecer el orden de las mismas y como se llevan a cabo en cada una de las fases. Estos modelos por una parte suministran una guía para los ingenieros de software con el fin de ordenar las diversas actividades técnicas en el proyecto, por otra parte suministran un marco para la administración del desarrollo y el mantenimiento del software.

- **Modelo en V**

El modelo V es un modelo de prueba que tiene la facilidad de que en el momento en el cual se está realizando una fase es posible realizar la documentación para las pruebas que se van a realizar. Permite combinar los tipos de pruebas con actividades de verificación y validación sobre los documentos de requisitos y diseño que optimizan la capacidad de aceptar el producto resultante. [14] [17]

- **La verificación** se fundamenta en demostrar si el producto se está construyendo correctamente.
- **La validación** certifica que el producto cumple con las exigencias definidas por el cliente.

Puede notarse que su primera mitad son las fases en que se divide el ciclo de vida, y la otra mitad tiene como finalidad hacer las pruebas asociadas a cada una de las etapas de la mitad anterior. Este modelo demuestra las relaciones entre cada fase del ciclo vital de desarrollo y su fase de pruebas. Definitivamente se trata de un modelo más robusto y completo produciendo así software de mayor calidad. Ver Figura # 7.

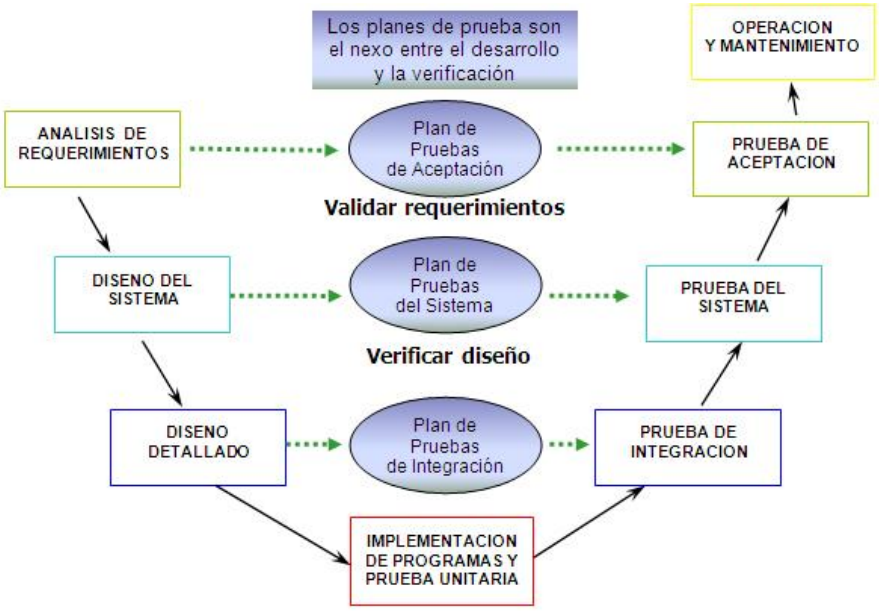


Figura # 7 Modelo en V

Partiendo de los requisitos (a la izquierda) se deben planificar y preparar los niveles de pruebas correspondientes (a la derecha). En general, cada actividad de prueba a la derecha valida la actividad enfrentada de la izquierda. Los niveles superiores de pruebas en el diagrama están basados en caja negra (pruebas basadas en las especificaciones) y los niveles inferiores están basados en caja blanca (basadas en la estructura interna de los componentes del sistema). Este modelo sigue la estrategia clásica para las pruebas de software se comienza con probar de lo pequeño hasta lo más grande, por lo que se comienza con la prueba de unidad, luego se avanza con la prueba de integración, después con las pruebas del sistema y por ultimo con las pruebas de aceptación del cliente. Cada organización puede utilizar su versión del Modelo en V, basándose en su propia terminología.

- **Modelo en W** [17]

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

- En las primeras etapas se consideran labores de 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 y del diseño de detalle y las revisiones de código.

- En las etapas finales también se distingue del modelo original en V porque se desglosan las tareas de pruebas propiamente dichas y las labores de depuración y corrección de los errores detectados, que serán llevadas a cabo por desarrolladores. Ver Figura # 8

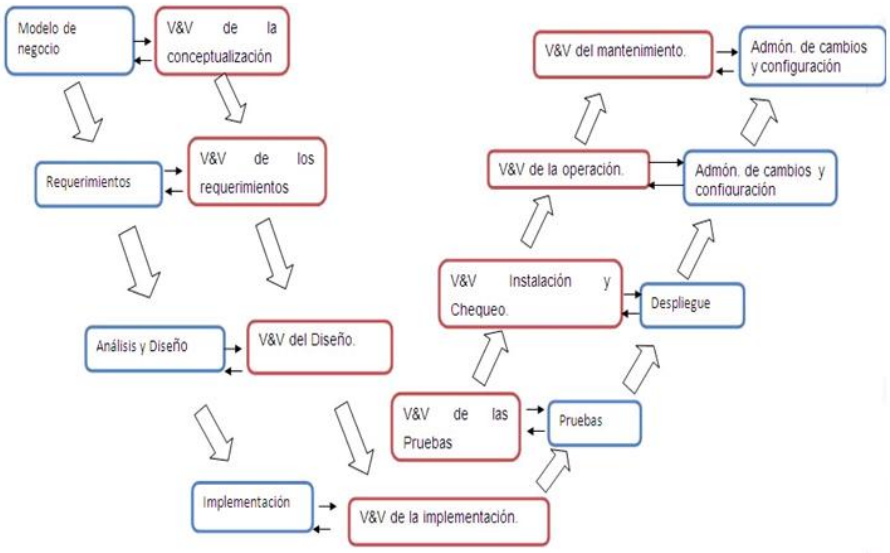


Figura # 8 Modelo W

Según el gráfico del modelo en W, las labores de pruebas aparecen en los bloques rojos. Los bloques azules representan el resto de actividades del ciclo de vida del software antes de su puesta en producción. En las primeras etapas se concentran labores de verificación de productos no software y de preparación para la verificación y validación del software. En las etapas finales se desarrolla todo el esfuerzo de verificación y validación del software producido.

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

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

- ✓ Pruebas sobre el software
  - Elaboración de las pruebas de aceptación
  - Elaboración de las pruebas de sistema
  - Elaboración de las pruebas de integración de subsistemas

- Elaboración de las pruebas unitarias
- Revisiones del código
- Ejecución de las pruebas
- Depuración e introducción de cambios

Pruebas de Verificación y Validación de productos no software.

Estas pruebas cubren la mayor parte de productos en forma de documentos, dependiendo de las características del proyecto podrán inspeccionarse mayor o menor número de productos "No Software".

Se basan fundamentalmente en revisiones por iguales y en las que no necesariamente debe intervenir personal dedicado a pruebas. Los documentos se someten a un proceso de revisión establecido en el plan de calidad del proyecto, donde se define quien lo revisa (normalmente personal con al menos el mismo nivel técnico que el autor del documento y eventualmente el cliente), como se corrigen los defectos y como finalmente se aprueba el documento para su uso en las siguientes etapas del ciclo de vida.

- **Modelo Mariposa** [17]

Este modelo se centra en la verificación y validación de productos software por lo que se adapta perfectamente a las tareas de pruebas del software que aparecen reflejadas tanto en el modelo en V como en el modelo en W. Va más al detalle y muestra aún más en detalle que es lo que hace el modelo W, la complejidad de la interacción entre los equipos de desarrollo y pruebas. Ver Figura # 9

El modelo da una visión de la complejidad de las tareas desarrolladas de manera gráfica. Para ello emplea una mariposa en la que las áreas ocupadas por las alas y el cuerpo mantienen una relación aproximada con el esfuerzo que se dedica a cada una de las actividades recogidas por el modelo.

El porqué de una mariposa tiene su origen en la teoría del Caos una pequeña perturbación en una parte de un sistema (vuelo de mariposa en Asia) origina una gran perturbación en otra parte del mismo sistema (huracán en Europa) El desarrollo de un sistema software tiene ciertas similitudes. Pequeñas modificaciones o errores introducidos en el código pueden llevar a comportamientos absolutamente anormales.

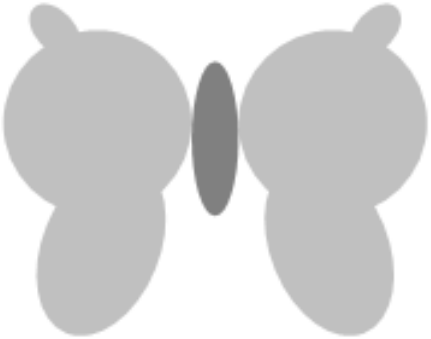


Figura # 9 Modelo mariposa

El modelo establece tres grupos de actividades que se recogen en el plan de pruebas y que desde el punto de vista de envergadura de las mismas aparece reflejada en el dibujo por áreas siendo las siguientes:

- Análisis de Pruebas (Ala izquierda de la mariposa)
- Diseño de Pruebas (Ala derecha)
- Ejecución de Pruebas (Cuerpo de la mariposa)

1.8 Conclusiones

En este capítulo se realizo un estudio de algunos aspectos referentes a la calidad de software en general, recopilando ideas y conceptos de diferentes bibliografías y autores que ayudaron en gran medida a ampliar los conocimientos acerca del tema. Se analizaron algunos modelos de procesos de desarrollo de software y modelos de pruebas que se utilizaron en la adquisición de conocimiento para el desarrollo de la propuesta. El proceso de desarrollo del CENTALAD es basado en componentes, debido a esto se decide utilizar el modelo de desarrollo basado en componentes en unión al modelo V como modelo de pruebas a modo de adaptación para llevar a cabo la propuesta, introduciendo de esta manera buenas prácticas de Calidad Total en la gestión de la calidad de los proyectos productivos del Centro. Fue seleccionado el Modelo V porque en estructura cubre en todo momento con las perspectivas que se persigue con la propuesta, define las pruebas a realizar luego de la obtención del producto y te permite revisar la documentación por cada una de las fases, favoreciendo esto a la reducción de errores al finalizar el producto. De esta manera se puede proceder al desarrollo de la propuesta que dará solución al problema planteado por lo que en el capitulo siguiente se abordara sobre la estructura que poseerá misma.

CAPÍTULO 2: DESARROLLO DE LA PROPUESTA

2.1 Introducción

“No es el conocimiento el que nos hace triunfar, sino la aplicación de él.”

En el presente capítulo se plasma la estructura y principales componentes de la guía propuesta, tomando como base el procedimiento establecido por la Dirección Central de Calidad UCI en relación a la organización del proceso de pruebas, se ha modificado y adaptado a las necesidades del Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD). El Modelo basado en Componentes fue escogido como modelo de proceso de desarrollo a seguir con una adaptación del Modelo V, con el fin de proponer una guía que disponga de buenas prácticas de Calidad Total abarcando desde la especificación de requisitos hasta la liberación del software, y de esta forma lograr mejores resultados en el proceso de pruebas con el aumento en la detección de No Conformidades.

2.2 Modelo en V definido por CENTALAD

El Modelo en V como modelo de prueba fue el seleccionado para desarrollar la guía propuesta ya que sus características se ajustan a los intereses del Centro. Debido a que puede ser modificable por cualquier empresa que lo vaya a utilizar, en nuestro caso el Modelo en V definido quedo de la siguiente manera: Ver Figura # 10

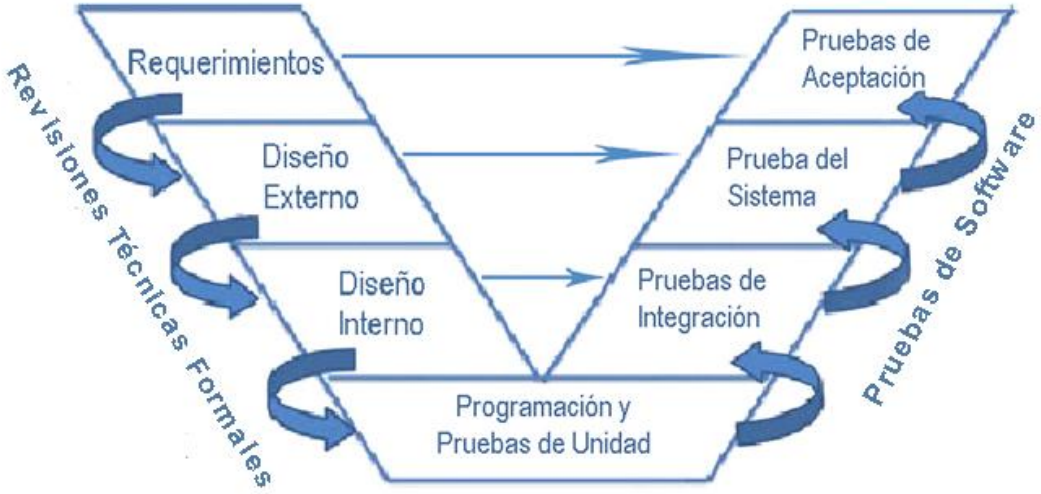


Figura # 10 Modelo en V definido por CENTALAD

- ✓ **Requerimientos:** está orientado al "cliente", es donde se comprende lo que desean los clientes, se recopilan, examinan y formulan los requisitos del cliente y cualquier restricción que se pueda aplicar**.** La documentación que es generada en esta etapa, es todo lo referente a especificación de requerimientos donde se encuentran definidos todos los servicios requeridos del sistema y las restricciones sobre las que debe operar. Esta fase se relaciona con la fase de *Requerimientos* del modelo basado en componentes.
- ✓ **Diseño Externo:** es donde se especifican las características funcionales del sistema propuesto. Es más bien un diseño del exterior del software donde se realiza la descripción de los casos de usos con sus respectivos prototipos de interfaz para una mejor comprensión de lo que se va a desarrollar y se construye la arquitectura que cumpla con las necesidades del sistema. Esta fase se vincula a las fases *Evaluación de paquetes & Selección* y *Ajuste & análisis funcionalidades* del modelo basado en componentes
- ✓ **Diseño Interno:** es donde se definen los componentes del diseño mas detallados, la estructura de la integración de los módulos, como una definición precisa de cada subconjunto de la aplicación. Es como un diseño del exterior del software donde se desglosa con más detalle cada actividad o procedimiento. Esta fase se enlaza a la fase de *Diseño* del modelo basado en componentes.
- ✓ **Programación:** en la que se desarrollan los módulos del programa. Se lleva a cabo la codificación del sistema, lo que especifica el diseño debe satisfacer los requerimientos. El programador examina los documentos generados en las etapas anteriores para evitar inconsistencias entre los documentos y el procedimiento a realizar, tomando en cuenta los estándares y lenguajes de programación. Esta fase de relaciona con la fase de *Adición de funcionalidades y desarrollo* del modelo basado en componentes.

2.3 Organización del proceso de desarrollo en el CENTALAD

El Centro actualmente cuenta con una estructura organizativa basada en Líneas de Productos de Software (LPS). Cada Línea está compuesta por grupos de trabajos que son los encargados de desarrollar los productos de software.

2.3.1 ¿Qué es una Línea de Productos de Software (LPS)? [18]

La idea básica:

- Ensamblaje de partes de software previamente elaboradas.
- Inspirada en los procesos de producción de sistemas físicos.
- Producción de aviones, vehículos, computadores, aparatos electrónicos, etc.
- Fundamentada en la Reutilización de Software.
- Asume la existencia de una industria de partes.

Beneficios de los LPS

- La entrega de productos de software de una manera más rápida, económica y con una mejor calidad.
- Las LPS producen mejoras en:
  - Tiempo de entrega del producto
  - Costos de ingeniería
  - Tamaño del portafolio de productos
  - Reducción de las tasas de defectos
  - Calidad de los productos
- Beneficios tácticos y estratégicos:
  - Beneficios tácticos de ingeniería:
    - Reducción en el tiempo promedio de creación y entrega de nuevos productos
    - Reducción en el número promedio de defectos por producto
    - Reducción en el esfuerzo promedio requerido para desarrollar y mantener los productos
  - Beneficios estratégicos de negocios
    - Reducción en el tiempo de entrega y el tiempo de retorno de nuevos productos.
    - Mejoras en el valor competitivo del producto.
    - Mejor calidad de los productos.
    - Reducción de riesgos en la entrega de productos.

Aspectos conceptuales de los LPS

1. **La reutilización de activos de software** en LPS tiene varias características:
   - Es *estratégica*
     - Consolida lo común entre la línea de productos.
     - Maneja estratégicamente la variación entre los productos de la línea.

- Elimina la duplicación de esfuerzos de ingeniería.

- Es *predictiva*
  - La reutilización de activos se da en uno o más productos sobre una línea bien definida.
  - Se reutilizan arquitecturas de software, en lugar de reutilizar componentes de manera oportunista.
- Es *gestionada*

- Es sistemática, planificada, institucionalizada y mejorada.

2. Un **activo de software reutilizable** es un producto de software diseñado expresamente para ser utilizado múltiples veces en el desarrollo de diferentes sistemas o aplicaciones.

   Un activo de software puede ser:
   - ✓ Un componente de software
   - ✓ Una especificación de requisitos
   - ✓ Un modelo de negocios
   - ✓ Una especificación de diseño
   - ✓ Un algoritmo
   - ✓ Un patrón de diseño
   - ✓ Una arquitectura de dominio
   - ✓ Un esquema de base de datos
   - ✓ Una especificación de prueba
   - ✓ La documentación de un sistema
   - ✓ Un plan

3. Un **componente de software reutilizable (CSR)** es:
   - ✓ Una pieza de software funcional que es liberada independientemente de otras y que proporciona acceso a sus servicios a través de sus interfaces.
   - ✓ Puede ser liberado (desplegado e instanciado) independientemente de otros:
   - ✓ Ofrece servicios a través de sus interfaces
   - ✓ Para utilizar su funcionalidad se emplean sus interfaces

Características esenciales de los CSR:

- ✓ Identificable
- ✓ Autocontenido
- ✓ Rastreable a través de su ciclo de desarrollo
- ✓ Reemplazable por otro componente
- ✓ Accesible solamente a través de su interfaz
- ✓ Documentación de sus servicios

Tipos de CSR:

- ❖ Según su modificabilidad
  - Caja negra
  - Caja blanca
- ❖ Según su granularidad
  - Componentes de uso específico
  - Componentes de negocio
  - Marcos (*frameworks*)
  - Componentes de aplicación
- ❖ Según su fabricante
  - Componentes hechos en casa
  - COTS – Component Off The Shelf
- ❖ Según la tecnología usada
  - Componentes imperativos
    - Módulos, funciones
  - Componentes OO
    - Clases
  - Componentes distribuidos
    - Componentes CORBA
    - Componentes .NET
    - Componentes J2EE
    - Servicios Web

2.3.2 Líneas definidas por CENTALAD:

- Línea de Soluciones Integrales de Almacenes de Datos
  - ✓ Almacenes
  - ✓ Análisis
  - ✓ BI
  - ✓ ETL
- Línea de Herramientas de Procesamiento y Análisis de Datos
  - ✓ Análisis
  - ✓ Arquitectura
  - ✓ Desarrollo
- Línea de Desarrollo de Tecnologías de base de Datos
  - ✓ Administración de Servicios
  - ✓ Análisis y Arquitectura
  - ✓ Servidores de Migración

2.3.3 Organización del proceso de desarrollo en el Proyecto

La guía que se propone para ser aplicada en los proyectos que se desarrollan en CENTALAD a grandes rasgos consiste en llevar a cabo el proceso de pruebas paralelo al ciclo de vida del software.

❖ **Estrategia Propuesta**

Se definieron un grupo de actividades de calidad de las cuales se desglosaron varias tareas a realizar que favorecen a la organización del proceso de revisión a seguir. Ver Tabla # 2

Modelo en V	Actividades de calidad	Tareas de calidad a realizar
Requerimiento	RTF de los Requerimientos.	1. Revisión de la Documentación 2. Revisiones de los requerimientos de software.
Diseño Externo	RTF del Diseño Externo	1. Revisión de la documentación. 2. Análisis de la interfaz. 3. Revisiones de la arquitectura.
Diseño Interno	RTF del Diseño Interno	1. Revisión de la documentación.
Programación	RTF de la Programación	1. Revisión de la documentación

Tabla # 2: Proceso, Actividades y Tareas de calidad

Esta tabla muestra primeramente los niveles que identifican las fases del ciclo de vida del software definidos por el modelo en V, luego las actividades de calidad que se van a realizar en cada fase y las tareas de calidad que indican cómo se realizarán estas actividades durante el ciclo de vida. Para la identificación y descripción de las tareas se sigue una numeración, ganando así en organización. Es importante explicar que se debe consultar la tarea sobre la cual se esté trabajando pues antes de comenzar una tarea es necesario conocer las entradas de la misma y los productos a generar que constituyen la salida de dicha tarea.

- **Roles y responsabilidades**

Gerente de calidad:

- Asegura que la aplicación producida se ajusta a las especificaciones y está razonablemente libre de errores.
- Proporciona una estrategia de prueba.
- Coordina las pruebas de calidad interna, las pruebas de liberación y aceptación.
- Evalúa los resultados de las pruebas de calidad.
- Rige y dirige las actividades de aseguramiento y control de la calidad

Desarrollador de software:

- Diseña los casos de pruebas.
- Ejecuta las pruebas de unidad e integración.
- Ejecuta las RTF de su documentación en unión con el revisor técnico.

Jefe de Prueba:

- Diseña los casos de prueba.
- Define las listas de Chequeo.
- Evalúa y documenta el resultado de las pruebas realizadas al software.
- Realiza el plan de prueba.

Revisor Técnico:

- Ejecuta las RTF
- Elabora el informe de las RTF.

Probador (tester):

- Ejecuta las pruebas diseñadas.
- Elabora el documento de no conformidades.

❖ **Pasos a seguir por cada una de las actividades de calidad:**

RTF de los Requerimientos

➢ Tareas a realizar por el revisor técnico, el desarrollador de software y el jefe de prueba.

1. Revisión de la Documentación

   Entradas

   - Evaluación de Áreas de la Organización.
   - Historias de usuario.
   - Plan de Desarrollo de Software.
   - Plan de Mitigación de Riesgos.
   - Ambiente de desarrollo.
   - Plan de capacitación.
   - Roles y responsabilidades.
   - Documento Visión.

   Salidas

   - Listas de chequeo aplicadas
   - Documentos de No Conformidades
   - Plan de aseguramiento de la calidad

2. Revisiones de los requerimientos de software

   Entradas

   - Especificación de requisitos de software
   - Plan de gestión de requisitos

   Salida

   - Listas de chequeo aplicadas
   - Documentos de No Conformidades
   - Plan de pruebas de aceptación

RTF del Diseño Externo

- Tareas a realizar por el revisor técnico, el desarrollador de software y el jefe de prueba.

1. Revisión de la documentación.

   Entradas

   - Modelo del Negocio o Modelo del Dominio
   - Modelo del Sistema

   Salidas

   - Listas de chequeo aplicadas
   - Documentos de No Conformidades
   - Diseño de casos de pruebas.
   - Plan de pruebas del sistema.

2. Análisis de la interfaz.

   Entradas

   - Modelo del Sistema

   Salidas

   - Lista de chequeo aplicada
   - Documento de No Conformidades

3. Revisiones de la arquitectura.

   Entradas

   - Arquitectura de Información
   - Documento de Arquitectura de Software
   - Informe del Levantamiento de Información para la Arquitectura de Información
   - Modelo de despliegue

   Salidas

   - Listas de chequeo aplicadas
   - Documentos de No Conformidades

RTF del Diseño Interno

- Tareas a realizar por el revisor técnico, el desarrollador de software y el jefe de prueba.

1. Revisión de la documentación.

   Entradas

   - Modelo de Diseño

Salidas

- Lista de chequeo aplicada
- Documento de No Conformidades
- Diseño de casos de pruebas
- Plan de pruebas de integración

RTF de la Programación

- Tareas a realizar revisor técnico, el desarrollador de software y el jefe de prueba.

1. Revisión de la documentación.

Entradas

- Manual de Usuario

Salidas

- Lista de chequeo aplicada
- Documento de No Conformidades
- Diseño de casos de pruebas
- Plan de pruebas de unidad

Luego de la obtención del producto real se comienza con la realización de las pruebas internas que nos ayudarán a encontrar los errores que pueda presentar dicho producto. Como todo proceso de pruebas se comienza desde la más mínima que son las pruebas de unidad, se procede con las de integración de los módulos detectando así cualquier error de funcionalidad del sistema, seguidamente se realizan las pruebas de sistema para verificar el diseño detalladamente y por último las pruebas de aceptación para validar los requerimientos especificados por el cliente.

- **Procedimiento de las pruebas a realizar:**

Pruebas de unidad

- Para detectar errores en lógica, datos o algoritmos, en componentes o subsistemas individuales.
- Realizadas por el desarrollador del componente y un integrante del grupo de calidad para registrar las No Conformidades.
- Se utilizan técnicas de caja blanca.

Tareas a realizar por el desarrollador de software y el probador

Entradas

- Plan de pruebas de unidad
- Diseño de casos de pruebas
- Código fuente

Salidas

- Documento de No Conformidades

Herramienta

- Herramientas automatizadas

Pruebas de integración

- Se utilizara la integración incremental del tipo ascendente.
- Estas pruebas son para detectar errores en la integración de los componentes que ya fueron probados en las pruebas de unidad.
- Realizadas por los desarrolladores de los componentes que se integran y un integrante del grupo de calidad para registrar las no conformidades.
- Se utilizan técnicas de caja negra y caja blanca.

Tareas a realizar por el desarrollador de software y el probador

Entradas

- Plan de pruebas de integración
- Diseño de casos de pruebas
- Código fuente

Salidas

- Documento de No Conformidades

Herramienta

- Herramientas automatizadas

Pruebas del sistema

- Para detectar errores en comportamiento con respecto a especificación de requerimientos.
- Realizadas por el grupo de calidad y algunos desarrolladores.
- Se utilizan técnicas de caja negra basadas en especificación de requisitos y casos de uso.
- Dentro de las pruebas de sistemas se realizaran las siguientes:
  - ✓ Pruebas de Funcionalidad: para verificar la función del sistema, para validar que el sistema cumpla con los requisitos funcionales y no funcionales especificados en el diseño de la solución.

- ✓ Pruebas de Rendimiento o Carga: para probar el rendimiento del software en tiempo de ejecución, evalúan la aceptabilidad de un elemento de un sistema sobre diferentes cargas de trabajo.
- ✓ Pruebas de Seguridad: para verificar los mecanismos de protección.
- ✓ Pruebas de Volumen: Verifican que la aplicación funcione adecuadamente bajo los siguientes escenarios de volumen:
  - Determinan la cantidad de datos con la cual el sistema falla.
  - Carga máxima que el sistema soporta en un período dado.

Técnicas que utiliza:

1. Usar múltiples clientes, corriendo las mismas pruebas o pruebas complementarias para producir el peor caso de volumen por un período extendido.

2. Utilizar un tamaño máximo de Base de datos (actual o con datos representativos) y múltiples clientes para correr consultas simultáneamente para períodos extendidos.

- ✓ Pruebas de Compatibilidad: para verificar el funcionamiento del sistema sobre diferentes componentes de software y verificar la compatibilidad de la aplicación con sistemas operativos y navegadores.
- ✓ Pruebas de Estrés: esta es una de las más importantes porque se refieren a cargas extremas, memoria insuficiente, no disponibilidad de servicios y hardware o recursos compartidos limitados.

- Tareas a realizar por el probador y un desarrollador de software como observador

Entradas

- Plan de pruebas del sistema
- Diseño de casos de pruebas
- Producto de Software
- Base de Datos

Salidas

- Documento de No Conformidades

Herramienta

- Herramientas automatizadas

- **Pruebas de aceptación**
  - Son básicamente pruebas sobre el sistema completo, para detectar errores en comportamiento con respecto a especificación de requerimientos.
  - Realizadas por el cliente.
  - Se buscan inconsistencias entre la especificación de requisitos y el manual del usuario.
  - Dentro de las pruebas de aceptación se realizaran las siguientes:
  - ✓ Pruebas Alfa: son llevadas a cabo por el cliente, en el lugar de desarrollo, usando el software de forma natural con el desarrollador como observador.
  - ✓ Pruebas Beta: son realizadas por los usuarios finales del software en los lugares de trabajo, sin la presencia del desarrollador.
  - Tareas a realizar por el cliente y un desarrollador de software como observador

  Entradas
    - Plan de pruebas del aceptación
    - Manual de Usuario
    - Especificación de requisitos de software
    - Producto de Software
    - Base de Datos

  Salidas
    - Documento de No Conformidades

2.3.4 Pasos que guiarán el proceso de pruebas en el proyecto:

Paso 1: El jefe de Línea le asigna la tarea a los grupos de realizar un producto.

Paso 2: Cada grupo ejecuta la tarea asignada elaborando la documentación y los activos de software, realizando posteriormente las revisiones correspondientes.

Paso 3: Cada grupo libera su documentación y le entrega el expediente de proyecto al jefe de Línea.

Dentro de este paso se realiza:

- RTF de los Requerimientos
- RTF del Diseño Externo
- RTF del Diseño Interno
- RTF de la Programación
- Pruebas de Unidad

Es importante destacar que cada grupo se especializa en una de las actividades de calidad para llevar a cabo estas revisiones, es decir un grupo se encarga de hacer las RTF de los Requerimientos, otro realiza las RTF del Diseño Externo y así sucesivamente, cubriendo de tal manera todas las actividades que fueron definidas.

Personal que participa:

- Se crea un pequeño grupo de liberación donde se encuentra el desarrollador, pero participan otros miembros del grupo y algún miembro del grupo de calidad en caso necesario.

Paso 4: El jefe de Línea se encarga de entregarle al grupo de calidad el producto para que realice las pruebas previas a la liberación.

Dentro de este paso se realiza:

- Pruebas de Integración
- Pruebas de Sistema

Personal que participa:

- Grupo de calidad para revisar nuevamente la documentación luego de la revisión realizada por los grupos para detectar si existen más No Conformidades y realizar las pruebas internas al producto.
- Algún desarrollador

Paso 5: El grupo de calidad le solicita a Calidad UCI la liberación del producto.

Dentro de este paso se realiza:

- Pruebas de Sistema

Personal que participa:

- Calidad UCI

Paso 6: Se realizan las pruebas de aceptación con el cliente.

Dentro de este paso se realiza:

- Pruebas de Aceptación

Personal que participa:

- El Cliente con la presencia del desarrollador como observador.

2.4 Conclusiones

Las pruebas forman parte del ciclo de vida del software y se deben realizar a lo largo del desarrollo del mismo, por lo que es de gran importancia contar con una estructura organizativa para llevarla a cabo, que tenga plasmada las actividades a ejecutar que sirvan de guía para la realización de este proceso, evitando que el software sea rechazado al final del ciclo de desarrollo. La guía propuesta contiene los diferentes tipos de pruebas que miden los objetivos y metas fundamentales para obtener así resultados satisfactorios con mejor precisión, cuenta con una estructura que cubre todas las fases en las cuales esta divido el proceso de desarrollo logrando de esta manera una revisión estricta de cada paso que se realice para lograr aumentar la detección de No Conformidades para garantizar así la calidad final del producto.

CAPÍTULO 3: EJECUCIÓN Y ANÁLISIS DE LOS RESULTADOS OBTENIDOS

3.1 Introducción

Este capítulo se centra en la aplicación de la guía propuesta en algunos de los proyectos que se desarrollan en CENTALAD como ONE y Generador de Reportes (GR), en el diseño y aplicación de las actividades definidas para cada fase y en el registro y análisis de los resultados obtenidos después de su aplicación mediante estadísticas que lo reflejan. Dentro de las actividades tenemos la realización del Plan de Pruebas, el Plan de Aseguramiento de la Calidad del proyecto al cual fue aplicado dicha propuesta, revisiones técnicas formales (RTF) mediante el documento de No Conformidades para registrar los defectos encontrados, la aplicación de Listas de Chequeo que se definieron para comparar el resultado de su aplicación con el obtenido mediante el documento de No Conformidades para determinar cual vía es más efectiva, entre otras actividades de control. La corrección de los defectos posibilita el aumento de la fiabilidad del software dando una medida de la calidad del mismo y su correspondencia con los requerimientos establecidos.

3.2 Aplicación de la guía propuesta en el proyecto ONE

El proyecto ONE pertenece a la Línea de Soluciones Integrales de Almacenes de Datos la cual está compuesta por 4 grupos de trabajo, todos los proyectos que pertenecen a la misma están compuestos por estos grupos:

- ✓ Almacenes
- ✓ Análisis
- ✓ BI (Inteligencia de Negocio)
- ✓ ETL (Extracción, Transformación y Carga)

Se le comenzaron a realizar las RTF a la documentación utilizando para esto el documento de No Conformidades para registrar los defectos encontrados en cada uno de los documentos. La guía propuesta está basada en los principios de calidad total la cual es una alusión a la mejora continua siendo esto un aspecto fundamental para todos los proyectos que lo apliquen. Para contribuir con la mejora continua del proceso de pruebas se conformaron listas de chequeo a documentos que no las tenían definidas para poder llevar a cabo las RTF mediante las mismas y hacer una comparación de

los resultados de su utilización con respecto a los que se obtuvieron mediante el documento de No Conformidades y así verificar cual de las dos vías es más efectiva a la hora de detectar mas No Conformidades.

- **Ejecución de los pasos definidos para las pruebas**

Paso 1: El jefe de Línea le asigna la tarea a los grupos de realizar el proyecto ONE.

Paso 2: Cada grupo ejecuta la tarea asignada elaborando la documentación y los activos de software, realizando posteriormente las revisiones correspondientes.

Paso 3: Cada grupo libera su documentación y le entrega el expediente de proyecto al jefe de Línea.

Dentro de este paso se realiza:

- **RTF de los Requerimientos**

1. Revisión de la Documentación

Entradas

- Resumen Adicional de Entrevistas
- Especificación del Negocio
- Diseño Conceptual
- Especificación de Reglas del Negocio
- Ambiente de desarrollo ONE
- Evaluación de Áreas de la Organización
- Plan de mitigación de riesgos
- Roles y responsabilidades
- Plan de capacitación
- Documento Visión
- Diccionario de Datos

Salidas

- Documento de No Conformidades
- Plan de aseguramiento de la calidad

2. Revisiones de los requerimientos de software

Entradas

- Especificación de Requisitos
- Plan de Gestión de Requisitos

Salida

- Documento de No Conformidades
- Plan de pruebas de aceptación.

- **RTF del Diseño Externo**

1. Revisión de la documentación.

   Entradas

   - Especificaciones del Diseño Físico
   - Bus Matrix-1
   - Especificación de Dimensiones
   - Especificación de Tablas de Hecho
   - Diseño de Casos de Uso del Sistema

   Salidas

   - Documento de No Conformidades.
   - Plan de pruebas del sistema.

2. Análisis de la interfaz.

   Entradas

   - Diseño de Casos de Uso del Sistema

   Salidas

   - Documento de No Conformidades

3. Revisiones de la arquitectura.

   Entradas

   - Informe del Levantamiento de Información para la Arquitectura de Información
   - Arquitectura de Información
   - Arquitectura de Software

   Salidas

   - Documento de No Conformidades.

Personal que participa:

Cada grupo realiza su actividad de calidad correspondiente y miembros del grupo de calidad, en este caso quedaría así:

Grupos	Actividades de Calidad
Almacenes	RTF de los Requerimientos
Análisis	RTF del Diseño Externo

3.2.1 Resultados obtenidos en el proyecto ONE

Se realizo un cierre en el momento donde se encontraban las revisiones para hacer un análisis de los resultados obtenidos hasta ese momento con la aplicación del documento de No Conformidades. Luego se definieron Listas de Chequeo que fueron aplicadas a esos mismos documentos para establecer una comparación entre los resultados anteriores y los de la aplicación de las listas y así verificar cual de las dos vías es más efectiva a la hora de detectar mayor cantidad de No conformidades y además la que asegure en gran medida que el documento este estructurado según corresponde. En las siguientes tablas se muestran los resultados correspondientes. Ver Tabla # 3 y 4.

	1era Iteración				2era Iteración			
Documentos	Cant. NC	Cant. NC-NP	Cant. NC-RA	Cant. NC-PD	Cant. NC	Cant. NC-NP	Cant. NC-RA	Cant. NC-PD
Bus Matriz.	1	0	1	0	0	0	0	0
Especificación de dimensiones.	11	0	7	4	5	0	5	0
Especificación de tablas de hechos.	6	0	6	0	0	0	0	0
Especificación del diseño físico.	6	0	5	1	1	0	1	0
Diseño Conceptual	4	0	4	0	0	0	0	0
Evaluación de las áreas de la organización	1	0	1	0	0	0	0	0
Resumen adicional de entrevistas	11	0	9	2	4	0	4	0
Especificación de Requisitos	7	0	3	4	5	0	5	0
Plan de Gestión de Requisitos	3	0	3	0	0	0	0	0
Diccionario de Datos	0	0	0	0	0	0	0	0
Especificación Registro de Sistemas de Fuentes	7	0	7	0	0	0	0	0
Especificación Reglas del Negocio	6	0	5	1	1	0	1	0
Especificación del Negocio	9	0	6	3	3	0	3	0
Mapa Lógico de Datos	1	0	1	0	0	0	0	0
Arquitectura de la Información	6	0	4	2	8	0	8	0
Informe del Levantamiento de Información	3	0	2	1	2	0	2	0
Acta de Constitución del proyecto.	7	0	4	3	3	0	3	0
Documento Visión.	1	0	1	0	0	0	0	0
Ambiente de desarrollo.	5	0	5	0	0	0	0	0
Plan de mitigación de riesgos.	3	0	2	1	1	0	1	0
Roles y responsabilidades.	1	0	1	0	0	0	0	0
Plan de capacitación.	0	0	0	0	0	0	0	0
Total	99	0	77	22	33	0	33	0

Figura # 3 Estadística general de las No Conformidades por iteraciones

Se escogieron algunos de los documentos que fueron revisados anteriormente con el documento de No Conformidades y se le aplicaron las Listas de Chequeo para observar los resultados y establecer una comparación. En las siguientes tablas están los resultados obtenidos: Ver Tablas # 4 y 5.

Revisiones por Listas de Chequeo					
Documentos	Cant Indicadores	Indicadores eval- mal	Cant Críticos	Críticos eval- mal	Cant NC
Arquitectura de Información	21	19	11	9	7
Bus Matrix-1	10	2	6	0	1
Especificación de dimensiones	9	3	6	2	11
Roles y responsabilidades	11	4	6	1	1
Ambiente de desarrollo	12	4	5	2	5
Documento Visión	15	0	3	0	1
Plan de capacitación	13	1	6	0	0
Plan de Mitigación de Riesgos	15	1	8	0	3
Evaluación de áreas de la organización	7	1	4	0	1
Informe del Levantamiento de Información	13	9	7	5	6
Diccionario de Datos	15	2	10	0	0
Plan de Gestión de Requisitos	22	4	10	1	4
Total	163	50	82	20	40

Tabla # 4 Resultados de las Listas de Chequeo

Revisiones con el documento de No Conformidades	
Documentos	Cant. NC
Arquitectura de la Información	1
Bus Matriz.	11
Especificación de dimensiones.	1
Roles y responsabilidades.	0
Ambiente de desarrollo.	3
Documento Visión.	5
Plan de capacitación.	0
Plan de mitigación de riesgos.	1
Evaluación de las áreas de la organización	6
Informe del Levantamiento de Información	1
Diccionario de Datos	3
Plan de Gestión de Requisitos	3
Total	35

Tabla # 5 Resultados con el documento de No Conformidades

- **Alguno de los artefactos que fueron generados**

  - **Plan de Pruebas**

1. Referencias

Código	Título
[1]	Diseño de Casos de Usos del Sistema.doc
[2]	Plan de Aseguramiento de la Calidad.doc
[3]	Especificación de Requisitos.doc

2. Roles y Responsabilidades

Rol	Cantidad	Responsabilidad
Gerente de calidad	1	• Asegura que la aplicación producida se ajusta a las especificaciones y está razonablemente libre de errores. • Proporciona una estrategia de prueba. • Coordina las pruebas de calidad interna, las pruebas de liberación y aceptación.

		• Evalúa los resultados de las pruebas de calidad. • Rige y dirige las actividades de aseguramiento de la calidad.
Desarrollador de software	1	• Diseña los casos de pruebas. • Ejecuta las pruebas de unidad e integración. • Ejecuta las RTF de su documentación en unión con el revisor técnico.
Jefe de Prueba	1	• Diseña los casos de prueba. • Define las listas de chequeo. • Evalúa y documenta el resultado de las pruebas realizadas al software. • Realiza el Plan de Prueba.
Revisor Técnico	1	• Ejecuta las RTF. • Elabora el informe de las RTF.
Probador	4	• Ejecuta las pruebas diseñadas. • Elabora el documento de no conformidades

3. Escenario de pruebas

Casos de Usos a probar:

- Analizar indicadores por Subordinación
- Analizar indicadores por Organismo
- Analizar indicadores por Actividad Económica
- Analizar indicadores por Provincia
- Analizar indicadores por Localización.
- Analizar indicadores por información externa a los cortes

4. Requerimientos a probar

Especificación de Requisitos.doc

5. Estrategia de pruebas a realizar

Esta estrategia precisa cómo conducir las pruebas de forma apropiada, especificando los tipos que se llevan a cabo, con sus objetivos, técnicas y herramientas que las soporten.

5.1 Tipos de pruebas

5.1.1 Pruebas de aceptación

- Son básicamente pruebas funcionales sobre el sistema completo, para detectar errores en comportamiento con respecto a especificación de requerimientos.
- Realizadas por el cliente con la presencia de un desarrollador para registrar las No conformidades que se vayan detectando.
- Se buscan inconsistencias entre la especificación de requisitos y el manual del usuario.
- Dentro de las pruebas de aceptación se realizaran las siguientes:
  - ✓ 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.
  - ✓ Pruebas Beta: son realizadas por los usuarios finales del software en los lugares de trabajo, sin la presencia del desarrollador.

5.1.2 Pruebas del sistema

- Para detectar errores en comportamiento con respecto a especificación de requerimientos.
- Realizadas por el grupo de calidad y algunos desarrolladores.
- Se utilizan técnicas de caja negra basada en especificaciones de requerimientos y casos de uso.
- Dentro de las pruebas de sistemas se realizaran las siguientes:
  - ✓ Pruebas de Funcionalidad: para verificar la función del sistema, para validar que el sistema cumpla con los requisitos funcionales y no funcionales especificados en el diseño de la solución.
  - ✓ Pruebas de Rendimiento o Carga: para probar el rendimiento del software en tiempo de ejecución, evalúan la aceptabilidad de un elemento de un sistema sobre diferentes cargas de trabajo.
  - ✓ Pruebas de Seguridad: para verificar los mecanismos de protección.
  - ✓ Pruebas de Volumen: Verifican que la aplicación funcione adecuadamente bajo los siguientes escenarios de volumen:
    - Determinan la cantidad de datos con la cual el sistema falla.

- Carga máxima que el sistema soporta en un período dado.

Técnicas que utiliza:

2. Usar múltiples clientes, corriendo las mismas pruebas o pruebas complementarias para producir el peor caso de volumen por un período extendido.

3. Utilizar un tamaño máximo de Base de datos (actual o con datos representativos) y múltiples clientes para correr consultas simultáneamente para períodos extendidos.

- ✓ Pruebas de Compatibilidad: para verificar el funcionamiento del sistema sobre diferentes componentes de software y verificar la compatibilidad de la aplicación con sistemas operativos y navegadores.
- ✓ Pruebas de Estrés: esta es una de las más importantes porque se refieren a cargas extremas, memoria insuficiente, no disponibilidad de servicios y hardware o recursos compartidos limitados. Verifican que el sistema funcione sin errores con el mínimo hardware y software requerido.
- ✓ Pruebas de Volumen:

Requerimientos mínimos para su aplicación:

- El sistema operativo sobre el que se trabajará será Windows XP o superior.
- En la PC donde se encuentre el Servidor de Bases de Datos debe tener instalado PostgreSql.
- Las PCs donde se ejecute el sistema requieren tener instalado un navegador Web.

6. Evaluación de las pruebas

Se registraron las No Conformidades detectadas durante la revisión del documento y se aplican los diseños de Casos de Pruebas para cada Caso de Uso registrándose también las No Conformidades luego de su aplicación.

7. Cronograma

No	Tarea	Fecha	Responsable	Participantes	Observaciones
1	Establecimiento del Plan de Aseguramiento de la Calidad	23/3/2009	Gerente de Calidad	Marielis Izquierdo Matías Daimi Bretones Lorenzo	
2	Revisión de la documentación del grupo de trabajo Almacenes	18/5/2009	Grupo Almacenes Revisor Técnico	Odaimy Caballero Padrón Amaia Yoldi Iglesias	
3	Revisión de la documentación del grupo de trabajo Análisis.	24/5/2009	Grupo de Análisis Revisor Técnico	Azalia García Rubio. Yasmani Romero Montero	
4	Revisión de la documentación del grupo de trabajo BI.	4/6/2009	Revisor Técnico	Laritza Rodríguez la Rosa	
5	Revisión de la documentación del grupo de trabajo ETL.	25/6/2009	Revisor Técnico	Odaimy Caballero Padrón	
6	Diseño de los casos de pruebas.	10/7/2009	Jefe de Pruebas Desarrollador de Software	Odaimy Caballero Padrón Daimi Bretones Lorenzo Julio Ernesto Ortiz	

3.3 Aplicación de la guía propuesta en el proyecto Generador de Reportes Dinámico (GR)

Es importante destacar que la mayoría de los proyectos que se desarrollan en el CENTALAD aun están en etapas iniciales por lo cual las pruebas internas es un aspecto que aun no se ha podido probar a cabalidad, a salvo de esto está el proyecto Generador de Reportes que si paso por el proceso de pruebas internas el cual será mostrado a continuación.

El Generador de Reportes pertenece a la Línea de Herramientas de Procesamiento y Análisis de Datos la cual está compuesta por 3 grupos de trabajo, todos los proyectos que pertenecen a la misma están compuestos por estos grupos:

- ✓ Análisis
- ✓ Arquitectura
- ✓ Desarrollo

❖ **Ejecución de los pasos definidos para las pruebas:**

Paso 4: El jefe de Línea se encarga de entregarle al grupo de calidad el producto para que realice las pruebas previas a la liberación.

Dentro de este paso se realiza:

✍ **Pruebas de Integración**

Entradas

- Plan de pruebas de integración
- Diseño de casos de pruebas
- Código fuente

Salidas

- Documento de No Conformidades

Herramienta

- Herramientas automatizadas

- **Pruebas de Sistema**

  Entradas

  - Plan de pruebas del sistema
  - Diseño de casos de pruebas
  - Producto de Software
  - Base de Datos

  Salidas

  - Documento de No Conformidades

  Herramienta

  - Herramientas automatizadas

Personal que participa:

- Grupo de calidad para revisar nuevamente la documentación luego de la revisión realizada por los grupos para detectar si existen más No Conformidades y realizar las pruebas internas al producto.
- Algún desarrollador.

Paso 5: El grupo de calidad le solicita a Calidad UCI la liberación del producto.

Dentro de este paso se realiza:

- Pruebas de Sistema

Personal que participa:

- Calidad UCI

Paso 6: Se realizan las pruebas de aceptación con el cliente.

Dentro de este paso se realiza:

- Pruebas de Aceptación

Personal que participa:

- El Cliente con la presencia del desarrollador como observador.

3.3.1 Resultados obtenidos en el proyecto GR

Los resultados obtenidos de las pruebas realizadas al GR fueron satisfactorios y para un mejor entendimiento se dividieron los resultados en dos etapas. La 1^{era} etapa la comprende los resultados de las pruebas realizadas en el proyecto por el grupo de calidad y algunos de los desarrolladores de los grupos, evidenciándose un resultado muy bueno con la revisión del proyecto en 3 iteraciones tanto por la parte de las pruebas internas donde se hicieron pruebas de sistema, como por la parte de la revisión con los diseños de casos de pruebas. La 2da etapa esta comprendida por los resultados que se obtuvieron de la liberación del proyecto por parte de ERP.

- **Resultados de la 1^{era} Etapa**

	1era Iteración				2era Iteración				3era Iteración			
Aplicación	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**
Total	30	0	11	19	20	0	14	6	6	0	6	0

Tabla # 6 Resultados de las pruebas a la Aplicación

	1era Iteración				2era Iteración				3era Iteración			
DCP	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**
Total	46	0	14	32	33	0	15	18	23	0	23	0

Tabla # 7 Resultados de las pruebas con DCP

❖ **Resultados de la 2da Etapa**

	1era Iteración				2era Iteración				3era Iteración			
Aplicación	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**
Total	21	9	5	7	21	3	16	5	6	0	6	0

Tabla # 8 Resultados de las pruebas a la Aplicación

	1era Iteración				2era Iteración			
DCP	**NC**	**NC-NP**	**NC-RA**	**NC-PD**	**NC**	**NC-NP**	**NC-RA**	**NC-PD**
Total	150	40	110	0	41	0	41	0

Tabla # 9 Resultados de las pruebas con DCP

➢ **Alguno de los artefactos que fueron generados**

✍ **Diseño de casos de pruebas del CU Autenticar**

1. Descripción general

El caso de uso se encarga de autenticar un usuario en el sistema, el usuario introduce el nombre que lo identifica en el sistema y la contraseña para acceder al mismo, estos datos se comprueban en el servidor, permitiendo la entrada del usuario al sistema.

2. Condiciones de ejecución

Se debe acceder a la dirección electrónica de la aplicación.

3. Secciones a probar en el caso de uso

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**
SC 1: Autenticar.	EC 1.1: Autenticación válida.	Debe permitir a los usuarios autenticarse y entrar a la aplicación con los privilegios correspondientes a su rol.
	EC 1.2: Autenticación con fallo de conexión al Servidor de base de datos.	Si el sistema al verificar las credenciales del usuario, detecta algún problema en la conexión entre el Servidor de aplicaciones y el Servidor de base de datos, el sistema debe mostrar un mensaje de error.
	EC 1.3: Autenticación con datos incompletos.	Si el usuario no introduce todos los datos, el sistema debe mostrar un mensaje indicando el error.
	EC 1.4: Autenticación con datos incorrectos.	Si el usuario introduce datos incorrectos, el sistema debe mostrar un mensaje indicando el error.

4. Descripción de variable

No	**Nombre de campo**	**Clasificación**	**Puede ser nulo**	**Descripción**
[1]	Usuario.	Campo de texto.	No.	Combinación de letras, números y caracteres.
[2]	Contraseña.	Campo de texto.	No.	

5. Matriz de datos

5.1 SC 1 Autenticar

Escenario	**Variable 1 (Usuario)**	**Variable 2 (Contraseña**	**Flujo central**	**Respuesta del sistema**	**Resultado**
EC 1.1: Autenticación válida.	V (administrador)	V (helvex1*)	1. Se introducen las credenciales válidas de autenticación en el campo Usuario y Contraseña. 2. Se presiona el botón **Iniciar sesión**.	El sistema debe validar que exista conexión con la base de datos y debe validar los datos entrados por el usuario. El sistema debe permitir la entrada al usuario con los privilegios correspondiente a su rol.	
EC 1.2: Autenticación con fallo de conexión al Servidor de base de datos.	V (administrador).	V (helvex1*).	1. Se introducen las credenciales de autenticación en el campo Usuario y Contraseña. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.	El sistema debe validar que exista conexión con la base de datos y debe emitir el mensaje de error: “Ha fallado la conexión con la base de datos”.	

Escenario	Variable 1 (Usuario)	Variable 2 (Contraseña	Flujo central	Respuesta del sistema	Resultado
EC 1.3: Autenticación con datos incompletos.	V (vacío).	V (helvex1*).	1. Se debe dejar vacío el campo Usuario. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.	El sistema debe emitir el mensaje de error: “No debe dejar campos vacíos”.	
	V (administrador).	V (vacío).	1. Se debe dejar vacío el campo Contraseña. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.		
	V (vacío)	V (vacío)	1. Se deben dejar vacíos los campos: Usuario y Contraseña. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.		

Escenario	Variable 1 (Usuario)	Variable 2 (Contraseña	Flujo central	Respuesta del sistema	Resultado
EC 1.4: Autenticación con datos incorrectos.	I (pedro)	V (helvex1*)	1. Se debe introducir una credencial no válida en el campo Usuario. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.	El sistema debe emitir el mensaje de error: “Usuario o contraseña incorrecta”.	
	V (administrador)	I (pedro)	1. Se debe introducir una credencial no válida en el campo Contraseña. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.		
	I (pedro).	I (pedro)	1. Se deben introducir credenciales no válidas en los campos: Usuario y Contraseña. 2. Se presiona en el botón **Iniciar sesión**. 3. Se presiona el botón **Aceptar**.		

3.4 Comparación de los proyectos con respecto a la cantidad de iteraciones

Los proyectos en la UCI pasan por un proceso de liberación donde quedan registrados los datos de la cantidad de iteraciones por las cuales transitaron. A continuación se mostraran los datos que fueron recogidos de los Boletines de Producción de la UCI donde se muestra que la mayoría de los proyectos oscilan entre las 2 y 5 iteraciones, sirviendo esto de comparación con la cantidad de iteraciones que fue liberado el proyecto GR demostrando que los resultados fueron satisfactorios y en tiempo. Ver Tabla # 10

Proyecto	Iteración
F3- CCV	4ta
F9- SACGIR	3era
F10- Asesoría para la migración a Software Libre del MINPPAL	4ta
F8- Software Educativo MENPET	2da
F8- CICPC	4ta
F7- ALAS_HIS	3era
F6- Árbol Genealógico	4ta
F3- Sistema de Reportes del MENPET	3era
F4- Gestión Integral de Recursos Humanos - Aduana.	3era
F2- CTAISC (SIGESC)	2da
F1- Identidad 2	3era
F6- CNBA MINPPAL	3era
F4- SIGEP	3era
F3- RN	3era
F5- Software Educativo del MENPET	3era
F7- SASA	3era
F10- MENPET	3era
F10- CNTI Habilitador Metodológico	4ta
F9- Plataforma del Canal de Televisión MENPET	3era
F2- Levantamiento Tecnológico MENPET	3era
F4- SIGEP	4ta
F6 - Genética Médica	2da
F3 - SINAPSIS	3era
F6 - Visualizador y Editor Molecular	3era
F6 - Plataforma de Tareas Distribuidas	4ta
F10 - CNTI	5ta
CENTALAD- Generador de Reportes Dinámicos	**3era**

Tabla # 10 Estadística de las iteraciones por proyectos de la UCI

3.5 Conclusiones

Después de la aplicación de la propuesta se puede decir que se contribuyó en gran medida a introducir buenas practicas de Calidad Total en los proyectos productivos que le fue aplicada la propuesta demostrando que la misma es una solución para la mejora de los procesos de gestión de la calidad. Los procesos quedaron debidamente descritos y aplicados en cada uno de los proyectos siguiendo la estructura propuesta. Los resultados obtenidos en la liberación del proyecto GR demuestra claramente que fueron satisfactorios y que no sobrepasaron la cantidad de iteraciones establecidas por el Laboratorio de Calidad UCI. La guía que se propuso coloca en una mejor posición a la gestión de la calidad respecto al proceso que se llevaba anteriormente en CENTALAD, esto se logra a partir de la aplicación parcial del proceso definido.

CONCLUSIONES

- Para la realización de la guía propuesta se efectuó un estudio de temas referentes a la gestión de la calidad que aportaron conocimiento para su desarrollo y puesta en práctica.

- La guía propuesta fue un gran aporte para el CENTALAD por la presencia de buenas prácticas de Calidad Total que aportaron una mejora para los procesos de gestión de la calidad de los proyectos.

- La propuesta tuvo una puesta en práctica de forma parcial en algunos de los proyectos del CENTALAD y se realizo un análisis demostrando que la utilización de la guía es la solución para lograr una estructura organizativa en los procesos de gestión de la calidad de los proyectos productivos del Centro.

- Hubo un número considerable de No Conformidades detectadas en los documentos por fases de cada uno de los proyectos contribuyendo así a la reducción de errores a la hora de ser entregados para su liberación.

- Se obtuvieron resultados muy buenos con respecto a la cantidad de iteraciones en que fue liberado el proyecto Generador de Reportes cumpliendo con las establecidas por la dirección de Calidad UCI.

RECOMENDACIONES

El presente trabajo está encaminado a organizar el proceso de pruebas y mejorar la calidad de las mismas y por consiguiente, elevar los índices de calidad requeridos para los productos de software que se desarrollan en el CENTALAD y en los proyectos que utilicen dicho trabajo.

Recomendamos que:

- Continuar mejorando las Listas de Chequeo de los documentos de cada una de las fases del desarrollo según la metodología que se utilice.

- Se continúe el estudio y la profundización de todos los aspectos que se refieren a las pruebas de software.

- Se tenga en cuenta la propuesta obtenida, a la hora de realizar las pruebas en los diferentes proyectos de la Universidad, para que sea evaluada y teniendo en cuenta los resultados de la evaluación del proceso, se continúe perfeccionando y enriqueciendo.

- En cuanto a las herramientas automatizadas se continúe investigando y profundizando acerca de las mismas.

REFERENCIAS BIBLIOGRÁFICAS

[1]. **Ruffinatti, Adrián F.** WinRED. [En línea] 25 de Mayo de 2005. [Citado el: 4 de Febrero de 2009.] http://winred.com/management/la-industria-del-software-en-la-india-un-exito-casual/gmx-niv116-con2774.htm.

[2]. La Industria del Software Costarricense. **Gómez, Daira.** 18, Costa Rica : Éxito Empresarial, 2004.

[3]. **Coca, Yandira Mouriz y Cruz, Maite González.** Revista Ciencias. [En línea] 27 de Marzo de 2007. [Citado el: 5 de Febrero de 2009.] http://www.revistaciencias.com/publicaciones/EEZVpElyyplXUCmqSb.php.

[4]. **Ramos, Daniel Rojas.** El Prisma. [En línea] [Citado el: 30 de Abril de 2009.] http://www.elprisma.com/apuntes/ingenieria_industrial/teoriasdelacalidad/default.asp.

[5]. **Pérez, Memo.** GestioPolis. [En línea] [Citado el: 7 de Febrero de 2009.] http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/caltotalmemo.htm.

[6]. Conferencia de Gestión de Software. Pérez, Dr.C Pedro Y. Piñero. La Habana : Planificación de proyecto, 2008.

[7]. **Pillou, Jean-Francois.** Kioskea. [En línea] 16 de Octubre de 2008. [Citado el: 12 de Marzo de 2009.] http://es.kioskea.net/contents/qualite/management-qualite.php3.

[8]. **Jorrin, Michael González.** *Proceso de pruebas para la liberación de productos de software.* La Habana : Tesis de Maestria, 2007.

[9]. **Pressman.** Ingeniería de Software, un enfoque práctico. Madrid, 2001.

[10]. **Schuldt, Jurgen E.** [En línea] http://www.geocities.com/WallStreet/Exchange/9158/tq.htm.

[11]. **MEDEL, HECTOR ROJAS.** Monografias. [En línea] [Citado el: 9 de Febrero de 2009.] http://www.monografias.com/trabajos15/calidad-serv/calidad-serv.shtml.

[12]. **Gonzalez, Carlos.** Monografias. [En línea] [Citado el: 20 de Febrero de 2009.] http://www.monografias.com/trabajos11/conge/conge.shtml.

[13]. **Departamento de Sistemas Informáticos y Computación, Universidad de Valencia.** Proceso de desarrollo de software. [En línea] http://www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc.

[14]. **Boucchechter, Israel y Rojas, Richard.** Ciclos de Vida de Ingeniería del Software. [En línea] 17 de Enero de 2005. [Citado el: 23 de Abril de 2009.] http://carolina.terna.net/ingsw2/Datos/Cascada-ModeloV.doc.

[15]. Conferencia de Modelos de Desarrollo de Sofware. **Pérez, Dr.C Profesor Auxiliar Pedro Y. Piñero y Vázquez, MsC Maikel Yelandi Leyva.** La Habana : s.n., 2008.

[16]. **Alonso, Enrique Barreiro.** Escuela Superior de Ingeniería Informática. *España.* [En línea] [Citado el: 14 de mayo de 2009.] http://trevinca.ei.uvigo.es/~ebalonso/asignaturas/esx/guiones/esxClase4.pdf.

[17]. **Kynetia, Compañia.** *Madrid.* [En línea] Metodologías de pruebas. [Citado el: 14 de mayo de 2009.] http://www.kynetia.es/calidad/metodologia-de-pruebas.html.

[18]. **Jonás A. Montilva C., Ph.D.** Desarrollo de Software Basado en Lineas de Productos de Software. Venezuela : Universidad de Los Andes.

[19]. **C, Jonás A. Montilva, Arapé, Nelson y Colmenares, Juan Andrés.** Webdelprofesor. [En línea] Universidad de Mérida. [Citado el: 15 de mayo de 2009.] http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03%20Desarrollo%20de%20componentes.pdf.

[20]. **Commons, Creative.** Kioskea. [En línea] 16 de octubre de 2008. [Citado el: 20 de mayo de 2009.] http://es.kioskea.net/contents/genie-logiciel/methodes-agiles.php3.

BIBLIOGRAFÍA

1. **Ruffinatti, Adrián F.** WinRED. [En línea] 25 de Mayo de 2005. [Citado el: 4 de Febrero de 2009.] http://winred.com/management/la-industria-del-software-en-la-india-un-exito-casual/gmx-niv116-con2774.htm.

2. La Industria del Software Costarricense. **Gómez, Daira.** 18, Costa Rica : Éxito Empresarial, 2004.

3. **Coca, Yandira Mouriz y Cruz, Maite González.** Revista Ciencias. [En línea] 27 de Marzo de 2007. [Citado el: 5 de Febrero de 2009.] http://www.revistaciencias.com/publicaciones/EEZVpElyypIXUCmqSb.php.

4. **Ramos, Daniel Rojas.** El Prisma. [En línea] [Citado el: 30 de Abril de 2009.] http://www.elprisma.com/apuntes/ingenieria_industrial/teoriasdelacalidad/default.asp.

5. **Pérez, Memo.** GestioPolis. [En línea] [Citado el: 7 de Febrero de 2009.] http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/caltotalmemo.htm.

6. Conferencia de Gestión de Software. Pérez, Dr.C Pedro Y. Piñero. La Habana : Planificación de proyecto, 2008.

7. **Pillou, Jean-Francois.** Kioskea. [En línea] 16 de Octubre de 2008. [Citado el: 12 de Marzo de 2009.] http://es.kioskea.net/contents/qualite/management-qualite.php3.

8. **Jorrin, Michael González.** *Proceso de pruebas para la liberación de productos de software.* La Habana : Tesis de Maestria, 2007.

9. **Pressman.** Ingeniería de Software, un enfoque práctico. Madrid, 2001.

10. **Schuldt, Jurgen E.** [En línea] http://www.geocities.com/WallStreet/Exchange/9158/tq.htm.

11. **MEDEL, HECTOR ROJAS.** Monografias. [En línea] [Citado el: 9 de Febrero de 2009.] http://www.monografias.com/trabajos15/calidad-serv/calidad-serv.shtml.

12. **Gonzalez, Carlos.** Monografias. [En línea] [Citado el: 20 de Febrero de 2009.] http://www.monografias.com/trabajos11/conge/conge.shtml.

13. **Departamento de Sistemas Informáticos y Computación, Universidad de Valencia.** Proceso de desarrollo de software. [En línea] http://www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc.

14. **Boucchechter, Israel y Rojas, Richard.** Ciclos de Vida de Ingeniería del Software. [En línea] 17 de Enero de 2005. [Citado el: 23 de Abril de 2009.] http://carolina.terna.net/ingsw2/Datos/Cascada-ModeloV.doc.

15. Conferencia de Modelos de Desarrollo de Sofware. **Pérez, Dr.C Profesor Auxiliar Pedro Y. Piñero y Vázquez, MsC Maikel Yelandi Leyva.** La Habana : s.n., 2008.

16. **Alonso, Enrique Barreiro.** Escuela Superior de Ingeniería Informática. *España.* [En línea] [Citado el: 14 de mayo de 2009.] http://trevinca.ei.uvigo.es/~ebalonso/asignaturas/esx/guiones/esxClase4.pdf.

17. **Kynetia, Compañia.** *Madrid.* [En línea] Metodologías de pruebas. [Citado el: 14 de mayo de 2009.] http://www.kynetia.es/calidad/metodologia-de-pruebas.html.

18. **Jonás A. Montilva C., Ph.D.** Desarrollo de Software Basado en Lineas de Productos de Software. Venezuela : Universidad de Los Andes.

19. **C, Jonás A. Montilva, Arapé, Nelson y Colmenares, Juan Andrés.** Webdelprofesor. [En línea] Universidad de Mérida. [Citado el: 15 de mayo de 2009.] http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03%20Desarrollo%20de%20componentes.pdf.

20. **Batista, María Lilia Rodríguez.** *Propuesta de Procedimiento para el Aseguramiento de la Calidad en la Facultad 7.* La Habana : Trabajo de Diploma, 2007.

21. **Alón, Lídice Delgado y Pérez, Heney Díaz.** *Propuesta de Mecanismo de Gestión de Calidad Interna para el proyecto Registro y Notarías.* La Habana : Trabajo de Diploma, 2007.

22. **Alvarez, Janier Esquijarosa.** *Sistema Integrado de Gestión Estadística. Rol de Administrador de Calidad.* La Habana : Trabajo de Diploma, 2007.

23. **Montoya, Yinimary Ortega y Zamora, Isis Margarita Blanco.** *Estrategia de Control de la Calidad mediante revisiones y auditorías para el proyecto CICPC.* La Habana : Trabajo de Diploma, 2007.

24. **Sánchez, Jacqueline Marín, Montalván, Deborat Pérez y Cabrera, Damarys Silva.** *Estrategia para estandarizar el proceso de pruebas de liberación de las aplicaciones médicas desarrolladas en la facultad 7.* La Habana, 2008

25. **ARubinstein y JJCukier.** *Subsidios y Novedades del SEI.* s.l. : CMMI.

26. *Modelo para la Capacitación de los Especialistas en Pruebas de Sistemas Software.* **Palomo, Miguel Ángel García y Elcuera, Mamdouh.** Madrid : s.n., 2007.

27. **Institute, Project Management.** *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK).* EE.UU : 3era Edición, 2004.

28. **Almenares, Liudmila Sanchez.** *Prueba Automática de Carga y Estrés.* La Habana : UCI, 2009.

29. **Jiménez, Yudaika Ray.** *Diseño del proceso de Aseguramiento de la Calidad del Software para SIGIA.* La Habana : Trabajo de Diploma, 2007.

30. **Batista, Dayma Direntau y Herrera, Adriana Román.** *Pruebas de Software para el módulo Servicio Autónomo en el proyecto Registros y Notarías.* La Habana : Trabajo de diploma, 2007.

31. **Ilina, Lisset Torres y González, Reinel Pérez.** *Propuesta de un sistema de métricas para la evaluación de los proyectos de gestión de la UCI .* La Habana : Trabajo de Diploma, 2007.

32. **Jorrín, Ing. Michael González y Aguilar, Ing. Violena Hernández.** *Pruebas de Aceptación para un software con la presencia de una Entidad Certificadora de la Calidad.* La Habana : UCI.

33. **Jorrín, Ing. Michael González.** *Manual del Ingeniero de Pruebas.* La Habana : UCI.

34. **Commons, Creative.** Kioskea. [En línea] 16 de octubre de 2008. [Citado el: 20 de mayo de 2009.] http://es.kioskea.net/contents/genie-logiciel/methodes-agiles.php3.

ANEXOS

Anexo # 1 Plan de Aseguramiento de la Calidad

1. Introducción

1.1 Propósito

Este documento es el encargado de describir cómo se asegurará la calidad de los productos del Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD).

1.2 Alcance

Este Plan es elaborado a partir del propuesto por la IEEE **Standard for Software Quality Assurance Plans** (Std **730-1998**). Ha sido adaptado a las condiciones específicas de los proyectos del CENTALAD. Por tanto, su uso es particularmente para los miembros del proyecto.

1.3 Definiciones, Acrónimos y Abreviaturas

Revisiones Técnicas Formales (RTF).

Casos de Prueba (CP)

1.4 Referencias a documentos

Código	Título
[1]	Artefactos x proyecto.xls
[2]	Plan de prueba.doc
[3]	Plan de desarrollo de Software.doc
[4]	Artefactos de Soluciones BI&W Versión 1.1.doc

2. Objetivos

La documentación y la gestión del proyecto son realizadas a lo largo de todo el ciclo del desarrollo de software. Ambos aspectos son de suma importancia para el desarrollo y mantenimiento de las aplicaciones.

La documentación de un programa empieza a la vez que la construcción del mismo y finaliza justo antes de la entrega del programa o aplicación al cliente. Así mismo, la documentación que se entrega al cliente tendrá que coincidir con la versión final de los programas que componen la aplicación.

Los objetivos que se persiguen con este plan de calidad son los siguientes:

- Mejorar la calidad y fiabilidad del producto.
- Detectar, registrar y corregir los defectos cuanto antes en el ciclo de vida del software.
- Disminuir los riesgos y las desviaciones.
- La corrección del producto final respecto a las necesidades del usuario.
- Verificar si el producto satisface sus especificaciones o los atributos de calidad fijados.
- Verificar la correcta estructuración de los expedientes de proyectos teniendo en cuenta los artefactos generados en cada una de las fases.

3. Gestión

3.1 Organización

Rol	Descripción	Nombre y apellidos
Gerente de calidad	• Asegura que la aplicación producida se ajusta a las especificaciones y está razonablemente libre de errores. • Proporciona una estrategia de prueba. • Coordina las pruebas de calidad interna, las pruebas de liberación y aceptación. • Evalúa los resultados de las pruebas de calidad. • Rige y dirige las actividades de aseguramiento de la calidad.	Ing. Marielis Izquierdo Matías
Jefe de Prueba	• Diseña los casos de prueba. • Define las listas de Chequeo. • Evalúa y documenta el resultado de las pruebas realizadas al software. • Realiza la estrategia y plan de prueba.	Ing. Daimi Bretones Lorenzo
Revisor Técnico	• Ejecuta las RTF • Elabora el informe de las RTF.	Odaimy Caballero Padrón

Probador	• Ejecuta las pruebas diseñadas • Anota los resultados obtenidos	Amaia Yoldi Iglesias. Azalia García Rubio. Yasmani Romero Montero. Laritza Rodríguez la Rosa.

3.2 Tareas y responsabilidades

Tarea de Aseguramiento de calidad	Artefactos	Precondición	Poscondición	Responsable
Establecimiento del Plan de Aseguramiento de la Calidad.	Plan de Aseguramiento de la Calidad.			Gerente de Calidad
Definir el Plan de Prueba.	Plan de Pruebas.			Jefe de Prueba
Definir estrategia de prueba para el sistema.	Plan de Pruebas.			Gerente de Calidad
Diseñar los casos de prueba.	Casos de Prueba.			Jefe de Prueba
Revisión de la documentación mediante el Documento de No Conformidades.	Todos los artefactos de la fase de Inicio.	Tener definido los artefactos		Revisor Técnico
	Todos los artefactos de la fase de Elaboración.	Tener definido los artefactos		Revisor Técnico
	Todos los artefactos de la fase de Construcción.	Tener definido los artefactos		Revisor Técnico
	Todos los artefactos de la fase de Transición.	Tener definido los artefactos		Revisor Técnico
Pruebas internas al sistema.		Implementación		Probador

Liberación del producto		Prueba	Implantación	Dirección de Calidad

4. Documentación

- Plan de Aseguramiento de la calidad de cada uno de los proyectos del CENTALAD.
- Plan de Pruebas de cada uno de los proyectos del CENTALAD.

5. Plan de revisiones y auditorías

5.1 Tareas Generales

- Prueba a versiones liberadas: revisión del código y pruebas.
- Auditorías internas: auditorías al proyecto completo, artefactos y procesos.
- Revisión expediente proyecto: revisión a la documentación del proyecto.
- Certificación final del producto: inspección general al producto y artefactos asociados (documentación y demás), y solicitud a instancias superiores para certificación del mismo.

5.2 Artefactos entregables por fases

5.2.1 Según expediente de proyecto definido por Calidad UCI

Plantilla/Fase	Inicio	Elaboración	Construcción	Transición
Requisitos				
Plantilla DCS Diagrama de Proceso-Nombre del Proceso	x			
Plantilla DCS Especificación de requisitos	x			
Plantilla DCS Evaluación de Áreas de la Organización	x			
Plantilla DCS Modelo de Casos de uso del sistema	x			
Plantilla DCS Modelo del Dominio	x			
Plantilla DCS Modelo del Negocio	x			
Plantilla DCS Plan de gestión de requisitos	x			
Arquitectura y diseño				
Plantilla DCS Arquitectura de Información		x		
Plantilla DCS Documento de Arquitectura de Software		x		
Plantilla DCS Informe del Levantamiento de Información para la Arquitectura de Información	x			
Plantilla DCS Modelo de Diseño		x		
Implementación y pruebas				
Código fuente			x	
Manual de usuario			x	
Plantilla DCS Diseño casos de prueba			x	
Plantilla DCS Plan de pruebas		x		
Despliegue e instalación				
Plantilla DCS Modelo de Despliegue			x	
Plantilla DCS Presupuesto	x			
Plantilla DCS Lista de riesgos	x			
Plantilla DCS Plan Mitigación de Riesgos	x			
Plantilla DCS Ambiente de desarrollo	x			
Plantilla DCS Plan de capacitación	x			
Plantilla DCS Roles y responsabilidad	x			
Plantilla DCS Documento Visión	x			
Plantilla DCS Diagnóstico	x			
Plantilla DCS Minuta de reunión	x	x	x	x
Soporte				
Plantilla DCS Glosario de términos	x			
Plantilla DCS Listas de chequeo	x			
Plantilla DCS No Conformidades			x	
Plantilla DCS Plan aseguramiento de la calidad	x			
Plantilla DCS Plan de mediciones	x			
Plantilla DCS Respuestas a No Conformidades			x	
Plantilla DCS Solicitud de cambio	x	x	x	x
Plantilla DCS Pedido de cambio	x	x	x	x
Plantilla DCS Plan Gestión de Configuración	x	x	x	x

5.2.2 Cronograma de revisiones.

Nombre del Proyecto	Fase	Fecha de Cierre	Fecha de Revisión Grupo de Calidad	Fecha de Revisión Calidad UCI
PostgreSQL				
	Inicio	15/5/2009	25/5/2009	15/6/2009
	Elaboración	15/7/2009	20/7/2009	15/9/2009
	Construcción	31/1/2010	8/2/2010	28/2/2010
	Transición	1/3/2010	15/3/2010	30/4/2010
INE				
	Inicio	30/5/2009	8/6/2009	30/7/2009
	Elaboración	30/9/2009	15/10/2009	30/11/2009
	Construcción	15/3/2010	20/4/2010	15/5/2010
	Transición	30/4/2010	10/5/2010	30/6/2010
PADTSI				
	Inicio	30/4/2009	13/5/2009	30/6/2009
	Elaboración	30/5/2009	15/6/2009	30/7/2009
	Construcción	30/9/2009	14/10/2009	30/11/2009
	Transición	30/1/2010	18/2/2010	30/3/2010
ONE				
	Inicio	27/04/2009	18/05/2009	30/6/2009
	Elaboración	20/05/2009	24/05/2009	15/6/2009
	Construcción	30/07/2009	13/07/2009	30/9/2009
	Transición	31/9/2009	10/10/2009	25/11/2009

5.3 Resolución de problemas y actividades de corrección

Pasos a seguir:

- Llevar el registro de las no conformidades detectadas.
- Informar al líder sobre los problemas detectados.
- Definir el problema y poner en marcha acciones contenedoras.

6. Plan de Pruebas

Ver el Plan de Pruebas de cada uno de los proyectos.

Anexo # 2 Plantilla Lista de Chequeo de Roles y Responsabilidades

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos**	**Comentarios**

				afectados	
	1. ¿Se especificó los roles existentes?				
	2. ¿Se definió las responsabilidades de cada rol?				
	3. ¿Se definen los equipos de trabajo por cada una de las fases?				
	4. ¿Los elementos de la tabla de los roles y responsabilidades fueron descritos correctamente?				
	5. ¿Los elementos de la tabla de los equipos de trabajo por fase fueron descritos correctamente?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

crítico	1. ¿Ha identificado errores ortográficos?				
crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del

								se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Procede	equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

Anexo # 3 Plantilla Lista de Chequeo del Ambiente de Desarrollo

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de				

	proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Se describió el ambiente de desarrollo?				
	2. ¿Se definieron los servidores para el desarrollo?				
	3. ¿Los elementos de la tabla de los servidores fueron descritos correctamente?				
	4. ¿Se definieron las PC clientes para el desarrollo?				
	5. ¿Los elementos de la tabla de las PC clientes fueron descritos correctamente?				
	6. ¿Se definió el modelo de despliegue del ambiente de desarrollo?				

Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Ha identificado errores ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Proceder	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

Anexo # 4 Plantilla Lista de Chequeo de Bus Matrix-1

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores,

especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].
Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.
Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.
Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.
Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.
Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.
Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.
Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.
Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.
N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Se definieron las necesidades del cliente?				
Crítico	2. ¿Se especificaron las listas de Data Marts?				
Crítico	3. ¿Se especificaron la lista de				

	dimensiones?				
Crítico	4. ¿Se confecciono la Matriz Bus?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	5. ¿Ha identificado errores ortográficos?				
Crítico	6. ¿Se entiende claramente lo que se ha especificado en el documento?				
	7. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	8. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Proceder	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

Anexo # 5 Plantilla Lista de Chequeo de Especificación de Dimensiones

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores,

especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
Crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Se realizo el Modelo dimensional?				
Crítico	2. ¿Se especificaron cada una de las dimensiones?				
Crítico	3. ¿Se especificaron las tablas de cada dimensión?				

Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	9. ¿Ha identificado errores ortográficos?				
Crítico	10. ¿Se entiende claramente lo que se ha especificado en el documento?				
	11. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	12. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Proceder	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

Anexo # 6 Plantilla Lista de Chequeo del Documento Visión

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores,

especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Se describieron las oportunidades de negocio con el proyecto?				
	2. ¿Se identificaron los posibles problemas que pueden ser resueltos con el proyecto?				

	3. ¿Se detectaron las principales características del Mercado que motivan el proyecto?				
	4. ¿Se definió el perfil de los involucrados?				
	5. ¿Se definió el perfil de los usuarios?				
	6. ¿Se registraron las principales necesidades de los involucrados y usuarios?				
	7. ¿Se definió la visión general del producto?				
	8. ¿Se registraron las restricciones de costo y precio que son relevantes?				
	9. ¿Se describieron las características básicas del producto?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Ha identificado errores				

	ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la

								se revisó.] RA: Resuelta PD: Pendiente NP: No Procede	no conformidad explica por qué.]

Anexo # 7 Plantilla Lista de Chequeo de la Arquitectura de Información

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Objetivos del Producto					
crítico	1. ¿Especifican que se pretende lograr con el				

	producto?				
	2. ¿Están iniciados con verbos que indican las acciones a realizar?				
Audiencia del Producto					
	4. ¿Se especifica quiénes serán los usuarios del producto?				
	5. ¿Se especifican los usuarios principales?				
crítico	6. ¿Se especifican las características de la audiencia principal?				
crítico	7. ¿Se especifican las necesidades de la audiencia que el producto pretende satisfacer?				
	8. ¿Se especifican sus expectativas?				
Contenido del Producto					
crítico	9. ¿Están numerados los contenidos a incluir en el producto?				
crítico	10. ¿Los contenidos del producto están agrupados y etiquetados? (la				

	taxonomía)				
crítico	11. ¿Existe un mapa de navegación del producto?				
crítico	12. ¿El mapa de navegación representa las secciones, niveles y contenidos relacionados?				
	13. ¿El sistema de navegación es consistente, uniforme y visible?				
crítico	13. ¿Se representa el diseño de la estructura de todas las pantallas tipo?				
	14. ¿Se identifican las diferentes áreas en la estructura de las pantallas?				
	15. ¿En los casos donde se realizan procesos complejos (más de 5 actividades) se incluye un diagrama de flujo sencillo que ejemplifique las posibles interacciones y sus resultados con las pantallas correspondientes?				
Semántica del documento					

Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios
Crítico	1. ¿Ha identificado errores ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en

								fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Procede	caso de no proceder la no conformidad explica por qué.]

Anexo # 8 Plantilla Lista de Chequeo del Plan de Mitigación de Riesgos

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

	1. ¿Se listaron los riesgos existentes?				
	2. ¿Se siguió la estructura para la descripción del riesgo?				
	3. ¿Se especifico los indicadores de cada uno de los riesgos?				
	4. ¿Se especificó la estrategia de mitigación para cada uno de los riesgos?				
	5. ¿Se especificó el plan de contingencia para cada uno de los riesgos?				
	6. ¿Se definió la gestión de riesgos?				
	7. ¿Se identificaron las tareas para la gestión de riesgos?				
	8. ¿Se especifico la organización y responsabilidades?				
	9. ¿Se establecieron las herramientas y técnicas a utilizar?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

Crítico	1. ¿Ha identificado errores ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en

								fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Procede	caso de no proceder la no conformidad explica por qué.]

Anexo # 9 Plantilla Lista de Chequeo del Diccionario de Datos

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Todos los datos				

	utilizados o producidos por el sistema han sido registrados y descritos en detalles?				
	2. De ser necesario, ¿ha incluido otro nombre del elemento de datos? (Alias)				
crítico	3. ¿Cada descripción del elemento de datos es clara y detallada?				
crítico	4. ¿Cada uno de los elementos o actividades del sistema tiene un único significado?				
	5. ¿Han sido registrados todos los detalles adicionales relacionados al flujo de datos del sistema?				
crítico	6. ¿Ha especificado si el elemento de datos contiene valor numérico,				

	caracteres o alfabético?				
crítico	7. ¿Ha especificado el máximo de caracteres o dígitos que puede tener un elemento de datos?				
	8. ¿Ha indicado cómo se presenta el dato al mostrarse en pantalla o al imprimirse en un reporte?				
crítico	9. ¿Ha registrado que tipo de valor específico debe tener el elemento de datos?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Ha identificado errores ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se				

	refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	< 1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Proceder	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

Anexo # 10 Plantilla Lista del Plan de Gestión de Requisitos

1. Introducción

Esta lista de chequeo cuenta con varios puntos, los cuales serán clasificados antes de ser aplicados, teniendo en cuenta su pertinencia, además de tener un punto donde se referencia las observaciones en cuanto al aspecto a evaluar.

2. Propósito y objetivos

El objetivo general de la lista de chequeo es evaluar las especificaciones del [nombre del artefacto] de los proyectos de la universidad. Esta plantilla ha sido confeccionada para guiar a desarrolladores, especialistas o expertos técnicos en la verificación y evaluación de las especificaciones del [nombre del artefacto].

Los aspectos definidos en esta lista de chequeo podrán ser referenciados en otras actividades de chequeo, en dependencia de lo que se necesite verificar.

Esta plantilla permitirá recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados.

3. Alcance

Esta plantilla es aplicable a cada una de las Revisiones de especificaciones del [nombre del artefacto] que se desarrollen.

4. Resumen

La plantilla tiene un carácter flexible para el especialista de calidad o el desarrollador teniendo en cuenta que pueden surgir modificaciones e inclusiones.

Forma de Uso:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.

Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

5. Estructura de la lista de chequeo

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				

Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Se describe quien es responsable del desarrollo de las actividades descritas en el flujo de trabajo de requisito?				
	2. ¿Describe las herramientas de software que serán utilizadas para la gestión de los requisitos?				
crítico	3. ¿Describe las herramientas y procedimientos que usados para el control de versiones de los Requisitos generados durante todo el				

	ciclo de vida?				
crítico	4. ¿Describe como los elementos serán nombrados, marcados y numerados?				
	5. ¿Se utiliza la tabla de contenidos del expediente de proyecto?				
crítico	6. ¿En la tabla de especifican correctamente los Artefactos, Elemento de traceabilidad y su descripción				
crítico	7. ¿Se definen los elementos de seguimiento?				
	8. ¿De cada elemento se definen un grupo de reglas o guías para aplicar la traceabilidad?				
crítico	9. ¿Se especifican los				

	atributos para elementos de seguimiento?				
	10. ¿Se especifica la lista de los atributos que serán utilizados para caracterizarlo?				
	11. Se especifica el conjunto de posibles estado de un elemento de seguimiento:				
	• Se especifican las características que están en discusión pero que aun no han sido revisadas y aceptadas.				
	• Se especifican las características que ya fue aprobada para				

	su implementación.				
	• Se especifican las características rechazadas.				
	• Se especifican la característica incorporada en un momento específico del proyecto.				
	12. Se especifica el conjunto de posibles beneficios que reporta al negocio:				
	• Se especifican las características esenciales para el negocio.				
	• Se especifican las				

	características importantes que no incluirla pudiera afectar la satisfacción de los clientes y usuarios.				
	• Se especifican las características que no afectan en nada la satisfacción del cliente				
	13. Se especifica el Esfuerzo en cuanto a los requerimientos: Bajo, Medio, Alto.				
	14. Se especifica la Estabilidad en cuanto a los cambios que pudiera tener un				

	requerimiento: Bajo, Medio, Alto**.**				
	15. ¿Se especifican los entregables del proyecto?				
	16. ¿Los entregables se especifican en una lista tabular de los artefactos que serán creados durante el proyecto, incluso las fechas de entrega designadas?				
	17. ¿Se hace referencia al documento de los procesos de Gestión de Configuración establecidos en la organización?				
Semántica del documento					
Peso	**Indicadores**	**Eval**	**(NP)**	**Cantidad de elementos**	**Comentarios**

	a Evaluar			**afectados**	
Crítico	1. ¿Ha identificado errores ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

6. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Proceder	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

GLOSARIO

(BI) Inteligencia de Negocios

(CENTALAD) Centro de Tecnologías de Almacenamiento y Análisis de Datos

(CMMI) Modelo Integrado de Capacidad y Madurez

(CALISOFT) Centro Nacional de Calidad de Software

(COTS) Component off The Shelf

(CSR) Componente de software reutilizable

(DCP) Diseño de casos de pruebas

(ETL) Extracción, Transformación y Carga (Load). Procedimientos (herramientas) destinados a obtener los datos de las fuentes operacionales, limpiarlos, convertirlos a los formatos de utilización y cargarlos en el repositorio final.

(GR) Generador de Reportes Dinámico

(ICSW) Industria Cubana del Software

(ISW) Ingeniería de Software

(IEEE) Instituto de Ingenieros Eléctricos y Electrónicos

(LPS) Líneas de Productos de Software

(OO) Orientado a Objeto

(ONE) Oficina Nacional de Estadísticas

(PMBok) Libro publicado por el Project Management Institute (PMI)

(QA) Aseguramiento de Calidad

(RTF) Revisiones Técnicas Formales

(SPICE) Software Process Improvement and Capability Determination

(SEI) Software Engineering Institute

(TI) Tecnologías de la Información

(TIC) Tecnologías de la información y la comunicación

(UCI) Universidad de las Ciencias Informáticas