UNIVERSIDAD GABRIELA MISTRAL
FACULTAD DE INGENIERIA

MODELO DE CALIDAD DE SOFTWARE BASADO EN CMMI & PMI, APLICADO AL DESARROLLO DE PROYECTOS DE TECNOLOGIAS DE LA INFORMACION

Tesis para optar al título de Ingeniero de Ejecución en Informática

Autor : Edgardo Patricio Meza Orellana.
Profesor Guía : Jaime Orellana Rebolledo.
Profesor Integrante : Jorge Tapia Castillo.

Santiago – Chile

Noviembre, 2010

INDICE

AGRADECIMIENTOS

Agradezco al buen Dios y todas sus manifestaciones por permitirme Vivir, Pensar, Amar y Evolucionar, a mi familia, padres y en especial, a mi Esposa e Hijos que sacrificaron su tiempo valioso en pro de mi desarrollo profesional brindándome apoyo, comprensión y amor en estos años.

I. INTRODUCCION

La Ingeniería de Software, una de las ingenierías más jóvenes, está evolucionando rápidamente y uno de los mayores pilares para esto es el enfoque en el diseño y calidad,estableciendo procesos que cubran todo el ciclo de vida del software.

Tener procesos de desarrollo de software ayuda a entender y analizar las diferentes etapas del ciclo vida dentro del desarrollo de software, promueve la utilización de las mejores prácticas y estándares de la industria, así como ayuda a definir procesos de mejora continua, que finalmente optimizan los costos productivos sin sacrificar la **calidad del producto final**.

Las teorías de gestión de procesos de software son una síntesis de los conceptos de gestión de la calidad trabajadas por Deming, Crosby y Juran[1] entre otros; que en los últimos 30 años han sido usadas para resolver problemas comunes en muchas áreas. Se han descubierto soluciones para un sin números de casos y problemáticas, pero existe una gran brecha entre el estado de la práctica y el estado del arte; sin embargo muchos de estos conceptos y soluciones han sido usados para construir lo que hoy conocemos como modelos de mejora de procesos.

Dentro de este contexto el modelo de Capability Maturity Model Integration (**CMMI**) desarrollado por el Software Engineering Institute (**SEI**), es un modelo de mejora continua que provee a las organizaciones desarrolladoras de software elementos esenciales para mejorar sus procesos y entregar finalmente un software o **Producto de Calidad,** para ello CMMI postula: "***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.***"

A sí mismo, se ha querido incluir, en este trabajo, las mejores prácticas o estándares de la industria referentes a la gestión o dirección de proyectos (**Project Management**), suministradas y recopiladas por el Project Management Institute (**PMI**); organismo norteamericano dedicado a promover la dirección de proyectos como una

[1] Tres expertos que han dedicado sus vidas a difundir los conceptos de calidad y sus contribuciones en las organizaciones para mejorar sus procesos, productos y servicios.

profesión, incorporando conocimiento, habilidades y destrezas necesarias en la **Dirección Profesional de Proyectos**.

Dado lo anterior el Modelo de Calidad de Software[2], que este trabajo de investigación pretende demostrar, se basa en los aspectos de mejora continua de procesos en el desarrollo de software que plantea el CMMI y las mejores prácticas en la dirección de proyectos que postula el PMI. Dentro de estos dos modelos hemos querido enfatizar las áreas de procesos y conocimientos referente al aseguramiento de la calidad (**QA**[3]) y el control de la calidad (**QC**[4]), permitiendo ser utilizados como herramientas dentro del desarrollo de software, para que finalmente sea entregado un producto de calidad, en los tiempos, costos establecidos y lo que es más importante a satisfacción y expectativas de nuestros clientes.

1.1 Motivación

Según estudios,**CHAOS[5],** más del 50% de los proyectos de desarrollo de software no son terminados en los plazos, ni costos acordados y lo que es más importante, no satisfacen los requerimientos originales, ni expectativas de los clientes y usuarios finales, es decir son proyectos fracasados que generan un producto de muy baja calidad.

Por ello la principal motivación de este trabajo es suministrar un modelo de calidad de software basado en las mejores prácticas o estándares, en procesos de mejora continua (**CMMI**) para organizaciones de fábrica de software y dirección de proyectos (**PMI**), que permitan entregar un producto final de calidad, en los plazos y costos establecidos.

Dentro del mismo contexto hemos querido incorporar en el modelo de calidad de software propuesto, las experiencias del tesista en el área de proyectos de desarrollo y

[2] Los modelos de calidad nos dice el ¿Qué? Hay que hacer y no el ¿Cómo? hay que hacerlo

[3] Quality Assurance o Aseguramiento de Calidad

[4] Quality Control o Control de Calidad

[5] The **Chaos** Report, estudios estadísticos sobre la base de los resultados de la dirección de proyectos en compañías de informática hecho por The Standish Group.

adquisición de tecnologías de la información, además de continuidad operacional de los mismos.

1.2 Hipótesis

Se pretende demostrar que, incorporando un modelo de calidad de software, en el ciclo de vida del desarrollo de proyectos de software, mejorando los procesos de las organizaciones que los sustentan e incorporando las mejores prácticas de la industria, permitirán entregar un producto final lo más apegado a los requerimientos, en los plazos y tiempos acordados.

1.3 Objetivo General

Proporcionar un modelo de calidad de software basado en **CMMI** y **PMI** que sirva como herramienta de **QA** y **QC** que acompañe en todo el ciclo de vida dentro del desarrollo de software, además de incorporar las mejores prácticas en la dirección y gestión de proyectos **TI**[6], para así disminuir la inmensa brecha que existe entre un proyecto exitoso que entrega un producto de calidad a los clientes y el que no lo es.

1.4 Objetivos Específicos

- Recoger las buenas prácticas de las áreas de procesos QA y QC de losmodelo de mejora continua en el desarrollo de proyectos de software CMMI y la guía de dirección de proyectos PMBOK.
- Generar un modelo o interface integraday modular que permita ser aplicada a los proyectos de tecnologías de la información, asegurando la calidad del producto de software.
- Proporcionar técnicas y herramientas para que el líder de proyecto y su grupo puedan utilizar en pro de la calidad del producto de software, independiente

[6] Abreviatura para indicar Tecnologías de la Información

de las metodologías, estándares o procedimientos que cualquier compañía posea.

1.5 Alcances

En la actualidad existe un sin número de estándares y modelos de calidad de software que son factibles de aplicar por las organizaciones que desarrollan, compran o mantienen software, este trabajo de investigación ha querido incorporar las mejores prácticas del modelo de mejora continua **CMMI** y dirección de proyectos del **PMI**, por ser los modelos o estándares más usados en el área de proyectos y continuidad operacional, ámbito en el cual se desempeña el tesista.

1.6 Antecedentes

En la actualidad existen muchas empresas que, dentro de su organización, recae la responsabilidad de liderar proyectos de desarrollo, mantención o adquisición de nuevas tecnologías de información, y han tenido que incorporar modelos de mejoras continuas o mejores prácticas dentro de sus procesos, para así asegurar y promover el exitoso termino de proyectos de desarrollo de software, entregando un producto de calidad a sus clientes internos y externos.

Según los estudios estadísticos a nivel mundial la mayoría de los proyectos de desarrollo de software no son entregados en los plazos, costos y calidad esperada por los clientes, esto ha motivado que organizaciones como el **Software Engineering Institute** y el **Project Management Institute**, hayan desarrollado el **CMMI** y el **PMI** respectivamente como modelos de calidad. Dentro de ese contexto se ha querido extraer las mejores prácticas de estos modelos y utilizarlas como herramientas dentro de las actividades de Ingeniería de Software.

II. MARCO TEORICO

A continuación se explicarán las grandes áreas o modelos en las que se basa este trabajo de investigación, en primer lugar hablaremos de la ingeniería de software con sus metodologías y procesos acompañado de los conceptos y prácticas de calidad de software, en segundo lugar detallaremos los modelos de mejoras continua **CMMI** y sus áreas de Validación y Verificación en conjunto con Control y Aseguramiento de la calidad, para luego finalizar con la explicación del **PMI** dentro del área de gestión de proyectos y como una buena gestión de un proyecto puede influir positiva o negativamente el producto final.

Explicados y desarrollados los aspectos fundamentales de estas áreas, pasaremos a dar forma al modelo de calidad de software planteado, recorriendo y aplicando las mejores prácticas a cada fase dentro del ciclo de vida o médelo del desarrollo de software.

2.1 Concepto Inicial de Software y Ciclo de Vida del Software.

El software es un elemento lógico e intangible, por lo tanto posee algunas características muy diferentes en relación a los productos de fabricación físicos o hardware.

El **software se desarrolla, no se fabrica** en el sentido clásico de la palabra. Ambas actividades se dirigen a la construcción de un "**producto**", pero los métodos son diferentes. Los costos del software se encuentran en la ingeniería, esto implica que los proyectos no se pueden gestionar como si lo fueran de fabricación.

A mediados de la década de 1980, se introdujo el concepto de "fábrica de software", que recomienda el uso de herramientas para el desarrollo automático del software.

Si se representa gráficamente la cantidad de fallos en función del tiempo, para el hardware se tiene la curva de fallos del hardware, másconocida como "curva de bañera"(Fig. 2.1).

Figura. 2.1. Curva de fallos del hardware

Al principio de su vida hay bastantes fallos (normalmente por defectos de diseño y/o fabricación), una vez corregidos se llega a un nivel estacionario (bastante bajo). Sin embargo conforme pasa el tiempo, aparecen de nuevo, por efecto de suciedad, malos usos o abusos, temperaturas extremas, falta de mantenimiento preventivo y otras causas. El hardware empieza a estropearse.

El **software no se estropea**. La gráfica de fallos en función del tiempo, tendría forma de caída desde el principio, hasta mantenerse estable por tiempo casi indefinido (Fig. 2.2).

Figura. 2.2. Curva de fallos Ideal del Software

El **software no es susceptible a los males del entorno** que provocan el deterioro del hardware. Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida, sin embargo, una vez corregidas no se producen

nuevos errores. Aunque no se estropea, si puede deteriorarse. Esto sucede debido a los cambios que se efectúan durante su vida (Fig. 2.3).

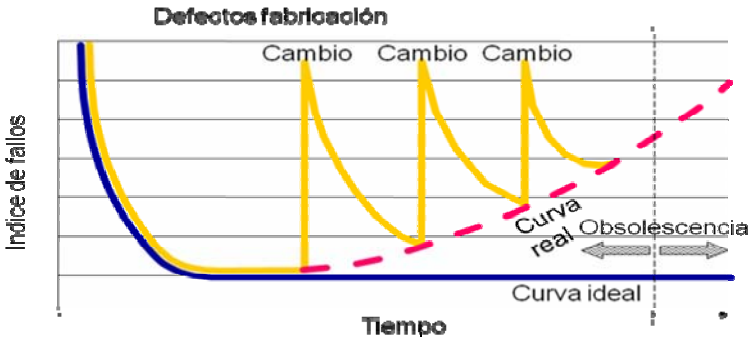


Figura. 2.3. Curva de Real de fallos del Software

Cuando un componente hardware se estropea, se cambia por otro que actúa como una "pieza de repuesto", mientras que para el software, no es habitual este proceso, lo cual significa que el mantenimiento de los programas es muy complejo.

La mayoría del software se **construye a medida**, en vez de ensamblar componentes previamente creados. En cambio el hardware se dispone de todo tipo de circuitos integrados, para fabricar de manera rápida un equipo completo. Los ingenieros de software no disponen de esta comodidad, aunque ya se están dando los primeros pasos en esta dirección, que facilitaría tanto el desarrollo de aplicaciones informáticas.

La formalización del proceso de desarrollo se define como un marco de referencia denominado ciclo de desarrollo del software o ciclo de vida del desarrollo del software. Se puede describir como, "el período de tiempo que comienza con la decisión de desarrollar un producto software y finaliza cuando se ha entregado éste". Este ciclo, por lo general incluye, una fase de requisitos, fase de diseño, fase de implantación, fase de prueba, y a veces, fase de instalación y aceptación.

2.2 Ingeniería de Software

Para desarrollar o construir un software se deben establecer, ordenar, priorizar y coordinar una serie de actividades, estas actividades son recogidas en una disciplina de la ingeniería conocida como **Ingeniería del Software** (SE del inglés Software Engineering), sobre las que se han basado y siguen basándose los desarrollos actuales de los sistemas informáticos.

2.2.1 Definiciones de Ingeniería del Software

El término de Ingeniería del Software ha sido definido por varios autores y, al igual que hace R. S. Pressman[7], citamos como una de las primeras la definida por Fritz Bauer[8]:

"La ingeniería del software es el establecimiento y uso de principios robustos de ingeniería con el fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales".

Esta definición no hace referencia alguna a aspectos como la satisfacción del cliente o de la importancia de realizar mediciones ni de la calidad del producto final (tema principal de esta tesis).

Posteriormente aparecen nuevas definiciones, siendo hoy en día la más comúnmente aceptada la establecida en IEEE[9]:

"Ingeniería de Software: Es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software"

La Ingeniería del Software, por tanto, no sólo cubre los aspectos puramente tecnológicos de la producción del software, sino que involucra además la gestión de los

[7] Roger S. Pressman, Ingeniería de Software un Enfoque Practico (Quinta Edición).

[8] El término ingeniería de software fue utilizado por primera vez por Fritz Bauer en la primera conferencia sobre desarrollo de software patrocinada por el Comité de Ciencia de la OTAN celebrada en Garmisch, Alemania en 1968.

[9] **IEEE** corresponde a las siglas de ***The Institute of Electrical and Electronics Engineers***, el **Instituto de Ingenieros Eléctricos y Electrónicos**, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

presupuestos, de proyectos y de los equipos de desarrollo, así como también de la planificación de dichos proyectos.

Comprende, además, un conjunto de tres elementos clave: Los modelos de proceso o métodos, las herramientas y los procedimientos. Los primeros indican cómo proceder para construir técnicamente software, las herramientas proporcionan el soporte automático o semiautomático para que los ingenieros software puedan desarrollar los métodos, y los procedimientos definen las secuencias de aplicación de los métodos, siendo los que sirven de enlace entre los métodos y las herramientas.

Para cada uno de los tres elementos citados existen varias propuestas que aunque son diferentes pueden combinarse para una mejor adaptación al proyecto o al equipo de desarrollo que lo lleve a cabo.

2.2.2 Modelos de Procesos de la Ingeniería del Software

Se entiende por proceso un conjunto organizado de actividades que transforma entradas (inputs) en salidas (outputs). Las descripciones de un proceso juntan o encapsulan conocimientos que podrán reutilizarse.

“Podemos considerar ejemplos de procesos básicos en la vida cotidiana el manual de instrucciones del refrigerador, un libro de recetas de cocina, el manual o libro de estándares corporativos para los empleados de un banco o el manual de calidad para la fabricación de repuestos de aviones, etc.”

Y por proceso de software se denomina al conjunto estructurado de actividades requeridas para desarrollar un sistema software (especificación, diseño, desarrollo y validación). Así pues, un Modelo de Proceso Software consistirá en la representación abstracta del proceso, constituyendo una descripción de un proceso software desde una perspectiva particular.

Como se ha mencionado anteriormente, existen varios modelos deproceso que describen cómo proceder eficientemente para laproducción de software. A pesar de que

no es objeto de este trabajodetallar todos y cada uno de ellos[10], se cree conveniente explicar laidea de algunos de ellos, puesto que por su importancia y/oaceptación han marcado la evolución del desarrollo del software y, sobre todo, porque son necesarios para comprender el modelo decalidad que posteriormente se presenta en este trabajo deinvestigación.

2.2.3 ModeloLineal o Cascada

Las dos figuras siguientes ilustran dos variantes del ciclo de vida más conocido para la Ingeniería del Software, denominado según los autores como “modelo lineal” (Fig. 2.4) o “modelo en cascada” (waterfall model) (Fig. 2.5),que sugiere un enfoque sistemático y secuencial del desarrollo del software que comienza a nivel del sistema y progresa a través del análisis, el diseño, la codificación, la prueba y el mantenimiento.

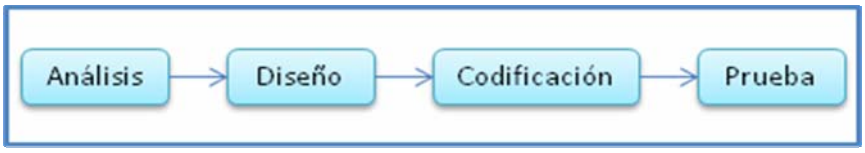
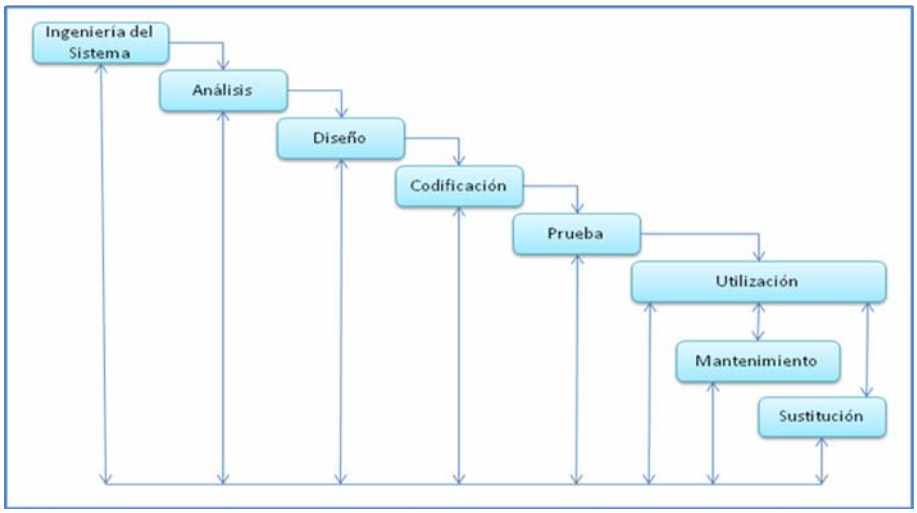


Figura. 2.4. Modelo Lineal o Cascada (Inicial)



[10]Es muy fácil además encontrar información detallada de todos los métodos en publicaciones especializadas como los libros anteriormente mencionados

Figura. 2.5. Modelo Lineal o Cascada Mejorado

De los dos esquemas anteriores, más que la misma concepción de una idea, el segundo constituye una versión más evolucionada y realista del primero, proporcionando una realimentación necesaria en todo proceso software.

Las diferentes fases que abarca son[11]:

Ingeniería y análisis del sistema: Debido a que un sistema software siempre forma parte de un sistema mayor, se comienza estableciendo los requerimientos de todos los elementos del sistema y asignando luego algún subconjunto de estos requerimientos al software. Esta visión del sistema es esencial cuando el software debe interrelacionarse con otros elementos tales como hardware, personas o bases de datos. Esta fase abarca los requerimientos globales a nivel de sistema con una pequeña cantidad de análisis y diseño a nivel superior.

Análisis de los requerimientos del software: El proceso de recogida de los requerimientos se centra e intensifica especialmente en el software. Para comprender la naturaleza de los programas que hay que construir, el ingeniero de software o "analista" debe comprender el dominio de la información del software, así como la función, rendimiento e interfaces requeridas. Los requerimientos, tanto del sistema como del software, se documentan y revisan con el cliente.

Diseño: El diseño del software es realmente un proceso múltiple que se enfoca sobre tres atributos distintos del programa: a) La estructura de los datos, la arquitectura del software y el detalle procedimental; b) El proceso de diseño traduce los requerimientos en alguna forma de representación del software;c) Como los requerimientos, el diseño se documenta y forma parte de la configuración del software.

Codificación (o implementación): El diseño debe traducirse en una forma inteligible para las máquinas, cuya tarea se ejecuta en esta fase del proceso. Si el diseño se realiza de una manera detallada esta codificación se producirá de manera

[11] Estas fases o etapas no son exclusivas de este ciclo de vida o modelo: son comunes al resto de métodos y, aunque cada una introduce matices propios, la esencia es la misma.

mecánica, mientras que si el diseño ha sido pobre el proceso de codificación será más laborioso y más propenso a errores. El resultado de esta fase es lo que comúnmente se denomina “código” o genéricamente “programa”.

Prueba (o Testing): Una vez que se ha codificado el software comienza la prueba del mismo. Dicha prueba se enfoca sobre la lógica interna del software, asegurando que todas las sentencias se han probado, y sobre las funciones externas, esto es, realizando pruebas para asegurar que la entrada definida producirá los resultados que realmente se requieren.

Utilización: Una vez superada la fase de pruebas, el software se entrega al cliente y comienza la vida útil del mismo. La fase de utilización se solapa con las posteriores - el mantenimiento y la sustitución - y dura hasta que el software, ya sea reemplazado por otro o deje de utilizarse.

Mantenimiento: El software sufrirá indudablemente cambios después de que éste sea entregado al cliente. Los cambios ocurrirán debido a:

- Que se han encontrado errores (*Mantenimiento Correctivo*)
- Que el software debe adaptarse por cambios del entorno externo (*Mantenimiento Evolutivo*)
- Que se requiere nuevas funcionales o de mejoras de rendimiento.

En cualquier caso, el mantenimiento supone volver atrás en el ciclo de vida, a las etapas de codificación, diseño o análisis dependiendo de la magnitud del cambio.

Sustitución: La vida del software no es ilimitada y cualquier aplicación, por buena que sea, acaba por ser sustituida por otra más amplia, rápida, bonita y fácil de usar.

La sustitución de un software que está funcionando por otro que acaba de ser desarrollado es una tarea que hay que planificar cuidadosamente y que hay que llevar a cabo de forma organizada.

Es conveniente realizarla por fases, si esto es posible, no sustituyendo todas las aplicaciones de golpe, puesto que la sustitución conlleva normalmente un aumento detrabajo para los usuarios, que tienen que acostumbrarse a las nuevas aplicaciones (*labor no menor y que en muchos casos la asume el área de Gestión del Cambio de los proyectos o depto. involucrados*), y también para los implementadores, que tienen que corregir los errores que aparecen. Es necesario hacer un trasvasije de la información que maneja el sistema viejo a la estructura y el formato requeridos por el nuevo (*Otra labor muy importante y de ruta crítica en proyectos de gran envergadura*)

Además, en ocasiones, es conveniente mantener los dos sistemas funcionando en paralelo durante algún tiempo para comprobar que el sistema nuevo funcione correctamente y para asegurarnos el funcionamiento normal de la empresa, aún en el caso de que el sistema nuevo falle y tenga que volver a alguna de las fases de desarrollo.

Algunas de las ventajas de este modelo son su fácil implementación, entendimiento y gestión, es ampliamente usado en la mayoría de los proyectos estándares de desarrollo de software, promueve “buenas prácticas”, por ejemplo: definir antes de diseñar y la documentación es inherente a cada fase.

Algunos inconvenientes de este modelo es que incorpora iteraciones en forma no explícita lo que puede producir confusiones en el desarrollo de un proyecto, la participación del cliente se realiza sólo en algunas fases del ciclo, lo que da espacio para una gran variedad de malas interpretaciones, además de ser poco realista al esperar que los requerimientos sean claramente especificados en etapas muy tempranas del desarrollo.

Otros problemas de este modelo son el particionamiento inflexible de las fases en las que se divide el proyecto software, la dificultad de responder a nuevas necesidades del cliente. Este proceso sólo es válido para el caso ideal en el que los requerimientos están exactamente definidos y detallados desde el principio. La versión 2 del modelo, al tratarse de un modelo iterativo, refleja mejor la realidad del desarrollo software.

2.2.4 Modelo en Base a Prototipos

Dos de las críticas que se hacían al modelo de ciclo de vida en cascada eran que es difícil tener claros todos los requisitos del sistema al inicio del proyecto, y que no se dispone de una versión operativa del programa hasta las fases finales del desarrollo, lo que dificulta la detección de errores y deja también para el final el descubrimiento de los requisitos inadvertidos en las fases de análisis. Para paliar estas deficiencias se ha propuesto un modelo de ciclo de vida basado en la construcción de prototipos.

En primer lugar, hay que ver si el sistema que tenemos que desarrollar es un buen candidato a utilizar el paradigma de ciclo de vida de construcción de prototipos o al modelo en espiral. En general, cualquier aplicación que presente mucha interacción con el usuario, o que necesite algoritmos que puedan construirse de manera evolutiva, yendo de lo más general a lo más específico, es una buena candidata. No obstante, hay que tener en cuenta la complejidad, si la aplicación necesita que se desarrolle una gran cantidad de código para poder tener un prototipo que enseñar al usuario, las ventajas de la construcción de prototipos se verán superadas por el esfuerzo de desarrollar un prototipo que al final habrá que desechar o modificar mucho. También hay que tener en cuenta la disposición del cliente para probar un prototipo y sugerir modificaciones de los requisitos. Puede ser que el cliente ‘no tenga tiempo para andar jugando’ o ‘no vea las ventajas de este método de desarrollo’.

También es conveniente construir prototipos para probar la eficiencia de los algoritmos que se van a implementar, o para comprobar el rendimiento de un determinado componente del sistema, por ejemplo, una base de datos o el soporte hardware, en condiciones similares a las que existirán durante la utilización del sistema.

Es bastante frecuente que el producto de ingeniería desarrollado presente un buen rendimiento durante la fase de pruebas realizada por los ingenieros antes de entregarlo al cliente (pruebas que se realizarán normalmente con unos pocos registros en la base de datos o un único terminal conectado al sistema), pero que sea muy ineficiente, o incluso inviable, a la hora de almacenar o procesar el volumen real de información que debe manejar el cliente. En estos casos, la construcción de un

prototipo de parte del sistema y la realización de pruebas de rendimiento, sirven para decidir, antes de empezar la fase de diseño, cuál es el modelo más adecuado de entre la gama disponible para el soporte hardware o cómo deben hacerse los accesos a la base de datos para obtener buenas respuestas en tiempo cuando la aplicación esté ya en funcionamiento.

En otros casos, el prototipo servirá para modelar y poder mostrar al cliente cómo va a realizarse la E/S[12]de datos en la aplicación, de forma que éste pueda hacerse una idea de cómo va a ser el sistema final, pudiendo entonces detectar deficiencias o errores en la especificación aunque el modelo no sea más que una cáscara vacía.Según esto, un prototipo puede tener alguna de las tres formas siguientes:

- En papel o ejecutable en ordenador, que describa la interacción hombre-máquina y los listados del sistema.
- Que implemente algún(os) subconjunto(s) de la función requerida, y que sirva para evaluar el rendimiento de un algoritmo o las necesidades de capacidad de almacenamiento y velocidad de cálculo del sistema final.
- Un programa que realice en todo o en parte la función deseada pero que tenga características (rendimiento, consideración de casos particulares, etc.) que deban ser mejoradas durante el desarrollo del proyecto.

La figura 2.6muestra la secuencia de tareas de este paradigma.

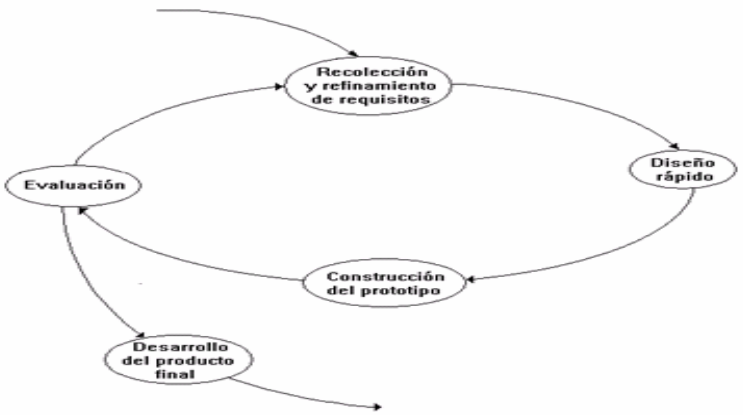


Figura. 2.6. Construcción de prototipos

[12]Sigla para referirse a las Entradas y Salidas de una aplicación

No obstante, hay que tener en cuenta que el prototipo no es el sistema final, puesto que normalmente apenas es utilizable. Será demasiado lento, demasiado grande, inadecuado para el volumen de datos necesario, contendrá errores (debido al diseño rápido), será demasiado general (sin considerar casos particulares, que debe tener en cuenta el sistema final) o estará codificado en un lenguaje o para máquinas inadecuadas, o a partir de componentes software previamente existentes. No hay que preocuparse de haber desperdiciado tiempo o esfuerzos construyendo prototipos que luego serán desechados, si con ello hemos conseguido tener más clara la especificación del proyecto, puesto que el tiempo perdido lo ahorraremos en las fases siguientes, que podrán realizarse con menos esfuerzo y en las que se cometerán menos errores que nos obliguen a volver atrás en el ciclo de vida.

Hay que tener en cuenta que un análisis de requisitos incorrecto o incompleto, cuyos errores y deficiencias se detecten a la hora de las pruebas o tras entregar el software al cliente, nos obligará a repetir de nuevo las fases de análisis, diseño y codificación, que habíamos realizado cuidadosamente, pensando que estábamos desarrollando el producto final. Al tener que repetir estas fases, estaremos desechando una gran cantidad de trabajo, normalmente muy superior al esfuerzo de construir un prototipo basado en un diseño rápido, en la reutilización de trozos de software preexistentes y en herramientas de generación de código para informes y manejo de ventanas.

Uno de los problemas que suelen aparecer, siguiendo el paradigma de construcción de prototipos, es que con demasiada frecuencia el prototipo pasa a ser parte del sistema final, bien sea por presiones del cliente, que quiere tener el sistema funcionando lo antes posible o bien porque los técnicos se han acostumbrado a la máquina, el sistema operativo o el lenguaje con el que se desarrolló el prototipo. Se olvida aquí que el prototipo ha sido construido de forma acelerada, sin tener en cuenta consideraciones de eficiencia, calidad del software o facilidad de mantenimiento, o que las elecciones de lenguaje, sistema operativo o máquina para desarrollarlo se han hecho basándose en criterios como el mejor conocimiento de esas herramientas por parte los técnicos que en que sean adecuadas para el producto final.

El utilizar el prototipo, en el producto final, conduce a que éste contenga numerosos errores latentes, sea ineficiente, poco fiable, incompleto o difícil de mantener. En definitiva a que tenga poca calidad, y eso es precisamente lo que queremos evitar aplicando la ingeniería del software y un modelo de calidad de software que garantice la calidad del producto final.

2.2.5 Modelo de Desarrollo Evolutivo en Espiral

Los modelos evolutivos parten del concepto de que el software, al igual que todos los sistemas complejos, evoluciona con el tiempo. Los condicionantes del producto pueden variar durante el proceso, y éste debe responder eficientemente a la nueva situación.

Estos modelos son repetitivos (iterativos) y se caracterizan por la forma en la que permiten a los ingenieros del software desarrollar versiones cada vez más completas del sistema.

Como es de suponer, existen varios esquemas que dan soporte al concepto del modelo evolutivo, siendo, seguramente, el modelo en Espiral el más conocido de ellos.

El modelo en espiral, indicado en la Fig. 2.7, combina las principales ventajas del modelo de ciclo de vida en cascada y del modelo de construcción de prototipos. Proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos, mucho más realista que el ciclo de vida clásico, y permite la utilización de prototipos en cualquier etapa de la evolución del proyecto.

Este es un modelo relativamente nuevo (fue propuesto por Barry Boehm[13]a fines de los ‘80) y no ha sido tan usado como los dos anteriores, aunque es de esperar que se extienda cada vez más.

Otra característica de este modelo es que incorpora en el ciclo de vida el análisis de riesgos. Los prototipos se utilizan como mecanismo de reducción del riesgo,

13Barry Boehm, Pdh Profesor Emérito de Ingeniería de Software en el Departamento de Ciencias de la Computación de la Universidad de California, y conocido por sus muchas contribuciones a la ingeniería de software.

permitiendo finalizar el proyecto antes de haberse embarcado en el desarrollo del producto final, si el riesgo es demasiado grande.

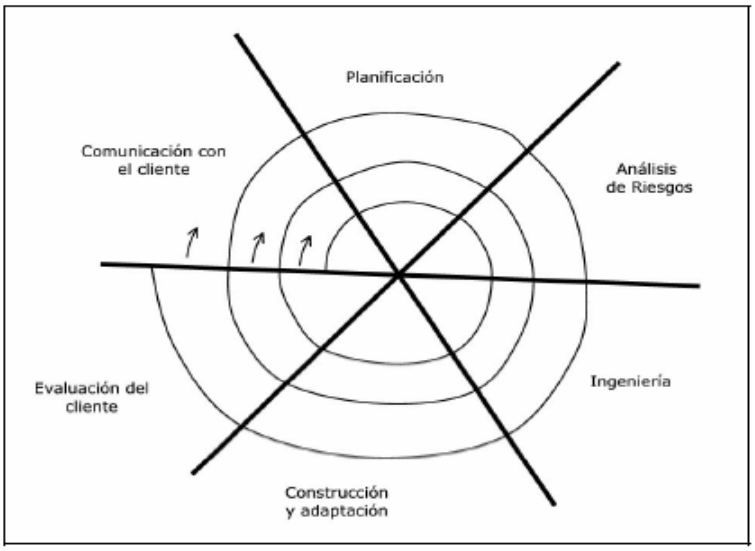


Figura. 2.7. Modelo evolutivo Espiral

Este modelo, que basa su proceso en el desarrollo rápido de versiones incrementales del software a implementar, se divide en una serie de regiones de tareas (o actividades de trabajo). Éstas son:

- **Comunicación con el cliente**: Agrupa los aspectos de la comunicación entre el desarrollador y el cliente.
- **Planificación**: Definición de los recursos materiales, humanos y temporales del proyecto.
- **Análisis de riesgos**: Evaluación y gestión de los posibles riesgos que pueden aparecer durante el desarrollo.
- **Ingeniería:** Construcción de representaciones o prototipos del sistema.
- **Construcción y acción:** Incluye todo lo relacionado con el desarrollo, prueba, instalación y soporte al usuario.
- **Evaluación del cliente:** Obtención de las reacciones del cliente.

Este modelo mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida en cascada, pero lo incorpora al marco de trabajo iterativo que refleja de forma más sensata el mundo real. Algunas de sus ventajas serian:Más realistas: el modelo refleja la naturaleza iterativa del desarrollo de proyectos con requerimientos no claros, Flexible: incorpora las ventajas del modelo cascada y de prototipos y fuerte enfoque en la disminución de riesgos.

Los principales problemas de este modelo son la falta de visibilidad del proceso, una pobre estructuración y unos requerimientos técnicos muy específicos, como por ejemplo los lenguajes de prototipado rápido. Su éxito descansa en la experiencia y habilidades de quienes evalúan los riesgos, No siempre bienvenido como elemento de contrato con clientes debido a su naturaleza cambiante. Exige un mayor esfuerzo de gestión.

2.2.6 Modelo de Desarrollo Evolutivo WinWin.

El modelo de desarrollo evolutivo, conocido con el nombre WinWin, que fue propuesto por B. BOHEM, es visto como una variación o una evolución del modelo en espiral que, como hemos visto, utiliza una aproximación cíclica para el desarrollo incremental de sistemas software.

WinWin literalmente significa “Ganar Ganar” y hace referencia a que el desarrollo del proceso software se basa en una constante negociación entre el cliente y el desarrollador en busca del beneficio mutuo y constante, o sea, el cliente quiere “ganar” y el desarrollador también quiere “ganar”, por lo que el centro de la negociación entre ambos adquiere una especial relevancia en la fase de los requisitos del sistema.

Cada ciclo envuelve cuatro actividades principales:

- Elaboración del sistema o subsistemas y los objetivos, restricciones y alternativas del proceso.
- Evaluar las alternativas respecto a los objetivos y restricciones. Identificar y resolver el mayor número de fuentes de producto y riesgos posibles.
- Elaboración de la definición del producto y del proceso.

Planificación del próximo ciclo y actualización de la planificación del ciclo de vida. Ello incluye, si es necesario, el particionamiento del sistema en subsistemas a desplegarse en paralelo. Puede incluir, además, la definición de un plan para finalizar el proyecto si éste es de riesgo demasiado alto o no es factible.

Una dificultad proviene de contestar a la pregunta ¿cómo se elaboran los objetivos, las restricciones y las alternativas?.En el modelo WinWin, como lo expresa la Fig. 2.8, ésta tarea se consigue identificandoa los principales implicados y estableciendo sus condiciones para “ganar con el sistema”. Este concepto es una incorporación importante al modelo en espiral, puesto que si no se determinan éstas condiciones podrá desarrollarse un producto funcionalmente correcto pero totalmente insatisfactorio.

Otro aspecto importante que este modelo añade al modelo en espiral, es la introducción de una serie de hitos tangibles hacia el proceso software. BOHEM describe tres hitos críticos:

a) Objetivos del Ciclo de Vida (OCV)
b) Arquitectura del Ciclo de Vida (ACV)
c) Capacidad Operacional Inicial (COI)

Los OCV ayudan a describir los objetivos del sistema y conducen a la ACV, sirviendo también para elaborar los objetivos iníciales. Los OCV identifican además los objetivos del sistema y la arquitectura muestra lo lejos que estamos de dichos objetivos, además de indicarnos los COI que nuestra infractuctura tecnológica debe soportar inicialmente.

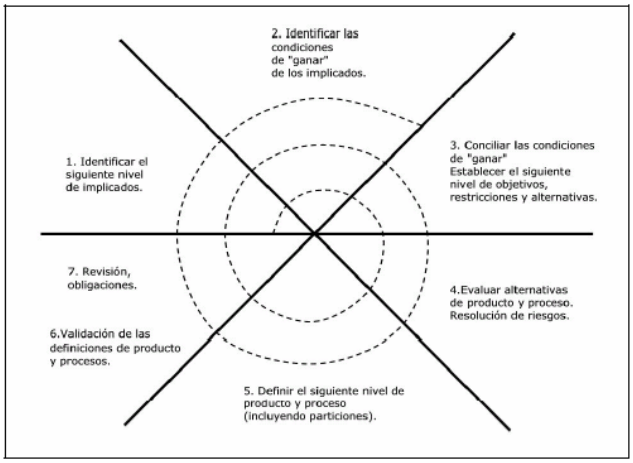


Figura. 2.8. Modelo evolutivo WinWin

2.2.7 Modelos Iterativo e Incremental

Es un modelo, en el cual se divide el esfuerzo de desarrollo de un proyectode software en partes más pequeñas o mini proyectos. Cada mini proyecto es unaiteración que resulta en un incremento. Las iteraciones hacen referencia a pasosen el flujo de trabajo, y los incrementos a crecimientos en el producto.

Este modelo esta derivado del ciclo de vida en cascada clásico, su principal finalidad es reducir elriesgo que surge entre las necesidades del usuario y el producto final por malos entendidos o errores durante la etapa de levantamiento de requerimientos.

Es la iteración de varios ciclos de vida en cascada. Al final de cada iteración se le entregaal cliente una versión mejorada o con mayores funcionalidades del producto, tal como se muestra en la figura 2.9. Elcliente es quien luego de cada iteración, evalúa el producto y lo corrige o propone mejoras.

Es un modelo ideal a seguir cuando el usuario necesitaentregas rápidas aunque el proyecto no esté terminado.

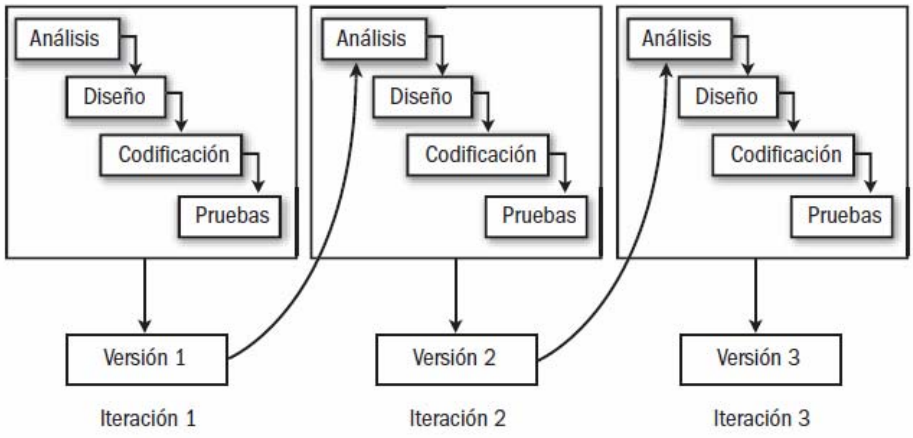


Figura. 2.9. Modelo iterativo e Incremental

Algunos de los beneficios del enfoque iterativo e incremental se listan a continuación:

- La iteración controlada reduce el riesgo de pérdida en el valor del productocompleto, es decir, solo se perderá el esfuerzo de una iteración.
- El cliente se involucra, teniendo la oportunidad de probar y dar su opiniónen el avance del proyecto.
- El progreso se puede medir en periodos cortos de tiempo.
- Las pruebas del software y la integración son constantes.
- Además si existieran errores producidos en un incremento se puedensolucionar en el próximo incremento.

En esta sección hemos querido indagar sobre algunos modelos o ciclos de vida más usados dentro de la ingeniería de software, la utilización o elección del ciclo de vida o modelo va a depender del tipo de proyecto a desarrollar y la orientación del mismo, además puede surgir la necesidad de utilizar ciclos de vida combinados o modificar algún ciclo de vida o modelo a conveniencia del proyecto, pero lo fundamental es establecerlo, ya que esto permitirá al líder del proyecto y su equipo organizase de mejor forma para enfrentar el proyecto.

2.3 Calidad de Software

Dentro de la ingeniería de software, se han desarrollado modelos o estándares cuyo propósito es ser una guía en el desarrollo y evaluación de productos de calidad.

En la literatura podemos encontrar que calidad adquiere distintos significados para autores diferentes, no existe una definición universal. El diccionario "American Heritage"[14] define calidad como "una característica o atributo de algo", de lo que podemos desprender que la calidad de un producto o servicio puede medirse en base a sus características, lo cierto es que lo que puede ser calidad para una persona puede no serlo para otra, así la concepción que puede tener un desarrollador de la calidad de un producto de software puede ser distinta a la percepción que tiene el cliente final.

Para unificar criterios y definiciones diremos que la calidad del software estará dada por:

"La concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software"[15]

Dicho lo anterior pasaremos a explicar más en detalle algunos antecedentes históricos y evolutivos de la calidad de software, algunos modelos y su influencia en el modelo de calidad planteado en esta investigación.

2.3.1 Antecedentes Históricos

Desde finales del siglo XIX, con la revolución industrial, muchos fabricantes incipientes de la época quisieron otorgar calidad a sus productos, pero la calidad como tal era entendida en el sentido de otorgar uniformidad al producto final, para ello lo que se tenían eran departamentos de inspección donde se detectaban los problemas en los

[14] Diccionario inglés americano y el español latino, este diccionario incluye cerca de 2.500 palabras y frases.
[15] Roger S. Pressman, Ingeniería Del Software, un Enfoque Practico (Quinta Edición)

productos ya terminados y según las evaluaciones que se hacían estos se aprobaban o rechazaban, los productos que resultaban defectuosos simplemente se eliminaban.

A partir de la década de 1930 y hasta 1950 el enfoque de calidad varió gracias al aporte de Walter Shewart[16] y Edward Deming[17,] ellos desarrollan controles estadísticos para los procesos de producción, una de las primeras aplicaciones fue hecha en los laboratorios Bell en los Estados Unidos, pero con el surgimiento de la segunda guerra mundial se hizo masivo.

El concepto de Aseguramiento de la Calidad (QA, Quality Assurance) surgió a mediados del siglo pasado pero recién fue adoptado en la década de 1980, este concepto a diferencia de los dos anteriores estaba enfocado a la prevención de los problemas y hacía énfasis en los ciclos completos de la producción pero con la contribución de los grupos funcionales de cada etapa, estos grupos funcionales eran especializados en la calidad, su auge se debió a que obtuvo un apoyo muy fuerte en los sistemas de información, que ya habían aparecido para entonces, y que brindaban una manera más eficiente de gestión. Paralelamente, en el Japón, surgió una corriente distinta alejada de los conceptos occidentales, que adoptó el concepto de Gestión Total de la Calidad,la cual comprometía a todo el personal de la organización, en las tareas de aseguramiento de la calidad.

No fue sino hasta después de 1980 que el concepto de Gestión Estratégica de la Calidad apareció; fue producto de un acercamiento de los conceptos occidental y japonés, y se enfocó en la consecución de la calidad como una medida de competitividad de las organizaciones. La globalización de finales del siglo pasado hizo que las organizaciones colocaran más énfasis en las necesidades de los diferentes mercados y por consecuencia en las necesidades de sus clientes, y que la calidad del producto fuera determinada por el cumplimiento de estas necesidades, para lograr lo anterior la estrategia que adopta una organización es muy importante; el

[16] Fue un físico, ingeniero y estadístico, conocido como el padre del control estadístico de calidad.

[17] Edward Deming, un estadista, profesor y fundador de la Calidad Total. Ignorado por las corporaciones americanas, Deming fue a Japón en 1950 a la edad de 49 y enseñó a los administradores, ingenieros y científicos Japoneses como producir calidad

establecimiento de metas claras y el compromiso de toda la organización al logro de las mismas son los pilares donde descansa esta nueva forma de calidad.

2.3.2 Concepto de Calidad en la Actualidad

En la actualidad los conceptos de calidad son variados, pero como veremos todos tienen, como enfoque, la competitividad que alcanza la organización al ofrecer productos o servicios de calidad y hacen siempre énfasis en satisfacer las necesidades de los mercados y clientes.

La Organización Internacional de Estándares (ISO, por sus siglas eninglés) ha publicado hasta hoy varios estándares relacionados con la calidad engeneral y también de manera particular con la calidad de software, de todas lasnormas ISO publicadas, ISO 9000 corresponde al estándar de gestión de calidadampliamente aceptado y que ha sido la base para otras normas más específicas, como son:ISO 9000 [ISO9000, 2000], que conceptualiza la calidad como "**El grado en el que unconjunto de características inherentes cumple con los requisitos**", a su vez para lanorma ISO 8402 [ISO8402, 1994],define calidad como: "***Totalidad de características de un producto que le confieren su aptitud para satisfacer unas necesidades expresadas o implícitas***", debe entenderse que para esta norma laexpresión, "**necesidades expresadas o implícitas**", se refiere a que lasnecesidades pueden ser definidas por un contrato o son inherentes a lafuncionalidad del producto.

2.3.3 Sistemas de Calidad

Según la norma ISO 9000 [ISO9000, 2000], un sistema de calidad "**es la estructura organizativa, las responsabilidades, los procedimientos, los procesos y los recursos necesarios para llevar a cabo la gestión de la calidad**".

2.3.4 Contexto Básico de un Sistema de Calidad

Se puede decir que actualmente los sistemas de calidad estánenfocados en cinco principios básicos, donde todos tienen participación porigual y son imprescindibles para los objetivos de calidad planteados porcualquier organización, estos son:

a. **Enfoque en el Cliente**; actualmente es aceptado que la calidadestádefinida por los clientes, así, quienes determinan si un productoo servicio debe ser aceptado y satisfacer las necesidades para lasque fue desarrollado son los **clientes**, y no controles de calidad posteriores, en conclusión al concebir un servicio o producto, sedebe pensar ya en la necesidad que va a satisfacer, y en loscriterios que los futuros clientes o usuarios tendrán en cuenta paraadquirirlo. El objetivo principal es lograr la máxima satisfacción delos clientes.

b. **Compromiso**; el compromiso total de toda la organización enlograr todos los objetivos de calidad es muy importante, si bien laalta dirección posee un liderazgo activo, todos los miembros de laorganización deben saber y comprender que su labor es tambiénmuy importante pues son ellos los generadores de calidad en losproductos o servicios que se ofrecen.

c. **Medición**; es necesario tener un patrón a partir del cual se puedatener un seguimiento de los niveles de calidad, estas medicionespermiten tener idea de los niveles de defectos con respecto a estándares mínimos preestablecidos y con su estudio y análisisayudan a mejorar la calidad constantemente.

d. **Comunicación y Reconocimiento**; los resultados de la aplicaciónde las políticas de calidad deben ser comunicados a todos losmiembros de la organización, los logros particulares y los másdestacados deben ser reconocidos y estimulados, de este modo secrea una conciencia del valor que posee el trabajo en la generaciónde calidad y a la vez se estimula a realizar las cosas cada vezmejor.

e. **Mejora Continua;** el proceso de calidad es un ciclo que tiene un inicio pero que no tiene fin, con el fin de cada proceso se deben buscar los errores y

analizar la manera de hacerlo mejor en la próxima iteración, aún cuando todo parezca bien al final se deben buscar maneras de añadir cualidades o características de diferenciación al producto o servicio.

2.3.5 Concepto de Calidad de Software

Para entender el concepto de calidad aplicado al software es necesario detenerse un instante y repasar los conceptos volcados al principio de esta sección, sobre las particularidades del software y la calidad del mismo, así entenderemos de mejor forma que es el software como producto y las implicancias que se desprenden de la manera particular en que es desarrollado.

Para nadie, que esté comprometido con las tecnologías de la información y la ingeniería de software, es un secreto que el software es un producto que posee características muy especiales, al final del proceso de desarrollo de software lo que se obtiene es un producto que a diferencia de la mayoría de los productos conocidos "no se gasta con el uso y repararlo no significa una restauración a su estado original sino corregir defectos que estaban desde el momento de su entrega y que deben ser solucionados en la etapa de mantenimiento" [Piattini,1996][18].

A inicios de la pasada década de los 90's, el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en inglés) publicó un Diccionario de Computación como parte de su estándar IEEE 610 - 1990 y definía al software como:

"Los programas de ordenador, los procedimientos y, posiblemente la documentación asociada y los datos relativos a la operación de un sistema informático", además definió como calidad **"el grado con el que un sistema componente o proceso cumple con los requisitos específicos y las necesidades o expectativas del cliente o usuario".**

En 1991, la Organización Internacional de Estándares (ISO) y la ComisiónInternacional Electrotécnica (IEC), editaron de manera conjunta la normainternacional ISO/IEC 9126 [ISO/IEC-9126, 1991] que define calidad de softwarecomo: "**la totalidad de características de un producto de software que le confiere la capacidad de satisfacer necesidades explícitas e implícitas**", al respecto, sedebe agregar que las necesidades explícitas son los alcances y los

[18] Mario Piattini Velthuis , MSc and PhD in Computer Science Madrid Technical University

objetivospropuestos por quienes producen y desarrollan el software, por lo tanto sonfactores relativos a la calidad durante el proceso de desarrollo del software ysolamente es percibida por aquellos que tuvieron participación en este proceso.Necesidades implícitas, en cambio, son necesidades subjetivas de los usuarios,pero que pueden ser apreciadas tanto por los usuarios como por losdesarrolladores, a estas necesidades se les llama comúnmente calidad de uso.

Como aporte adicional, en 1993 Roger S. Pressman[19] definió lacalidad de software como: "***La concordancia del software con los requisitos explícitamente establecidos, con los estándares de desarrollo expresamente fijados y con los requisitos implícitos, no establecidos formalmente pero que desea el usuario***".

2.3.6 Puntos de Vista de la Calidad del Software

En la ingeniería del software o del conocimiento, la visión de la calidad no es única, ya que dependerá del punto de vista desde el cual se le analice, como se señala en la figura 2.9:

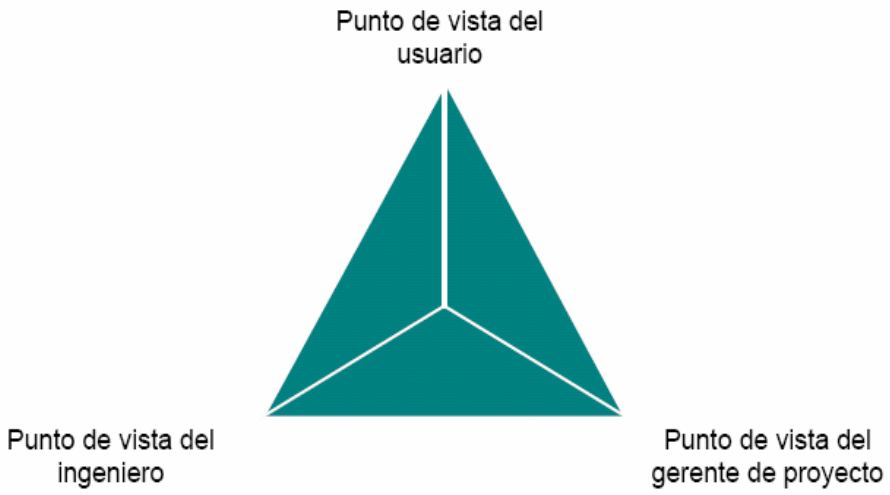


Figura. 2.9. Puntos de vista de la calidad de software

[19] Roger S. Pressman, Ingeniería Del Software, un Enfoque Practico (Quinta Edición)

Dependiendo del punto de vista, tal como se observa en la figura 2.9, se priorizarán distintos factores del software, con respecto a la calidad:

- **Punto de vista del usuario**: estará basado en los factores externos del producto, tales como funcionalidad y facilidad de operación o uso.
- **Punto de vista del ingeniero de software**: estará basado en los factores internos del producto, tales como modularidad y reusabilidad.
- **Punto de vista del gerente del proyecto:** El punto de vista del gerente del proyecto, con respecto a la calidad, estará basado en los factores relacionados con la gestión del proyecto, tales como costos y cronogramas acorde a lo planificado.

2.3.7 Factores que Determinan la Calidad del Software

Existen muchos factores que afectan a la calidad del software y se pueden clasificar de distintas formas. En este trabajo se presentarán sólo a modo descriptivo, algunos factores de calidad que se han propuesto.

McCall, Richards y Walters[20]han propuesto los siguientes:

- **Corrección**. El grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente, responde a la pregunta: ¿Hace lo que quiero?.
- **Fiabilidad.** El grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida (Hay que decir que se han propuesto otras definiciones de fiabilidad más completas), responde a la pregunta: ¿Lo hace en forma fiable todo el tiempo?.
- **Eficiencia**. La cantidad de recursos de computadora y de código requeridos por un programa para llevar a cabo sus funciones, responde a la pregunta: ¿Se ejecutará en mi hardware lo mejor que se pueda?.

[20] McCall J A, Richards PK y Walters GF; *Factors in software quality, Vols I,II,III*; US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.

- **Integridad.** El grado en que puede controlarse el acceso al software o a los datos, por personal no autorizado, responde a la pregunta: ¿Es seguro?.
- **Facilidad de uso**. El esfuerzo requerido para aprender un programa, trabajar con él, preparar su entrada e interpretar su salida, responde a la pregunta: ¿Está diseñado para ser usado?.
- **Facilidad de mantenimiento**. El esfuerzo requerido para localizar y arreglar un error en un programa (Se trata de una definición muy limitada), responde a la pregunta: ¿Permite ser corregirlo con relativa facilidad?.
- **Flexibilidad.** El esfuerzo requerido para modificar un programa operativo, responde a la pregunta: ¿Permite ser cambiado con relativa facilidad?.
- **Facilidad de prueba**. El esfuerzo requerido para probar un programa de forma que se asegure que realiza su función requerida, responde a la pregunta: ¿Permite ser probado con relativa facilidad?.
- **Portabilidad.** El esfuerzo requerido para transferir el programa desde un hardware y/o un entorno de sistemas de software a otro, responde a la pregunta: ¿Podré usarlo en otra computadora?.
- **Reusabilidad**. El grado en que un programa (o partes de un programa) se puede reutilizar en otras aplicaciones. Esto va relacionado con el empaquetamiento y el alcance de las funciones que realiza el programa, responde a la pregunta: ¿Podré reutilizar alguna parte del software?.
- **Facilidad de interoperación**. El esfuerzo requerido para acoplar un sistema a otro, responde a la pregunta: ¿Podré hacerlo interactuar con otros sistemas?.

Es difícil, y en algunos casos imposibles, desarrollar medidas directas de los factores de calidad anteriormente descritos. Por lo tanto, cada factor se descompone en atributos o criterios, más fácilmente de ser medibles. Cabe aclarar que cada uno de estos atributos puede corresponder a más de un factor de calidad, estos son:

- **Facilidad de auditoría**. La facilidad con que se puede comprobar la conformidad con los estándares.
- **Exactitud**. La precisión de los cálculos y del control.
- **Normalización de las comunicaciones**. El grado en que se usan el ancho de banda, los protocolos y las interfaces estándar.
- **Completitud**. El grado en que se ha conseguido la total implementación de las funciones requeridas.
- **Concisión.** Lo compacto que es el programa en términos de líneas de código.
- **Consistencia**. El uso de un diseño uniforme y de técnicas de documentación a lo largo del proyecto de desarrollo del software.
- **Estandarización en los datos**. El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
- **Tolerancia de errores**. El daño que se produce cuando el programa encuentra un error.
- **Eficiencia en la ejecución**. El rendimiento en tiempo de ejecución de un programa.
- **Facilidad de expansión.** El grado en que se puede ampliar el diseño arquitectónico, de datos o procedimental.
- **Generalidad**. La amplitud de aplicación potencial de los componentes del programa.
- **Independencia del hardware**. El grado en que el software es independiente del hardware sobre el que opera.
- **Instrumentación.** El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.
- **Modularidad**. La independencia funcional de los componentes del programa.
- **Facilidad de operación**. La facilidad de operación de un programa.
- **Seguridad.** La disponibilidad de mecanismos que controlen o protejan los programas o los datos.

- **Auto-documentación.** El grado en que el código fuente proporciona documentación significativa.
- **Simplicidad.** El grado en que un programa puede ser entendido sin dificultad.
- **Independencia del sistema de software**. El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno.
- **Facilidad de traza**. La posibilidad de seguir la pista a la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos.
- **Formación**. El grado en que el software ayuda para permitir que nuevos usuarios apliquen el sistema.

Otra lista de factores de calidad de software es la desarrollada por Robert Grady y sus colaboradores[21]. Los factores y sus atributos correspondientes son los siguientes:

- **La funcionalidad** se obtiene mediante la evaluación del conjunto de características y de posibilidades del programa, la generalidad de las funciones que se entregan y la seguridad de todo el sistema.
- **La facilidad de uso** se calcula considerando los factores humanos, la estética global, laconsistencia y la documentación.
- **La fiabilidad** se calcula midiendo la frecuencia de fallos y su importancia, la eficacia delos resultados de salida, el tiempo medio entre fallos, la posibilidad derecuperarse a los fallos y la previsibilidad del programa.
- **El rendimiento** se mide mediante la evaluación de la velocidad de proceso, el tiempo derespuesta, el consumo de recursos, el rendimiento total de procesamiento y la eficiencia.
- **La capacidad de soporte** combina la posibilidad de ampliar el programa (extensibilidad),la adaptabilidad y la utilidad (estos tres atributos representan

21 R. Grady y D. Caswell (1987). Desarrolladores del modelo de calidad de software FURPS+ para Hewlett Packard

un término más común —facilidad de mantenimiento), además de la facilidad de prueba, la compatibilidad, laposibilidad de configuración [posibilidad de organizar y controlar elementos de laconfiguración & software], la facilidad con la que se puede instalar un sistema y lafacilidad con la que se pueden localizar los problemas.

Además quisiéramos mencionar los factores de calidad de software expuesto por el estándar internacional**ISO/IEC 9126** (1991), este provee un marco de trabajo para laevaluación de la calidad del software, el cual puede ser aplicable a todo tipo de software.

Este estándar define seis factores que determinan la calidad del software:

1) **Funcionalidad**: Conjunto de atributos que se sostienen sobre la existencia de unconjunto de funciones y sus propiedades específicas. Las funciones son aquellasque satisfacen necesidades implícitas y establecidas.

2) **Fiabilidad**: Conjunto de atributos que se sostienen sobre la capacidad del softwarepara mantener su nivel de rendimiento bajo condiciones establecidas para unperíodo de tiempo establecido.

3) **Usabilidad**: Conjunto de atributos que se sostienen sobre el esfuerzo necesariopara el uso, y sobre la evaluación individual de tal uso, por un conjunto deusuarios implícitos o establecidos.

4) **Eficiencia**: Conjunto de atributos que se sostienen sobre la relación entre el nivelde rendimiento del software y la cantidad de recursos usados, bajo condicionesestablecidas.

5) **Mantenibilidad:** Conjunto de atributos que se sostienen sobre el esfuerzo necesariopara realizar modificaciones especificadas.

6) **Portabilidad:** conjunto de atributos que se sostienen sobre la habilidad delsoftware para ser transferido desde un entorno a otro.

Para cada factor se sugiere un conjunto de sub-características de calidad, las que sedefinen, según el estándar ISO/IEC 9126 (1991), de la siguiente forma:

1) Funcionalidad:

a. **Adecuación:** atributos del software que se sostienen sobre lapresencia y propiedad de un conjunto de funciones para tareas específicas.

b. **Exactitud**: atributos del software que se sostienen sobre laprovisión de resultados o efectos correctos y acordados.

c. **Interoperabilidad**: atributos del software que se sostienen en suhabilidad para interactuar con sistemas específicos.

d. **Adherencia a prescripciones**: atributos del software que hacen queel software se adhiera a regulaciones o convenciones o estándares legales relativosa la aplicación y a prescripciones similares.

e. **Seguridad**: atributos del software que se sostienen en su habilidad paraprevenir accesos no autorizados, ya sean accidentales o deliberados a programas odatos.

2) Fiabilidad

a. **Madurez:** atributos del software que se sostienen sobre la frecuencia defallas debido a faltas en el software.

b. **Tolerancia a Fallos:** atributos del software que se sostienen en suhabilidad para mantener un nivel específico de rendimiento en caso de que elsoftware falle o que su interfaz específica sea violada.

c. **Recuperabilidad:** atributos del software que se sostienen sobre lacapacidad de restablecer su nivel de rendimiento y recuperar los

datosdirectamente afectados en caso de falla, y sobre el tiempo y esfuerzo necesariopara ello.

3) **Usabilidad**

a. **Facilidad de comprensión**: atributos del software que sesostienen sobre los esfuerzos de los usuarios para reconocer el concepto lógico ysu aplicabilidad.

b. **Facilidad de aprendizaje**: atributos del software que se sostienensobre el esfuerzo de los usuarios para aprender su aplicación.

c. **Operatividad**: atributos del software que se sostienen sobre losesfuerzos de los usuarios para la operación y el control de la misma.

4) **Eficiencia**

a. **Eficiencia de tiempo:** atributos del software que se sostienensobre los tiempos de respuesta y procesamiento y sobre tasas de tiempo pararealizar su función.

b. **Eficiencia de recursos**: atributos del software que sesostienen sobre la cantidad de recursos usados y la duración de ellos en realizar sufunción.

5) **Mantenibilidad**

a. **Facilidad de análisis:** atributos del software que se sostienen sobreel esfuerzo necesario para diagnosticar deficiencias o causas de fallas, o para laidentificación de partes a ser modificadas.

b. **Facilidad para modificaciones**: atributos del software que sesostienen sobre el esfuerzo necesario para modificación, eliminación de faltas ocambio en el entorno.

c. **Estabilidad**: atributos del software que se sostienen sobre el riesgo deefectos inesperados de modificaciones.

d. **Facilidad de verificación**: atributos del software que se sostienensobre el esfuerzo necesario para validar el software modificado.

6) Portabilidad

a. **Adaptabilidad**: atributos del software que se sostienen sobre laoportunidad para su adaptación a entornos específicos diferentes sin aplicar otrasacciones o medios que aquellos dados para este propósito.

b. **Facilidad de instalación**: atributos del software que se sostienensobre el esfuerzo necesario para instalar el software en un entorno específico.

c. **Conformidad con prescripciones**: atributos del software que hacenque se ajusten a estándares o convenciones relativas a la portabilidad.

d. **Facilidad para reemplazo**: atributos del software que se sostienenen la oportunidad y el esfuerzo de usar este software en el lugar de otro softwareespecífico en el entorno de ese otro software.

Tanto los modelos **McCall**, **Grady**y el Estándar**ISO/IEC 9126**, presentan además fórmulas, matrices, pesos ponderados que permiten cuantificar cada uno de los factores presentados. Más allá de eso, los siguientes puntos deben quedar claros.

1) Los modelos aquí presentados son sólo una muestra de los disponibles, hay varios másy se actualizan frecuentemente.

2) Al planificar la calidad de un producto de software se debe seleccionar cuales de losfactores de calidad que van a ser considerados como

requisitos, a su vez. Para realizar estaselección, se debe tener en cuenta lo siguiente:

- Las características particulares de la aplicación a desarrollar o de su entorno. Asípor ejemplo, si la aplicación se desarrolla para un entorno en el que el hardwareevoluciona rápidamente, el factor “portabilidad” es importante; si se espera que las especificaciones del sistema cambien frecuentemente, la “flexibilidad” será esencial.
- El costo de los factores de calidad frente al beneficio que proporcionan, es decirrealizar un análisis costo-beneficio.
- Las interrelaciones entre factores: Algunos factores pueden ser conflictivos entre sí.La eficiencia, por ejemplo, está en conflicto con otros factores de calidad.

3) Es necesario medir cada uno de los factores y atributos seleccionados. Algunos sepueden ser medidos directamente y otros sólo pueden ser medidos indirectamente. En cualquiera de los dos casos la cuantificación es obligatoria. Respondiendo a otro de losprincipio del paradigma de la calidad (orientación a datos), las comparaciones se debenbasar sobre datos y mediciones concretas y objetivas, no sobre opiniones osubjetividades.

2.3.8 Evaluación de la Calidad en el Software

Por los conceptos ya explicados anteriormente se tiene claro que la**calidad del software** la define el **cumplimiento con los requisitos o requerimientos** para los que fuedesarrollado, pero surge la pregunta: ¿cómo asegurar de manera eficiente si unsoftware cumple con los requisitos explícitos e implícitos para los que fuecreado? Y casualmente esa es la principal dificultad que se enfrentan lasempresas desarrolladoras de software al intentar asegurar la calidad de susproductos, el software como ya se ha dicho, posee características muy diferentesa los productos comunes, normalmente en la manufactura de cualquier productouna de las medidas de calidad está dada por la

capacidad del producto asoportar el desgaste propio del uso, pero el software no es un producto que sedesgaste con el tiempo, por lo tanto estas formas de medición son inaplicables.

A continuación alguna de las formas más usuales de aseguramiento deCalidad de software:

2.3.8.1 Métricas de Calidad de Software

Durante el proceso de evaluación de la calidad por lo general se hacereferencia a las medidas del producto antes que a las medidas del proceso,una métrica constituye el valor de un atributo o una entidad, este es unconcepto de métrica muy genérico pero que también es aplicado al software,pero sin embargo las métricas suelen ser muy subjetivas, en la mayoría delos casos sólo miden algunas de las características de la calidad del softwarey dejan muchas cosas importantes sin ser medidas, suelen ser aplicadas pararealizar mediciones que se basan en la codificación del software, cantidad delíneas de código, comentarios, documentación del código, etc.

Otras métricas pueden basarse en relación con los flujos de control alinterior de los algoritmos, de modo que matemáticamente se pueden hacercálculos de los caminos independientes que puede seguir una rutina en unaaplicación, basado en la teoría de grafos, y que determina a su vez lasllamadas que se realizan entre rutinas o módulos, al final el objetivo es quelas métricas obtenidas nos permitan asegurar que el software es mantenible,es óptimo, entre otras características.

2.3.8.2 Verificación y Validación de Software

Una aspecto importante a tener en cuenta son las verificaciones yvalidaciones como parte del ciclo de vida en el proceso de desarrollo desoftware, éstas se utilizan para detectar fallas durante las etapas queconforman este ciclo, es decir, durante la especificación de los requisitos, enel análisis y diseño, desarrollo, e implementación. La verificación y validaciónson en realidad un conjunto de procedimientos y actividades que, basados en**técnicas específicas** y con la ayuda de **herramientas**, aseguran

durante elproceso de desarrollo que el software cumpla con el objetivo para el cual esconstruido.

En cada etapa se realizan pruebas de verificación y validaciónespecíficas que buscan detectar cuanto antes los defectos y corregirlos antesde pasar a la siguiente etapa. Estas verificaciones y validaciones al final noson más que pruebas de diferentes tipos acerca de las cuales se hará unacercamiento más detallado en un posterior capítulo.

2.3.8.3 Revisión de Software

El estándar ISO9000 – 2000 [ISO9000, 2000] indica de maneragenera l que para conseguir los objetivos de calidad se disponen de lossiguientes métodos:

Objetivo de Calidad	Métodos
Evaluación	Revisión
Verificación	Inspección
Validación	Pruebas
Confirmación de cumplimiento	Auditoria

Tabla. 1. Métodos de Revisión para los Objetivos de Calidad

Como se aprecia en la Tabla Nº 1, se definen objetivos de calidadbasados en evaluación, verificación, validación y confirmación decumplimiento con los requisitos, ISO9000 – 2000 define estos métodos paraasegurar su cumplimiento.

2.4 *CMMI*

Capability Maturity Model® Integration (CMMI) es un modelo propuesto para elmejoramiento de proceso que provee a las organizaciones con los elementos esencialesde procesos efectivos, la principal premisa de este modelo es que "***la calidad de un producto o de un sistema es en su mayor parte, es consecuencia de la calidad de los procesos empleados en su desarrollo y mantenimiento****".* CMMI puede ser personalizado por las organizaciones para queen base a sus necesidades específicas se puedan realizar procesos de mejora en unproyecto, división o en una organización.

CMMI ayuda a integrar las funciones uorganizaciones tradicionalmente separadas, definir objetivos y prioridades de unproceso de mejoramiento, proveer guías para la calidad de los procesos y proveerpuntos de referencia para la evaluación de los procesos actuales.

CMM nació en respuesta a la necesidad de organizaciones, en los Estados Unidos decontrolar los desarrollos de software contratados, que para 1982 tenían problemas detiempos que se alargaban y como consecuencia los presupuestos crecíandesmesuradamente. Se convocó a un concurso público para buscar la manera deresolver los problemas y en 1985 la Universidad Carnegie Mellon lo ganó y creo el**Software Engineering Institute** (SEI). El SEI es el instituto que mantiene el modelo decalidad CMM – CMMI.

CMMI provee las mejores prácticas para productos, desarrollo y mantenimiento deservicios. Los modelos CMMI permiten lo siguiente:

- Conectar de manera efectiva las actividades de gestión e ingeniería a los objetivosdel negocio.
- Expandir el alcance y visibilidad del ciclo de vida del producto y las actividades deingeniería para asegurar que el producto o servicio alcanzan las expectativas delcliente.
- Implementar prácticas de mayor madurez y robustas.

- Gestionar funciones críticas de la organización a sus productos o servicios.
- Cumplir mejor con los estándares relevantes de ISO.

2.4.1 El Cuerpo del Conocimiento CMMI

El objetivo de CMMI es completar a CMM, que no sólo contempla modelos dedesarrollo y mantenimiento, sino que proporciona un modelo de trabajo escalable.Actualmente hay cuatro cuerpos de conocimiento[22] cuando uno planea una mejora deprocesos usando CMMI:

- **Ingeniería de Sistemas**: cubre el desarrollo de sistemas integrales, los cuales nonecesariamente pueden incluir software. Se enfoca en transformar necesidades declientes, requerimientos, limitaciones en productos o soporte de estos productos.
- **Ingeniería de Software**: cubre el desarrollo de sistemas de software, orientándose aobjetivos específicos de procesos desarrollos de software, ordenados ycuantificables para el desarrollo, operación y mantenimiento de software.
- **Integración de Producto y Desarrollo de Procesos (IPPD):** objetivos sistematizadosque dependen de colaboraciones específicas con stakeholders[23] a lo largo de la vidadel producto para satisfacer necesidades de cliente y requerimientos.
- **Abastecimiento de Proveedores**: conforme crece el esfuerzo de trabajo este viene aser más complejo en donde se necesita afinar funciones o agregar modificaciones aproductos que son específicamente necesarios en proyectos. En muchascircunstanciases necesario actividades que cubran un amplio análisis y monitoreode los proveedores.

[22] Estos cuerpos de conocimiento también son llamados “disciplinas”, si el modelo necesita evolucionar o crecer es factible agregar más disciplinas o aéreas de conocimientos.

[23] Personas que tienen una fuerte relación con el proyecto, ya sea porque proporcionan datos al proyecto o porque reciben datos del mismo.

CMMI involucra los siguientes aspectos:

a) **Dos Representaciones: Continúa y Escalonada,** CMMI soporta dos caminos para el mejoramiento y a estos caminos se les conoce comorepresentaciones.

- **La Representación** Continua ofrece un camino con mayor flexibilidad para una mejorade procesos; permite a las organizaciones mejorar áreas de proceso individuales demanera incremental, para esta representación se aplica el termino Nivel de Capacidad.Los niveles de capacidad son usados para medir la mejora; desde procesos nogestionados hasta procesos optimizados.
- **La Representación Escalonada**permite a las organizaciones mejorarincrementalmenteun conjunto de áreas de procesos relacionadas, para esta representación se aplica eltermino Nivel de Madurez.

Nivel	Representación Continua Niveles de Capacidad	Representación Escalonada Niveles de Madurez
Nivel 0	Incompleto	N/A
Nivel 1	Ejecutado	Inicial
Nivel 2	Gestionado	Gestionado
Nivel 3	Definido	Definido
Nivel 4	Gestionado cuantitativamente	Gestionado cuantitativamente
Nivel 5	Optimizado	Optimizado

Tabla. 2. Comparación de niveles de capacidad y madurez

b) **Niveles de Capacidad:** El nivel de capacidad es un atributo de los procesos. El nivel de capacidad de unproceso indica si sólo se ejecuta, o si también se planifica y se encuentra organizativa yformalmente definido, se mide y se mejora de forma sistemática. Como se mencionóanteriormente, los niveles de capacidad se aplican al mejoramiento de procesos en lasorganizaciones en áreas de proceso individuales. Existen 6 niveles de capacidadnumerados del 0 al 5 y son los siguientes:

- **Incompleto o Nivel 0**.- Representa a un proceso incompleto que noes realizado o es parcialmente realizado. Una o más áreas de proceso no sonsatisfechas y no existen metas genéricas[24] para este nivel por ser un procesoparcialmente realizado.
- **Ejecutado o Nivel 1**.- Este es un proceso ejecutado que satisfacemetas específicas[25] del área de proceso. Soporta lo necesario para realizar el trabajo,más no para mantenerlo a lo largo del tiempo.
- **Gestionado o Nivel 2**.- Es un proceso que posee una estructurabásica para soportarlo. Es planeado y ejecutado de acuerdo con las políticas yprocedimientos.
- **Definido o Nivel 3**.- Es un proceso que ha sido definido de acuerdolos estándares y guías de procesos de la organización y contribuye con los resultadosy medidas de otros procesos.
- **Gestionado Cuantitativamente o Nivel 4**.- Es un proceso definidoque es controlado usando técnicas estadísticas y cuantitativas. Los objetivos de calidad y rendimiento del proceso son establecidos usando criterios deadministración de procesos.
- **Optimizado o Nivel 5**.- Es un proceso que ha mejorado en base alentendimiento de las variaciones mismas del proceso. El objetivo es optimizar elproceso constantemente a través de mejoras innovadoras e incrementales.

c) Niveles de Madurez: El modelo de calidad CMMI, en su representación escalonada, cataloga el modelo deprocesos de una empresa en niveles de madurez. Estos niveles aplican para procesos demejora en organizaciones a través de múltiples áreas de proceso. Existen 5 niveles demadurez y son los siguientes:

[24] Meta Genérica: están asociados a un nivel de capacidad, establece que una organización debe alcanzar en ese nivel de capacidad. El logro de cada una de estas metas en un área de proceso significa mejorar el control en la ejecución del área de proceso.

[25] Meta Específica: se aplican a una única área de proceso y localizan las particularidades que describen que se debe implementar para satisfacer el propósito del área de proceso.

- **Inicial o Nivel 1**.- Este es el nivel en donde están todas lasempresas que no tienen procesos. Los presupuestos se incrementan y no es posibleentregar el proyecto en fechas. No hay control sobre el estado del proyecto.
- **Gestionado o Nivel 2**.- Implica que el éxito de los resultadosobtenidos se pueden repetir. La principal diferencia entre este nivel y el anterior esque el proyecto es gestionado y controlado durante el desarrollo del mismo. Eldesarrollo es transparente y se puede saber el estado del proyecto en todo momento.
- **Definido o Nivel 3**.- Significa que la forma de desarrollarproyectos (gestión e ingeniería) está establecida, documentada y existen métricas(obtención de datos objetivos) para la obtención de objetivos concretos.Las empresas que adoptan CMMI en nuestro medio llegan hasta el nivel 3, ya que esun nivel que proporciona muchos beneficios y no ven la necesidad de ir más alláporque tienen cubiertas la mayoría de sus necesidades.
- **Gestionado Cuantitativamente o Nivel 4**.- Los proyectos usanobjetivos medibles para alcanzar las necesidades de los clientes y la organización.Se usan métricas para gestionar la organización.
- **Optimizado o Nivel 5**.- Los procesos de los proyectos y de laorganización están orientados a la mejora continua de procesos. Este nivel secaracteriza porque existen mejoras incrementales e innovadoras de los procesos queson identificados y evaluados mediante métricas para luego ser mejorados.

d) Áreas de Proceso: Las áreas de proceso identifican un conjunto de actividades relacionadas que cuando serealizan conjuntamente, logran alcanzar un conjunto de metas importantes.

Las Áreas deProceso contienen un conjunto de componentes que se detalla a continuación y quedan gráficamente expuestas en la figura 2.10.

- **Componentes Requeridos:**

  Objetivo genérico: Los objetivos genéricos asociados a un nivel de capacidadestablecen lo que una organización debe alcanzar en ese nivel de capacidad.

  Objetivo específico: Los objetivos específicos se aplican a una única área deproceso y localizan las particularidades que describen que se debe implementar parasatisfacer el propósito del área de proceso.

- **Componentes Esperados:**

  Práctica genérica: Una práctica genérica se aplica a cualquier área de procesoporque puede mejorar el funcionamiento y el control de cualquier proceso.

  Práctica específica: Una práctica específica es una actividad que se consideraimportante en la realización del objetivo específico al cual está asociado.

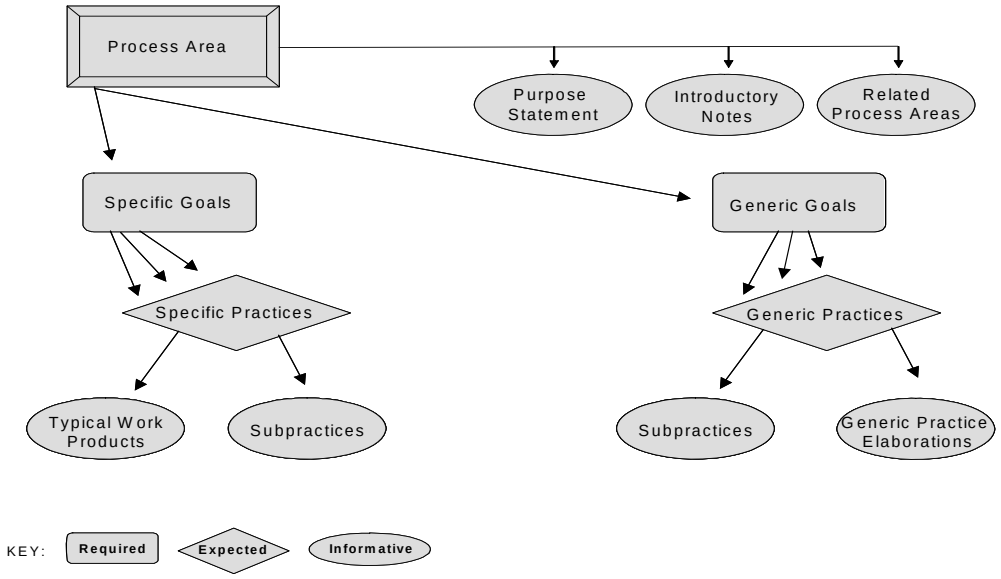


Figura. 2.10. Componentes del Modelo CMMI(Fuente: CMMI for Development v. 1.2)

A continuación las áreas de proceso organizadas según sus disciplinas:

Disciplinas	Área de proceso
Soporte	Análisis y resolución de problemas Gestión de la configuración Análisis y resolución de decisiones Medición y análisis Entorno organizativo para integración **Gestión calidad procesos y productos**
Gestión de proyectos	Gestión integral de proyecto Gestión integral de proveedores Gestión de equipos Monitorización y control de proyecto Planificación de proyecto Gestión cuantitativa de proyectos Gestión de riesgos Gestión y acuerdo con proveedores
Ingeniería	Integración de producto Desarrollo de requisitos Gestión de requisitos Solución técnica **Validación** **Verificación**
Gestión de procesos	Innovación y desarrollo Definición de procesos Procesos orientados a la organización Rendimiento de los procesos de la organización Formación

Tabla. 3. Las distintas disciplinas y sus áreas de procesos

2.5 PMI

En esta sección se explicará en qué consiste el PMI (Project Management Institute), su contribución a la dirección profesional de proyectos y en particular como se pueden utilizar estas buenas prácticas, reflejadas en la Guía del PMBOK[26], en pro de sacar un proyecto exitoso con un producto de calidad, enfocado en las áreas de procesos que tienen que ver con la calidad del proyecto y del producto.

2.5.1 Antecedentes

A lo largo de la historia, los proyectos han tenido una singular importancia, como prueba de esto, existen obras monumentales como las pirámides de México o Egipto, túneles y puentes capaces de conectar no solo ciudades sino países sobre o incluso por debajo del mar. Existen también eventos tan importantes a nivel mundial como lo podrían ser unas olimpiadas o incluso la exploración del espacio exterior, el desarrollo de un nuevo producto, una campaña publicitaria, la implementación de una nueva línea de producción, etc., que no dejan de ser proyectos a fin de cuentas.

Los métodos empleados para planear y ejecutar éstos y muchos otros proyectos en el paso del tiempo han sido muy diversos. Sin embargo, no fue sino hasta los últimos 50 años, desde la Segunda Guerra Mundial, que se han desarrollado los conceptos modernos y sistemáticos de la Administración de Proyectos, así como los métodos, sistemas y herramientas. En poco más de 10 años estos empezaron a difundirse rápidamente en todas las áreas o industrias, aprovechando al máximo los distintos medios de comunicación masivo como la Internet y la Red Mundial.

El desarrollo de la **Administración Profesional de Proyectos** como una profesión recibió el impulso inicial de dos industrias principalmente: La industria militar-aeroespacial y la industria de arquitectura-ingeniería-construcción. Hoy en día, los modernos conceptos de la Administración de Proyectos se emplean ampliamente en todo tipo de industria, negocio, empresa e institución gubernamental, en todo el mundo,

[26] La finalidad principal de la Guía del PMBOK® es identificar el subconjunto de Fundamentos de la Dirección de Proyectos generalmente reconocido como buenas prácticas.

incluso en el desarrollo de proyectos de software, que si bien es cierto no son productos tangibles o físicos como ya hemos visto, pero que al igual que la construcción de un edificio o un puente necesitan ser gestionadosy administrados eficientemente.

2.5.2 Definiciones

Para tener una idea más clara sobre lo que es o no es un proyecto es necesario que lo definamos primero dado que todos los días participamos en uno o más de ellos, algunos sencillos, otros complejos y otros más de carácter personal.

Un proyecto puede ser definido de varias formas, a continuación se presentan algunas definiciones de los principales organismos y autores en materia de administración de proyectos:

“Es la tarea asumida para crear un producto o un servicio nuevo”(PMI)

“Un proyecto es un esfuerzo emprendido para producir los resultados esperados por la parte que lo solicita”(Oberlender, 2000).

“Es un conjunto único de actividades interrelacionadas con tiempos de inicio y fin definidos, diseñado para alcanzar un objetivo común”(Nacional Competency Standards for Project Management, 1995).

“Es un proceso único, que consta de un conjunto de actividades coordinadas y controladas con fechas de inicio y fin, emprendidas para alcanzar un objetivo, conforme a requerimientos específicos, incluyendo restricciones de tiempo, costo y recursos” (ISO,1997).

Estas definiciones hacen referencia a dos características esenciales en cualquier proyecto para que se considere como tal y estas son que genere un **Producto o ServicioUNICO** y que sean **TEMPORALES**.

Cada proyecto significa generar algún producto o servicio que no se ha hecho con anterioridad y que por lo tanto es único. Se puede ver el concepto de único en la medida en que en cada proyecto tendrá un dueño diferente, un diseño distinto, una

localización diferente, diferentes contratistas también, por citar algunas diferencias que pueden ocurrir de un proyecto a otro. Puede que el objetivo sea el de producir el mismo tipo de producto, pero las condiciones del proyecto siempre son diferentes.

Se dice que son temporales porque tienen una fecha de inicio y otra de término. El término de un proyecto es cuando se han alcanzado los objetivos fijados en un inicio, o cuando está bien claro que los objetivos fijados no serán posibles de alcanzar, o también cuando ya no exista necesidad del proyecto y sea necesario terminarlo. La temporalidad de un proyecto significa que siempre tienen un inicio y un fin, no necesariamente que duran poco tiempo.

2.5.3 Ciclo de Vida del Proyecto

Al establecer que un proyecto es **temporal** se da por entendido que tiene un principio y un fin determinado. Esto supone una serie de procesos intermedios que nos llevan de un estado a otro. A estos procesos junto con la etapa de inicio y fin (o cierre) se le considera el ciclo de vida del proyecto. Este ciclo de vida está compuesto por los siguientes procesos:

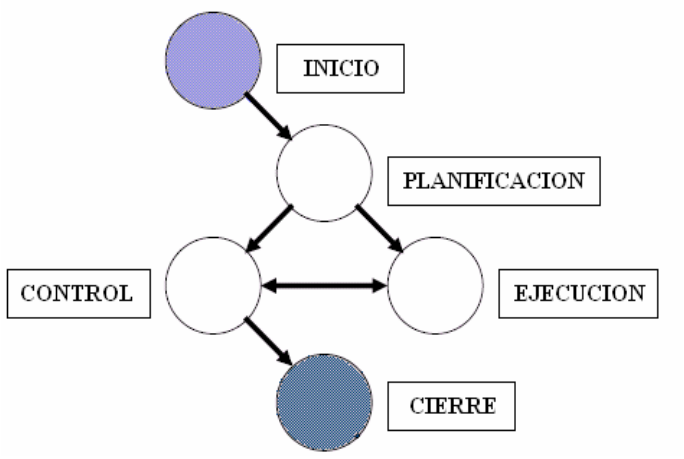
- **INICIO:**En la etapa de inicio se establece la visión del proyecto, el **¿qué?**; la misión por cumplir y sus objetivos, la justificación del mismo, las restricciones y supuestos.
- **PLANIFICACION:**En la etapa de planificación se desarrolla un plan que nos ayude a prever el ¿**cómo?** cumpliremos los objetivos, tomando en cuenta una serie de factores que afectan todo proyecto. Aquí se establecen las estrategias, con énfasis en la prevención en vez de la improvisación.
- **EJECUCION:** En la etapa de ejecución se implementa el plan, se hacen las contrataciones, se administran los contratos, se integra el equipo de trabajo, se distribuye la información y se ejecuta el proyecto conforme lo establecido.
- **CONTROL:** En la etapa de control se compara lo ejecutado o real contra lo que se planeó (control), de no identificar desviaciones, se continua con la ejecución. Si se encuentran desviaciones, en equipo se acuerda la acción

correctiva (planeación adicional), y luego se continúa con la ejecución, manteniendo informado al equipo.

- **CIERRE**: En la etapa de cierre se concluye y se cierran las relaciones contractuales profesionalmente para facilitar referencias posteriores al proyecto así como para el desarrollo de futuros proyectos. Por último, se elaboran los documentos con los resultados finales, archivos, cambios, directorios, evaluaciones y lecciones aprendidas, entre otros.

Al eliminar los procesos de inicio y cierre nos encontramos con una operación de rutina, en lugar de un proyecto. El ciclo repetido de mejora continua expuesto por **Deming**[27]y otros expertos en la materia: **Planificar** – **Hacer** – **Verificar** – **Actuar**, es similar a los procesos expuestos como se muestra a continuación en la figura2.11:

- **Planificar** = Planificación
- **Hacer** = Ejecución
- **Verificar** = Control
- **Actuar** = Planificación adicional, y ejecución



[27] Edward Deming, estadista, profesor y fundador de la Calidad Total.

Figura. 2.11. Ciclo de vida de un proyecto (Guía PMBOK)

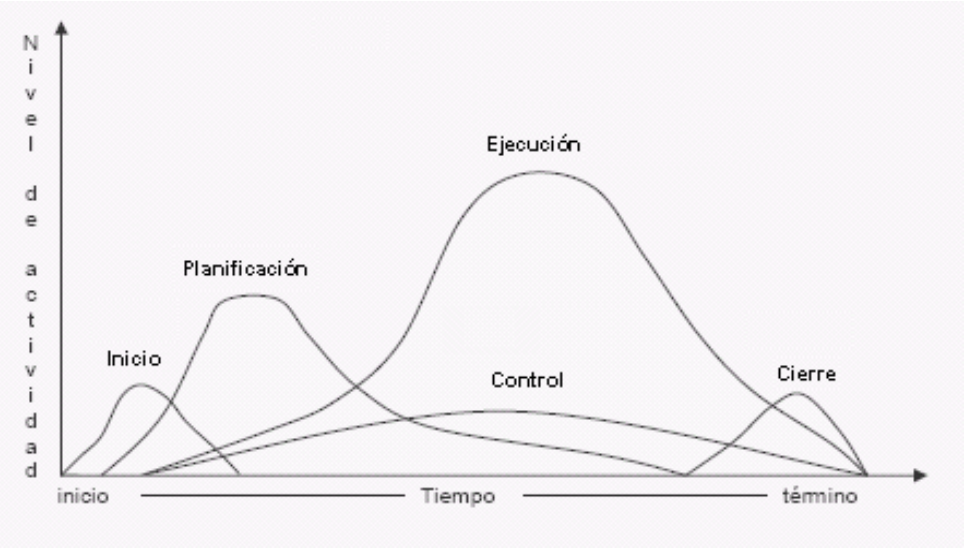


Figura. 2.12. Niveles de actividad de los 5 procesos dentro del ciclo de vida de un proyecto

Para entender mejor la relación de estos procesos a través del tiempo y el nivel de actividad que representan en el proyecto, se muestra en la figura 2.12,en el eje vertical el nivel de actividad y en eje horizontal el tiempo que transcurre desde el inicio hasta la finalización del proyecto.

En esta gráfica se puede apreciar que la curva de **Inicio** considera un lapso que se empalma con las curvas de **Planificación**, **Ejecución** y **Control**, ya que en muchos proyectos al inicio se establecen premisas que se deben de revisar en las etapas tempranas del proyecto, hasta confirmar su viabilidad.

También se puede observar, que en las etapas iníciales, la curva de Planificación muestra un nivel de actividad mayor, y disminuye hacia las etapas cercanas al Cierre, ilustrando que la planificación continua durante todo el proyecto, contrario al paradigma tradicional donde no hay tiempo para planificar, ya que se considera la planificación como un evento aislado y concluido antes de iniciar la ejecución. La razón de que la planificación sea continua, corresponde al ciclo **planificar – ejecutar – controlar –**

planificar (nuevamente), donde periódicamente se desarrolla una planificación adicional o estrategias correctivas a lo largo de la vida del proyecto.

La curva de **Ejecución** empieza muy cerca de la **Planificación** y tiende a incrementar progresivamente su nivel de actividad hasta llegar a clímax del proyecto. Ahí empieza a descender gradualmente hasta llegar al cierre. La gran mayoría de los proyectos se comportan de esta forma por ser **temporales**[28], es decir, que la ejecución se incorpora y desincorpora gradualmente a lo largo del proyecto.

La curva de **Control** inicia y termina junto con la **Ejecución**. Es decir, si no hay ejecución, no hay control. Esto es porque el control implica comparar la planificación con la ejecución, si no se cuenta con una planificación adecuada, el control no arroja datos significativos, por lo que si no hay planificación no hay control.

La curva de **Cierre** considera un tiempo de desarrollo, debido a los cierres contractuales y administrativos previos a la conclusión del proyecto.

2.5.4 Conocimientos, Habilidades y Herramientas

Para lograr una adecuada administración profesional de proyectos se deben aplicar los conocimientos, habilidades, técnicas y herramientas a las actividades de un proyecto en cada uno de sus procesos, con el fin de satisfacer, cumplir e incluso superar las necesidades y expectativas de los involucrados.

2.5.4.1 Conocimiento

Los CONOCIMIENTOS son el "**saber**" que está enfocado principalmente a dos tipos de conocimientos que son los siguientes:

- Conocimientos del Producto, Industria y Negocio.
- Conocimientos de la administración Profesional de Proyectos.

[28] Unas de las principales características de un proyecto, según PMI, es su temporalidad.

En el primer caso la importancia radica en conocer el medio en el que se mueve el tipo de proyecto a manejar, para así entender el funcionamiento o requerimientos específicos del tipo de proyecto. Si se trata de un proyecto para sacar un nuevo sistema de inventarios es necesario que el gerente de proyecto tenga conocimientos de sistemas; si es una línea de producción el gerente de proyecto deberá tener conocimientos acerca del funcionamiento de dicha línea de producción; si se trata de un proyecto de construcción el gerente de proyectos deberá tener conocimientos sobre los sistemas y procedimientos constructivos, etc.

En el segundo caso el gerente de proyectos deberá conocer los conceptos, filosofía, metodología, técnicas y herramientas para el manejo de proyectos en general.

2.5.4.2 Habilidades

Las **Habilidades** es el “saber hacer” y el “poder hacer”, todas estas pueden ser desarrolladas en la mayoría de los casos, pero es importante considerar cuales son las más relevantes para la administración de proyectos y así establecer un plan de acción para desarrollarlas en el equipo. Las habilidades clave para el gerente de proyecto se mencionan y describen a continuación:

- LIDERAZGO
- COMUNICACIÓN
- NEGOCIACIÓN
- SOLUCION DE PROBLEMAS
- HACER QUE LAS COSAS SUCEDAN

Se necesita LIDERAZGO en el gerente de proyecto, ya que se requiere de alguien que establezca una dirección, que pueda alinear al equipo y genere un ambiente que motive e inspire.

El gerente de proyecto tiene que tener la habilidad de establecer una buena COMUNICACIÓN con su equipo, ya sea escrita u oral, que sepa escuchar y hablar, que maneje una buena comunicación interna y externa, formal e informal, vertical y horizontal, con buenas técnicas de presentación y manejo de juntas para que, de esa manera, la información fluya de forma adecuada y así asegurar la comunicación efectiva entre la administración y otras organizaciones externas

El gerente de proyecto tiene que desarrollar la habilidad de NEGOCIAR ya que eso lo hará todo el tiempo; debe tener una filosofía de ganar-ganar, poder negociar los objetivos de tiempo, costo y alcance, negociar los términos y condiciones contractuales, negociar la asignación de recursos y todo lo que implique una negociación con otros participantes.

Todo gerente de proyecto debe ser capaz de dar una SOLUCION A LOS PROBLEMAS que se presenten en el transcurso de la vida del proyecto y para esto debe ser capaz primero de identificar y definir el problema y así solucionar las causas principales más que enfrascarse en los síntomas. Debe por lo tanto también ser lo suficientemente capacitado y facultado para la toma de decisiones que traigan como resultado la solución a los problemas que se presentan.

Por último quizás una de las habilidades más importantes es HACER QUE LAS COSAS SUCEDAN mediante la venta de ideas, el entendimiento de las estructuras formales e informales y el saber con quién dirigirse o a quien persuadir para que las cosas se den.

2.5.4.3 Herramientas

Por último tenemos las HERRAMIENTAS que nos pueden apoyar en la administración de cada uno de los procesos que involucra el proyecto y que se mencionan a continuación:

Para el **Inicio** del proyecto

- Mapas Mentales
- Charter Proyect

Para la **Planificación** del proyecto

- Plan del proyecto
- Declaración del alcance
- WBS
- Diagrama organizacional
- Matriz de roles y funciones
- Matriz de comunicación
- Calendario de eventos
- Estatus semanal
- Reporte mensual
- Programa del proyecto – ruta critica
- Estimados de costos
- Presupuesto base (baseline)
- Programa de erogaciones – flujo de efectivo
- Análisis de precedentes (benchmarking)
- Diagrama causa – efecto (con lista de verificación)
- Mapa de riesgos
- Matriz de administración de riesgos
- Matriz de abastecimientos
- Sistema de control de cambios

- Distribución de la información
- Lecciones aprendidas

Para la **Ejecución**del Proyecto

- Integración de equipos y comunicación
- Lista de aseguramiento de calidad
- Administración de concursos y cotizaciones
- Matriz de evaluación de alternativas
- Estado de cuenta de contrato
- Requisiciones de pago

Para el **Control** del Proyecto

- Control del programa
- Control presupuestal
- Valor ganado (earned value)
- Control de calidad
- Control de cambios
- Lecciones aprendidas
- Estatus semanal
- Reporte mensual

Para el **Cierre** del proyecto

- Cierre contractual
- Reporte final
- Cierre administrativo
- Lecciones al cierre

Cada una de estas herramientas se encuentra ligada a algún aspecto que la administración profesional de proyectos considera relevante para realizar de manera exitosa cualquier tipo de proyecto, conocidas como las 9 áreas del conocimiento se describen a continuación.

2.5.5 Las 9 Áreas del Conocimiento

En cada uno de los procesos que se llevan a cabo en la administración de los proyectos entran en juego 9 aspectos sumamente relevantes y que afectan todo proyecto por lo que estos deben estar no solo contemplados sino debidamente analizados y descritos para asegurar que el proyecto llegue a buen término. Estos aspectos o “áreas del conocimiento” como lo describen los especialistas en la administración profesional de proyectos son los siguientes:

- **INTEGRACIÒN**
- **ALCANCE**
- **TIEMPO**
- **COSTO**
- **CALIDAD**
- **RECURSOS HUMANOS**
- **COMUNICACIÒN**
- **RIESGO**
- **ABASTECIMIENTOS O ADQUISICIONES**

2.5.5.1 Alcance

El objetivo de la administración del **Alcance** es el asegurar que el proyecto incluya todo el trabajo requerido y solo el trabajo requerido para terminar el proyecto exitosamente. Esta área del conocimiento incluye aspectos como:

- **Iniciación**: Autorizar el proyecto o la fase.

- **Planificación del alcance**: Desarrollar una declaración escrita del alcance como la base para las decisiones futuras del proyecto.
- **Definición del alcance**: Subdividir los entregables principales del proyecto en componentes más pequeños o manejables.
- **Verificación del alcance**: Formalización de la aceptación del alcance del proyecto.
- **Control de cambios del alcance**: cambios que controlan el alcance del proyecto.

2.5.5.2 Tiempo

El objetivo de la administración del **Tiempo** es gestionar todos aquellos procesos requeridos para asegurar que se terminen las actividades puntualmente conforme se había establecido. Esta área comprende los siguientes aspectos.

- **Definición de la actividad**: Identificar las actividades específicas que se deben realizar para producir las distintas fases del proyecto.
- **Interrelación de las actividades**: Identificar y documentar las relaciones entre las actividades ya sean de dependencia o regidoras
- **Duraciones estimadas de las actividades**: Estimar el número de horas laborales que serán necesarias para terminar cada una de las actividades.
- **Desarrollo del programa**: Analizar las secuencias de las actividades, sus duraciones, y requisitos de recursos para así poder generar el programa del proyecto.
- **Control del programa**: Llevar el control del programa de actividades para identificar posibles desviaciones y evaluar el cumplimiento de las actividades programadas.

2.5.5.3 Costo

El objetivo de la administración del **Costo** es el asegurar que el proyecto sea concluido dentro del presupuesto aprobado. Para la adecuada estimación del presupuesto se cuenta con herramientas como la información histórica, la investigación de mercado, las cotizaciones y bases de datos que pueden orientar con respecto al orden de los gastos en los que se incurrirán. Dentro de los aspectos concernientes a los costos se pudieran mencionar los siguientes:

- **Planificación de los recursos:** Determinando qué recursos (gente, equipo, materiales) y qué cantidad de cada uno se deben utilizar para realizar las actividades del proyecto.
- **Costo estimado:** Desarrollar una aproximación (estimación) del costo de los recursos necesitó para terminar las actividades del proyecto.
- **Determinación del presupuesto:** es la asignación de la valoración de costos total de todas las actividades individuales de las que se compone el proyecto.
- **Control de costos**: controlar los cambios que pudieran incrementar el presupuesto de proyecto para reducir o incluso suprimir su impacto en la realización de las actividades
- **Costo estimado:** desarrollar una aproximación (estimación) del costo de los recursos necesitó para terminar las actividades del proyecto.
- **Determinación del presupuesto:** es la asignación de la valoración de costos total de todas las actividades individuales de las que se compone el proyecto.
- **Control de costos:** controlar los cambios que pudieran incrementar el presupuesto de proyecto para reducir o incluso suprimir su impacto en la realización de las actividades

2.5.5.4 Calidad

Entre los objetivos de la administración de la **Calidad** se encuentran el asegurar que el proyecto satisfaga las necesidades para el cual se inició, identificar los

estándares de calidad relevantes al proyecto así como determinar cómo satisfacer esos estándares. Los aspectos que cubre esta área son los siguientes:

- **Planificación de la calidad:** Identificar los estándares de calidad que son relevantes al proyecto y determinar cómo satisfacerlos.
- **Garantía de calidad:** Funcionamiento total de evaluación del proyecto sobre una base regular para proporcionar confianza que el proyecto satisfaga los estándares de calidad relevantes.
- **Control de calidad:** La supervisión de proyecto específico resulta para determinarse si se conforman con estándares e identificar relevantes de calidad maneras de eliminar causas del funcionamiento insatisfactorio.

2.5.5.5 Recursos Humanos

El objetivo de la administración de los **Recursos Humanos** es lograr el mejor desempeño de las personas participantes en el proyecto. Esta área comprende los aspectos siguientes:

- **Planeación de la organización:** Identificar, documentar, y asignar roles en el proyecto, delegar responsabilidades, y relaciones de trabajo.
- **Asignación del personal:** Conseguir los recursos humanos necesarios para trabajar en el proyecto y distribuirlos de la forma que se crea conveniente de acuerdo a los requerimientos de las actividades.
- **Desarrollo del equipo:** Desarrollar las habilidades tanto del individuo como del grupo en su conjunto para mejorar su desempeño individual y grupal con respecto el proyecto.

2.5.5.6 Comunicación

El objetivo de la administración de la Comunicación es lograr una comunicación efectiva entre los involucrados y asegurar la oportuna y apropiada generación, recolección, distribución, archivo y disposición final de la información del proyecto. Es

necesario planear tanto los contenidos, las frecuencias y las personas involucradas en las comunicaciones del proyecto. Esta área contempla los siguientes aspectos:

- **Planificación de las comunicaciones:** La determinación de la información y de las necesidades de comunicaciones de los involucrados: quién necesita qué información, cuando la necesitará, y cómo les será dada.
- **Distribución de la información:** Hacer que la información necesaria esté disponible para consulta de los involucrados de una manera oportuna y efectiva.
- **La evaluación de los reportes:** Desarrollar una manera clara y objetiva de evaluar los reportes generados y que sean entendibles para el resto de los involucrados y conozcan de esta manera el progreso del proyecto
- **Cierres administrativos:** Generar, recolectar, y distribuir la información para formalizar la terminación de la fase o del proyecto.

2.5.5.7 Riesgo

La administración del **Riesgo** tiene por objetivo reducir la repercusión negativa de los riesgos en el proyecto. Esto es mediante la identificación de áreas de oportunidad por lograr y las amenazas por controlar. Busca establecer un plan de manejo de riesgos con sus respectivos responsables. La esencia de la administración de riesgos está en prever continuamente posibles problemas para llevar a cabo acciones a tiempo en vez de improvisar y buscar soluciones tardías. En esta área se contemplan los siguientes aspectos:

- **Planificación del riesgo:** La planificación de los riesgos inherentes al proyecto.
- **Identificación del riesgo:** Determinándose qué riesgos pudieron afectar el proyecto y la documentación de sus características.
- **Análisis cualitativo del riesgo:** La ejecución de un análisis cualitativo de riesgos y las condiciones para dar la prioridad afecta los objetivos del proyecto.

- **Análisis cuantitativo del riesgo:** Midiendo la probabilidad y las consecuencias de riesgos y de estimar sus implicaciones para los objetivos del proyecto.
- **Planificación de la respuesta ante el riesgo:** Procedimientos y técnicas que se contemplan para realzar oportunidades y para reducir amenazas del riesgo a los objetives del proyecto.
- **Riesgo que se supervisa y se controlan:** Supervisando riesgos residuales, identificando los nuevos riesgos que ejecutan planes de la reducción del riesgo, y la evaluación de su eficacia a través del ciclo vital del proyecto.

2.5.5.8 Abastecimiento

La administración de Abastecimiento tiene por objetivo optimizar la adquisición de bienes y servicios externos a la organización a cargo del proyecto. Su administración abarca los siguientes aspectos:

- **Planificación del suministro:** La planificación de las compras, el que comprar y en qué momento comprarlo.
- **Las normas y especificaciones:** De todos los suministros que requiera el proyecto.
- **La identificación de los proveedores:** Identificar los posibles proveedores que pudieran suministrar lo requerido por el proyecto.
- **La emisión de solicitudes de compra:** Llevar el control de todas las compras.
- **La selección de proveedores:** Seleccionar los mejores proveedores para satisfacer las necesidades del proyecto.
- **La ejecución y cierre del contrato:** La terminación y el establecimiento del contrato.

2.5.5.9 Integración

Los objetivos de la administración de la INTEGRACIÓN es principalmente el asegurar que los diferentes elementos del proyecto sean propiamente coordinados. Esta integración comprende los siguientes aspectos:

- **El desarrollo del plan del proyecto**: Integra y coordina todos los planes del proyecto para generar un documento consistente y coherente.
- **El sistema de control de cambios:** Coordina los cambios que ocurren a lo largo de todos los procesos del proyecto.
- **Las lecciones aprendidas:**Lecciones que se presentaron y que fueron debidamente resueltas quedan como antecedente para futuros proyectos en los que se presenten circunstancias similares.

Dentro de los procesos de planificación, ejecución y control se pueden identificar la presencia de cada una de las áreas del conocimiento que se explicaron anteriormente y para cada una de ellas existen herramientas que nos permitirán planificar mejor, ejecutar mejor y controlar mejor nuestros proyectos. Estas relaciones se presentan en la Tabla 4.

AREA DEL CONOCIMIENTO	HERRAMIENTAS
ALCANCE	Declaración del alcance WBS
TIEMPO	Programa del proyecto (ruta crítica) Gantt del proyecto
COSTO	Estimados de costos Presupuesto base (baseline) Programa de erogaciones
CALIDAD	Diagrama de causa-efecto (con lista de verificación)
RECURSOS HUMANOS	Diagrama organizacional del proyecto Matriz de roles y funciones
COMUNICACIÓN	Matriz de comunicación Calendario de eventos Estatus semanal Reporte mensual
RIESGO	Mapa de riesgos Matriz de la administración de riesgos
ABASTECIMIENTOS	Matriz de abastecimientos
INTEGRACIÒN	Sistema de control de cambios Lecciones aprendidas

Tabla. 4. Relación entre las aéreas del conocimiento y las herramientas usadas

2.5.6 InteresadoClaves en el Proyecto o Stakeholders

Para poder cumplir con las expectativas de éxito de un proyecto, se depende en gran medida de la integración de muchas organizaciones y personas hacia un objetivo común: **el objetivo del proyecto**. Para poder cumplir las expectativas de los involucrados claves, primero se debe identificar, y para ello se deben definir de manera clara las organizaciones y personas que serán afectadas o beneficiadas por el desarrollo del proyecto. Luego se identificar 3 tipos de participantes:

- Equipo Directivo
- Equipo Ejecutor
- Involucrados Circunstanciales

El equipo directivo está integrado por el CLIENTE, que puede ser el contratante, propietario o desarrollador del proyecto, quien en un momento dado autoriza, define el alcance y establece los lineamientos y criterios de aceptación. También se encuentra dentro de este equipo el PATROCINADOR, que es la persona a cargo de la dirección del proyecto en la empresa y quien asegura la toma de decisiones a tiempo, apoya la asignación de recursos, supera conflictos y barreras organizacionales para una mejor realización del proyecto, apoya y asigna al gerente de proyecto.

El equipo ejecutor está integrado por el GERENTE DEL PROYECTO, como encargado del proyecto,el cual dirige al equipo del proyecto para alcanzar los objetivos, asegura la comunicación efectiva entre la administración y otras organizaciones, asegura que los problemas del proyecto sean identificados y resueltos a tiempo y adecuadamente. También forma parte de este bloque el resto de los MIEMBROS DEL EQUIPO, como el gerente de áreas, los organizadores, el staff y los proveedores quienes elaboran el plan del proyecto, ejecutan y controlan siguiendo el plan, colaboran en la integración de los equipos para lograr los objetivos del proyecto.

Por último se encuentran los INVOLUCRADOS CIRCUNSTANCIALES que aunque no se encuentren ligados directamente con el proyecto si resultan afectados o impactados de alguna manera como en el caso de los medios, los vecinos, etc.

2.5.7 Influencias Organizacionales

Es esencial evaluar la influencia de nuestra organización en el manejo de nuestros proyectos ya que las estructuras de las organizaciones impactan definitivamente el proceso en que los proyectos son realizados. En algunos casos las estructuras apoyan la cultura de la administración de proyectos y en otros casos sin embargo la obstaculizan por lo que se incluyen algunas recomendaciones prácticas para ayudar al equipo realizador a capitalizar las ventajas y a controlar o minimizar las desventajas.

Primero se pueden diferenciar dos estructuras organizacionales básicas: estructuras funcionales y estructuras en base a proyectos. Existe una tercera que se conoce como matricial y que considera la combinación de ambas estructuras y es la más común para empresas que, además de su operación diaria, continuamente trabajan con proyectos.

2.5.7.1 Estructuras Funcionales

Las Estructuras Funcionales están organizadas de acuerdo con las funciones de los diferentes departamentos: dirección, ventas, ingeniería, mantenimiento, operación, proyectos, etc. Este tipo de estructuras presentan las siguientes ventajas y retos a la administración profesional de proyectos (Fig. 2.13):

VENTAJAS

- Permite la especialización
- Aprovechamiento de la curva de aprendizaje técnico
- Los canales de comunicación verticales bien establecidos
- Provee continuidad en las disciplinas funcionales (políticas, procedimientos, etc.)

DESVENTAJAS

- Conflictos entre gerentes de proyectos y gerentes de departamentos
- Gerentes de proyecto sin atribuciones.

- Gerentes y miembros del equipo con exceso de carga de trabajo
- El equipo no hace suyo el proyecto
- Siempre se da prioridad a quien paga

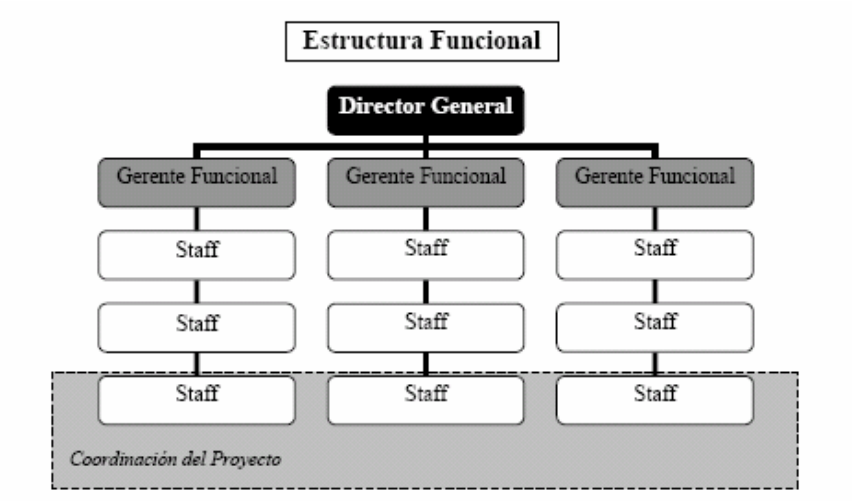


Figura. 2.13. Estructura Funcional

2.5.7.2 Estructuras en Base a Proyectos

Las Estructuras en Base a Proyectos están organizadas de tal manera que cada célula gira en torno a un proyecto: dirección, gerente de gerentes, gerente nivel 1, gerente nivel 2, etc. Este tipo de estructuras presentan las siguientes ventajas y desventajas a la administración profesional de proyectos; tal como se muestra en la figura 2.14.

VENTAJAS

- Equipo asignado 100% al proyecto
- Mayor compromiso en el proyecto
- Rapidez en el tiempo de respuesta
- Rendición de cuentas
- Los participantes del proyecto trabajan directamente para el gerente del proyecto

DESVENTAJAS

- Colaboradores sin sentido de pertenencia a la empresa
- Colaboradores con inseguridad laboral
- Es costoso mantener un equipo especializado en proyectos
- Oportunidades de crecimiento limitadas
- Tendencia a mantener al personal más tiempo del necesario

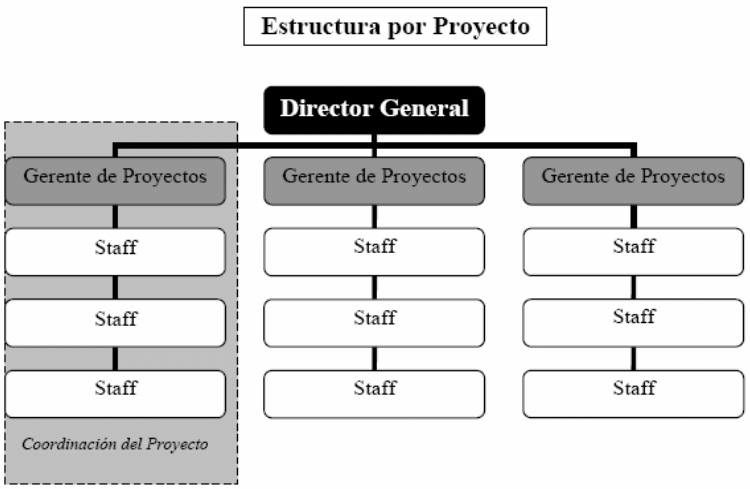


Figura. 2.14. Estructura por Proyectos

2.5.7.3 Estructura Matricial

En el caso de la Estructura Matricial se tiene un componente funcional, que puede ser mayor al 80% de su personal, y el resto dedicado a un departamento de proyectos, con un gerente de gerentes o director de proyectos, apoyado por un equipo de gerentes de proyectos. En muchos casos, este tipo de estructura tiende a homologarse al esquema funcional, presentando las ventajas y retos de ésta, como se indica en la figura 2.15

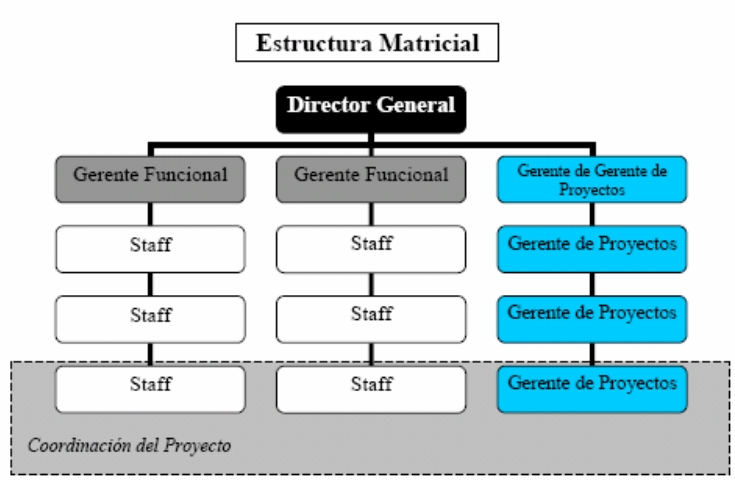


Figura. 2.15. Estructura Matricial

2.5.8 Factores del Éxito en los Proyectos

El entender los proyectos es un elemento crítico para la realización de las estrategias de negocios de cualquier organización, ya que estos son los medios por los cuales son implementadas aquellas, de ahí la importancia de conocer bien sus partes y sus características, y por lo tanto de la necesidad de una buena administración y búsqueda continua por generar proyectos exitosos.

De acuerdo con un artículo sobre la evolución de la Administración de Proyectos por Harold Kerzner[29], se identifican tres periodos en función a la definición del éxito en proyectos, los cuales se mencionan a continuación:

Periodo Tradicional (1960-1985). En este periodo se mide el éxito principalmente en términos técnicos. En este período y aun en casos actuales, al contratar el desarrollo de un proyecto, consideramos que generalmente no se cumplirá con el tiempo de entrega y el presupuesto y para considerar exitoso el proyecto será suficiente que funcione como esperamos. Con base en las expectativas expuestas, las habilidades requeridas por el encargado del proyecto serías más técnicas, enfocadas a la experiencia y conocimientos específicos sobre el tipo de proyecto en particular. El estilo de liderazgo es autoritario, efectuado por la persona con más experiencia

29 Harold Kerzner. Project Management: A Systems Approach to Planning, Scheduling, and Controlling, 8th Ed. edición, Wiley.

Periodo de Renacimiento (1985-1993). El éxito se mide en función del apego al tiempo, costo y calidad técnica. En dicho lapso y aún en la actualidad no es suficiente que el proyecto cumpla con los requerimientos técnicos si no se termina en el tiempo establecido y dentro de presupuesto. De ahí que las habilidades técnicas no son suficientes por parte del responsable, sino que se requiere del manejo efectivo de equipos humanos para lograr nuevos requerimientos. El liderazgo debe de ser más participativo que en el periodo tradicional.

Periodo Moderno (1993 – a la fecha) Se mide el éxito en función del apego al tiempo, costo, calidad técnica y aceptación del cliente. En este período se estáconsciente que aunque se entregue el proyecto dentro del presupuesto, a tiempo y con la calidad técnica estipulada, si el cliente no queda satisfecho, no se puede considerar que el proyecto fue exitoso. Con lo anterior, las habilidades del encargado no se limitan únicamente a la experiencia técnica y manejo de equipos humanos, sino que se debe de tener liderazgo efectivo, que permita una extensa comunicación, logrando que las cosas sucedan, que exista poder de negociación, que se permita una serie de soluciones al mismo problema, etc. También en esta etapa el cliente debe de estar involucrado en las decisiones que se toman según las fases del proyecto que se van desarrollando.

En todas las definiciones de éxito de un proyecto, en menor o mayor grado, pero siempre presentes, se encuentran tres de las nueve áreas del conocimiento de las que se habló anteriormente que son:

- **TIEMPO**
- **COSTO**
- **CALIDAD**

A estos 3 aéreas se les debe de sumar una cuarta que es el ALCANCE, la cual define las actividades que se realizarán y lo que quede fuera de proyecto sencillamente no se llevará a cabo.

Con estos 4 elementos se puede formar un triángulode relación en el cual el alcance se encuentra en el vértice superior y en sus vértices inferiores se encuentran el costo y el tiempo, dado que a un alcance mayor, un costo mayor y posiblemente un tiempo de entrega mayor. La calidad se integra a los lados de este triángulo, ya que la calidad se afectara al cambiar el alcance, el tiempo de entrega o el costo. Si el tiempo o el costo se reducen, la calidad puede quedar afectada Cada vértice de este triángulo tratara de estirar en su dirección desequilibrando el proyecto como un conjunto; por ejemplo, si se desea reducir el tiempo se tendrá que reducir el alcance o incrementar el costo, como se indica en la figura 2.16.

Una de las funciones más importantes del gerente del proyecto es lograr el equilibrio entre el **Alcance-Tiempo-Costo,** buscando no comprometer la **CALIDAD** en el proceso. Se debeestablecer desde un principio las fronteras de los 3 vértices, para monitorearlas muy de cerca en el desarrollo de los trabajos previos al diseño, durante este y a lo largo de la implementación, hasta llegar finalmente al cierre del proyecto.

Adicionalmente este triángulo se ve reforzado por el resto de las áreas del conocimiento como lo son los **recursos humanos, la comunicación, el riesgo, los abastecimientos y la integración**.

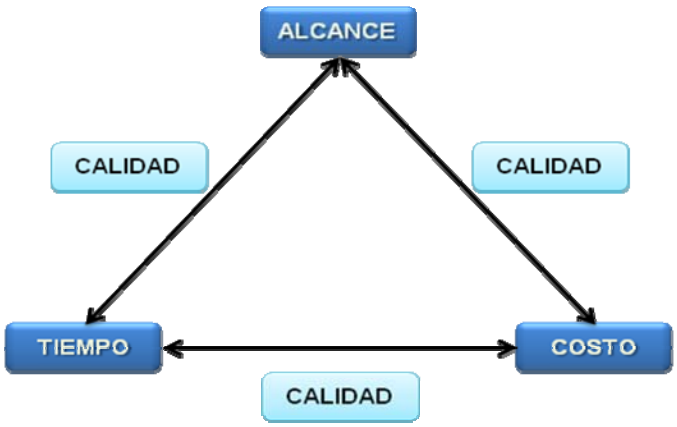


Figura. 2.16. Triangulo de Relación entre Alcance, Tiempo, Costo y Calidad.

Mediante un buen WBS (Work Breakdown Structure)[30] se puede manejar el **Alcance del Proyecto**. Esto resultará muy útil al momento que se tengan que definir las partidas del presupuesto y los tiempos estimados ya que es de la misma estructura del WBS que se generan esas herramientas.

Con el WBS desarrollado se pueden desarrollar las partidas que formarán parte del Presupuesto Base, al mismo tiempo que sirven para establecer la secuencia lógica de las actividades, sus relaciones y sus tiempos requeridos para así formar el Cronograma. Por último,teniendo claro que es lo que se va a hacer; se pueden establecer las Especificaciones que se requieren para las actividades considerándolas como requerimientos técnicos que se deben de cubrir para que pueda ser considerado exitoso el proyecto

El Presupuesto Base hace referencia al Costo, que es el dinero que se tiene contemplado para la realización de todas las actividades que conforman el cronograma, se basa en los costos de los recursos, personas, equipamiento y materiales necesarios para terminar el proyecto. Esto se encuentra reflejado en el presupuesto del proyecto

El Cronograma hace referencia al TIEMPO, que establece la duración que tendrá el proyecto. La fecha de inicio y de fin, de cada actividad que compone el proyecto, así como la relación que existe entre cada una de ellas. Esto queda establecido en el cronograma del proyecto.

La **Lista de Verificación** es la herramienta que nos sirve para evaluar si las especificaciones que se estuvieron planteando cumplen con los requerimientos técnicos de CALIDAD establecidos

Estas variables son factores que determinarán el éxito o fracaso de un proyecto y por lo mismo deben de ser consideradas como de alta prioridad. Entre ellas existe una relación muy estrecha de forma tal que si se llegara a alterar alguna de ellas repercutiría

30 La Estructura de desglose de tareas o Work Breakdown Structure, corresponde a una división jerárquica y de multinivel del trabajo a realizar, que cubrirá los requerimientos del proyecto. (PMBOK, 2004)

invariablemente en las otras dos. Aunque todas estas variables son importantes, normalmente uno de ellos tendrá más influencia en el proyecto

La relación entre estos elementos difiere de un proyecto a otro, y determina la clase de problemas que el administrador encontrará y las soluciones que puede implementar. Si se sabe dónde encontraremos delimitaciones y dónde podremos ser flexibles, nos será más fácil planear y administrar el proyecto.

Si estos factores son tan importantes para el éxito del proyecto, resulta lógico el interés por parte de los administradores de poder planificar, asegurar y controlar, en la medida que sea posible, tanto el tiempo, el costo y la calidad de todas las actividades que conforman el proyecto.

En el caso del tiempo y costo, las herramientas permiten planear con un buen grado de certidumbre el tiempo y costo parcial de cada una de las actividades que conforman el proyecto y por ende el del proyecto en general. Esto da la ventaja de que en caso de que se detecte cualquier anomalía o desviación durante la realización de alguna actividad, todavía se puede ajustar, si así lo requiriera (optimizando otras partidas del presupuesto, inyectando más recursos a actividades críticas, etc.) para que el proyecto quede dentro del tiempo y costo establecido.

Es muy difícil asegurar los costos y tiempos en un 100%, ya que estos podrían variar al ser afectados por factores externos. Pero aún así, se cuenta con la suficiente certeza que el proyecto se comporte en la mayoría de los casos como se planteó en el presupuesto y Cronograma Inicial.

En el caso de la calidad es menos preciso, ya que su control se basa solamente en el cumplimiento o no de especificaciones planteadas desde el inicio del proyecto, a través de una lista de verificación y estas están dirigidas a insumos y productos y no al proceso que supone el llevar de un estado a otro. De esta manera puede que la materia prima sea de primera calidad y aun así el resultado final no sea el deseado.

Otro inconveniente, en el caso de las herramientas que llevan el control de la calidad, es que en la mayoría de los casos, los resultados que arrojan son cualitativos y

no cuantitativos como en el caso del costo y tiempo en los cuales se puede saber a ciencia cierta qué tan desviado está el desempeño real del planeado en pesos y días.

El aseguramiento de la calidad mediante modelos o herramientas en términos cuantitativos y no solos de insumos y productos sino también de los procesos que los componen no es nuevo. Lamentablemente no existe ningún modelo en la actualidad que permita hacer lo mismo en la construcción y su aplicación ha quedado rezagada a otras industrias principalmente la manufacturera. El poder generar y aplicar un modelo de aseguramiento de calidad en la construcción permitiría entonces poder asegurar la calidad de los procesos y de esta manera estar seguros que los insumos que hayan sido aprobados terminen en productos que cumplan las especificaciones establecidas y evitar de esta manera duplicidad de trabajo e incurrir en tiempos y costos extras. También permitirá este modelo generar un “índice” de calidad en el cual de forma cuantitativa se pueda observar el desempeño real vs el planeado e incluso evaluar construcciones ya realizadas bajo los mismos parámetros de evaluación.

III. EL MODELO DE CALIDAD PROPUESTO

En el presente capítulo se esboza el Modelo General de Calidad de Software Propuesto, es decir, el diseño de acciones ymétodos que constituyen un enfoque práctico para la implementación del modelo de calidad enuna organización o proyecto de desarrollo de software. También se presenta el modelo de mejora continua del mismo.

3.1 Modelo de Calidad de Software Propuesto

Una metodología de desarrollo de software (ya sea para software convencional u otro tipo de desarrollo) establece una forma consistente de ejecutar lasactividades relacionadas aldesarrollo del software dentro de una organización. Toda metodología que apunte a ello debe describir o contener lo siguiente:

- ✓ **Elementos**: Cubren un conjunto de actividades bien definidas, relacionadas y agrupadas.
- ✓ **Arquitectura**: Establece el orden, las interfaces, las interdependencias, las integraciones y otras relacionesentre los elementos.
- ✓ **Adicionalmente**, la metodología podría describir estándares, métodos y herramientas.

Como lo hemos descrito en los capítulos anteriores, las metodologías de desarrollo de software son las que proveen continuidad en el proceso de desarrollo deuna organización y es una referencia para métricas y mejoras de dicho proceso.

Sobre la base del concepto de metodología y de los conceptos básicos ya definidos en los capítulos anteriores, se propone un enfoque o modelo de la calidad desoftware que tiene las siguientes características:

- ✓ El modelo de calidad de software sugerido es una interfaz, es decir, se puede implementar independientemente de la metodología o ciclo de vida seleccionado,

además su estructura modular permite la posibilidad de implementarse completa o parcialmente.

- ✓ El presente modelo de calidad está basado en las mejores prácticas de los modelos de mejora continua CMMI y PMI, dándole enfoque en las áreas de procesos de Validación, Verificación, Control y Aseguramiento de la Calidad del Producto
- ✓ El modelo que aquí se presenta no es una metodología en sí misma, sino que debeacoplarse a una metodología o estándares de desarrollo seleccionados por las organizaciones o proyectos para poder implementarse.
- ✓ Esta interfaz está compuesta por módulos independientes, donde cada uno de ellos seasocia a ciertos procesos de la metodología.
- ✓ Debido a las características dinámicas de los procesos de mejora continua, con los que se basa el modelo propuesto, se han incorporado procesos de mejora continua en cada uno de los módulos que lo contienen.

La interfaz cuenta también con un módulo de ejecución recurrente, en donde, para todas ycada uno de las entradas a cada uno de los otros módulos, se verifica la conformidad de lasmismas con la metodología, estándares y normas aplicables que estén vigentes en laorganización o proyecto.

El modelo la calidad del software sugerido no es dependiente de lametodología de desarrollo utilizada en una organización o proyecto, sino que se adapta a ella. Sobre labase de los procesos de la metodología de desarrollo, sus precedencias, sus ciclos de vida o ejecución, y por supuesto de las características del proyecto en curso, se seleccionanaquellas porciones de la interfaz que resulten necesarias para obtener los niveles de calidaddeseados y el orden de ejecución de las mismas.

La siguiente figura 3.1 muestra un ejemplo de la adaptabilidad del modelo de calidad de software propuesto, sobre cualquier modelo de desarrollo, estándar o metodología asumido por alguna organización o proyecto en particular.

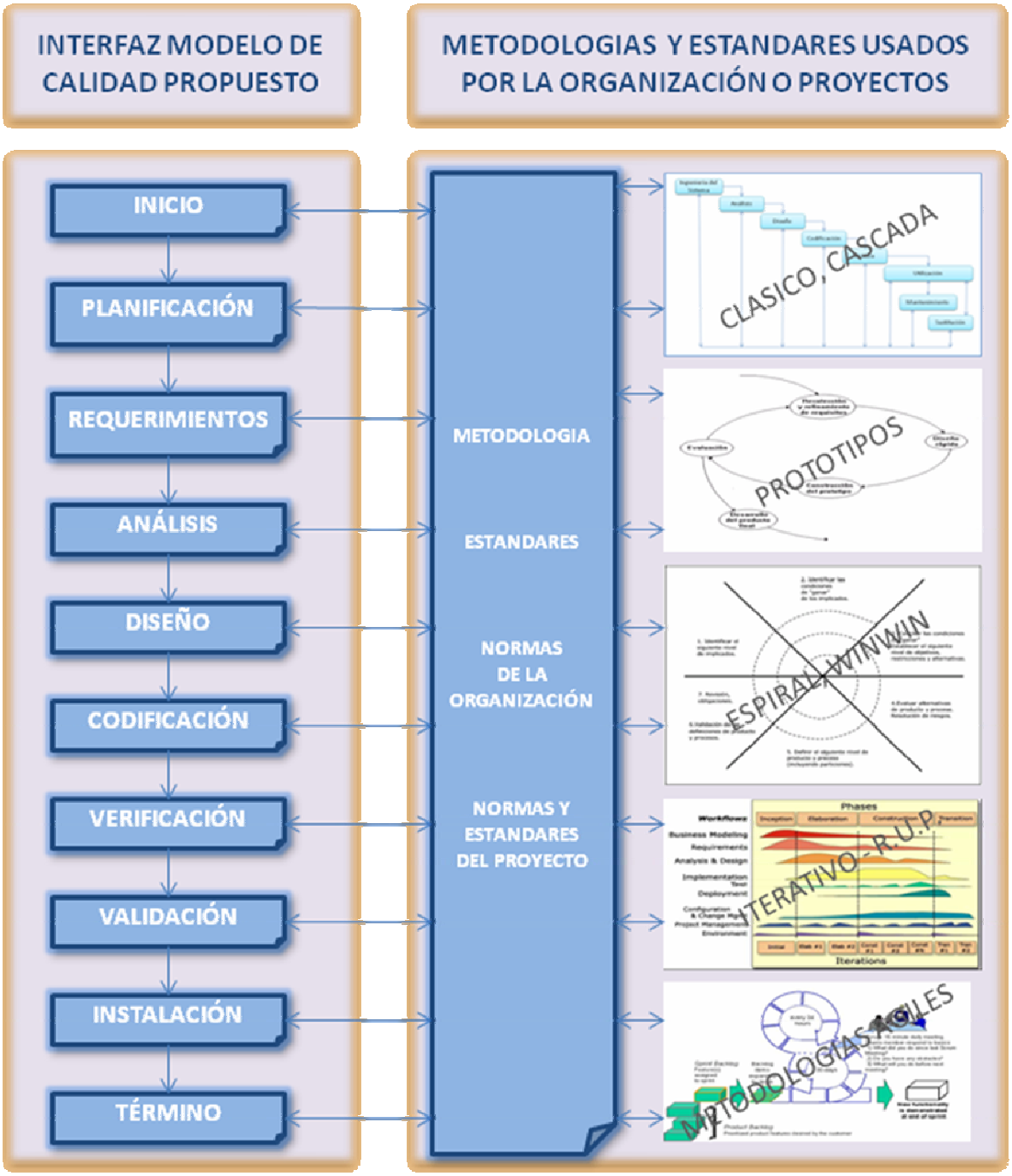


Figura. 3.1. Figura descriptiva de la aplicación del modelo de calidad propuesto sobre cualquier FrameWork de desarrollo.

3.2 Aplicación del Modelo de Calidad

El modelo aquí presentado, es un modelo genérico cuya flexibilidad y dinamismo permite su adaptacióna cada proyecto en particular. De esta forma, se podrán seleccionar los segmentos delmismo que cubran las necesidades de cada proyecto, evitando la realización de actividadesinnecesarias.

El modelo general de calidad de software, se documenta en un plan general dela calidad. La flexibilidad, que permite la adaptación del modelo genéricoo plan general de la calidad del software, a todo tipo de proyectos, seformaliza a través de los planes específicos de control y aseguramiento de la calidad del softwarepara cada proyecto.

El plan específico de aseguramiento y control de la calidad, para un proyecto en particular, seráentonces, la adaptación del plan general a las características particulares de ese proyecto.

La figura 3.2 muestra la relación entre el plan general del modelo de la calidaddel software y cada uno de los planes específicos:

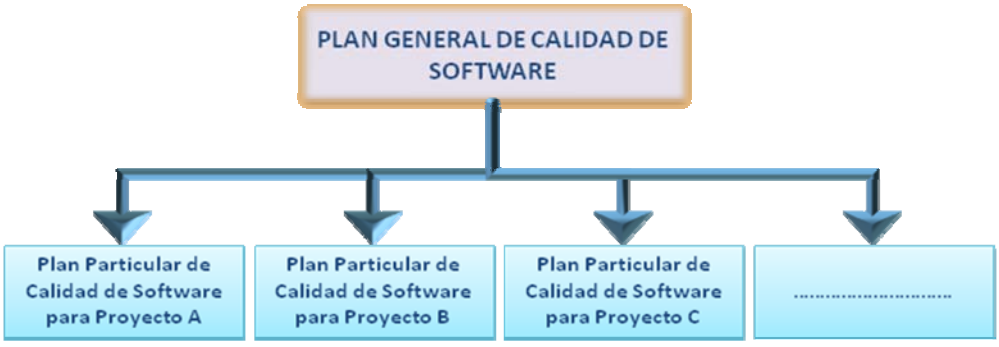


Figura. 3.2. Relación entre plan general de calidad de software y los planes específicos para cada proyecto.

3.3 Actualización del Plan

El modelo o plan general de la calidad del software aquí presentado nomantiene un estado estacionario o estático, sino que por el contrario es dinámico y sujeto a

permanentesmejoras y actualizaciones de acuerdo a la experiencia y a las necesidades que surjan comoconsecuencia de la aplicación del mismo.

La figura 3.3.1muestra el modelo de actualización y mejora continua del plan generalde calidad del software y de los planes específicos de calidad del software de cada proyecto, así como la adaptación del plan general, y laejecución y verificación de los planes específicos:

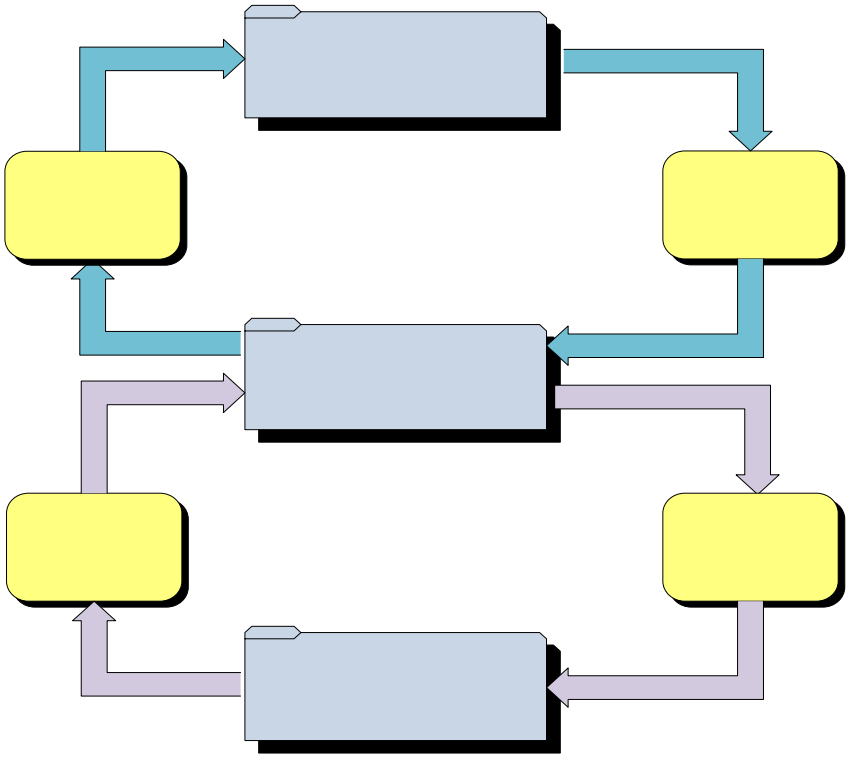


Figura. 3.3.1 Modelo de actualización y mejora continúa del plan general de calidad de software

3.3.1 Detalle del Modelo de Mejora Continua del Plan

A continuación se detalla el modelo de mejora continua del plan general de calidad, como así también de los planes específicos correspondientes a cada proyecto en particular.

Adicionalmente, y a modo de contexto, se detallan brevemente los pasos correspondientes a la adaptación del plangeneral, y la ejecución y verificación de los planes específicos:

- ***Paso 1:***

  - ✓ Entrada : Plan general de calidad de software.
  - ✓ Acción : Adaptación del plan general de calidad de software acada proyecto particular.
  - ✓ Salida : Plan específico de calidad de software para un proyectoParticular

- ***Paso 2:***

  - ✓ Entrada : Plan específico de calidad de software, para unproyecto particular
  - ✓ Acción : Ejecución del plan específico de calidad de software.
  - ✓ Salida : Productos resultantes de la ejecución del plan de calidad de software.

- ***Paso 3:***

  - ✓ Entrada :Productos resultantes de la ejecución del plan específico de la calidad de software.
  - ✓ Acción :Análisis de la calidad de los productos resultantes de la ejecución del planespecífico. Generación de recomendaciones y acciones a realizar, con sucorrespondiente justificación, para mejorar el plan específico de calidad de software.
  - ✓ Salida :Plan específico de calidad de software, para un proyecto particular, actualizado.

- ***Paso 4:***

  - ✓ Entrada :Actualizaciones realizadas al plan específico de calidad de software, para un proyecto particular.
  - ✓ Acción : Análisis de las actualizaciones realizadas al plan específico dela calidad del software, con sus correspondientes justificaciones.Actualización del plan general de calidad del software.
  - ✓ Salida : Plan general de calidad del software actualizado.

3.4 Módulos del Modelo de Calidad

En la siguiente sección se describirá cada uno de los módulos del modelo general de calidad de software propuesto, indicando para cada uno de ellos los siguientes Ítems:

- ***Objetivo:*** Descripción del propósito del módulo, para usar como una regla contra la quemedir el progreso del mismo.
- ***Entrada***: Documentos y/o información necesaria para completar el módulo. El nombre delos documentos es genérico, sin embargo este nombre debe ser adaptado al nombre delproducto resultante, del proceso lógico correspondiente, en la metodología utilizada.
- ***Proceso***: Descripción de las tareas y acciones que debe ejecutar el o los integrantes del equipode aseguramiento o control de calidad para dar por cumplido al módulo.
- ***Salida***: Productos que deben ser generados al final del módulo.
- ***Lista de verificación***: Checklist que utilizaranlos integrantes del proyecto responsables de asegurar y control de la calidad del producto para verificar que ha cumplido el módulo correctamente. Las lista de verificación se definen entablas, los campos de esas tablas se explican en la tabla 3.1:

COLUMNA	SIGNIFICADO
Numero	Identifica secuencialmente los ítems de control de calidad, el resultado positivo indicaría que ese paso ha sido realizado correctamente.
Ítem	Ítem específico de control de calidad que es usado para medir la efectividad de ejecución de este pasó.
Respuesta	El encargado de la calidad deberá indicar en esta columna si ha realizado el ítem referenciado. La respuesta puede ser: SI, NO o N/A si la misma no es aplicable al proyecto en cuestión.
Comentarios	Esta columna es utilizada para clarificar la respuesta de SI, NO o N/A para los ítems indicados. Generalmente la columna de comentarios sólo se completa en lasrespuestas por NO; las respuestas por NO deberán ser investigadas y se deberá tomar una determinación respecto a si este ítem deberá ser completado antes de considerar este paso

	completo.

Tabla. 3.1. Explicación de los campos que forman parte de las tablas que definen las listas de verificación.

- ***Métricas****: Mediciones a realizar durante la ejecución del módulo. Estas métricas searchivarán convenientemente de forma tal que permitan evaluar el avance y mejora de lacalidad del proyecto en curso. Asimismo se utilizarán como base para estimaciones ycomparaciones para proyectos futuros.*
- ***Involucrados:*** *Perfiles del equipo de desarrollo y del equipo de SQA involucrados en elmódulo. Los nombres de los perfiles del equipo de desarrollo son genéricos, sin embargoestos deben ser adaptados al nombre del perfil equivalente definido en la metodologíautilizada.*

3.5 Detalle de los Módulos del Modelo de Calidad

En el presente capítulo se describen los Módulos del Modelo, se presentan para cada unode ellos las acciones a desempeñar, su objetivo, sus entradas, salidas, lista de verificación,métricas y participantes involucrados.

3.5.1 INICIO

3.5.1.1 Objetivo

- Determinar el Project Chartero Acta de Constitución del Proyecto
- Establecer documentación inicial del proyecto
- Determinar inicialmente los Stakeholdermás importantes del proyecto
- Planificar el Kick-Off del Proyecto.

3.5.1.2 Entrada

- Requerimientos y declaración de intención inicial del proyecto
- Mapas mentales o antecedentes iníciales
- Casos de Negocio Inicial del Proyecto o Iniciativa

3.5.1.3 Proceso

En la figura 3.5.1 se detalla gráficamente, el proceso o módulo de Inicio del proyecto.

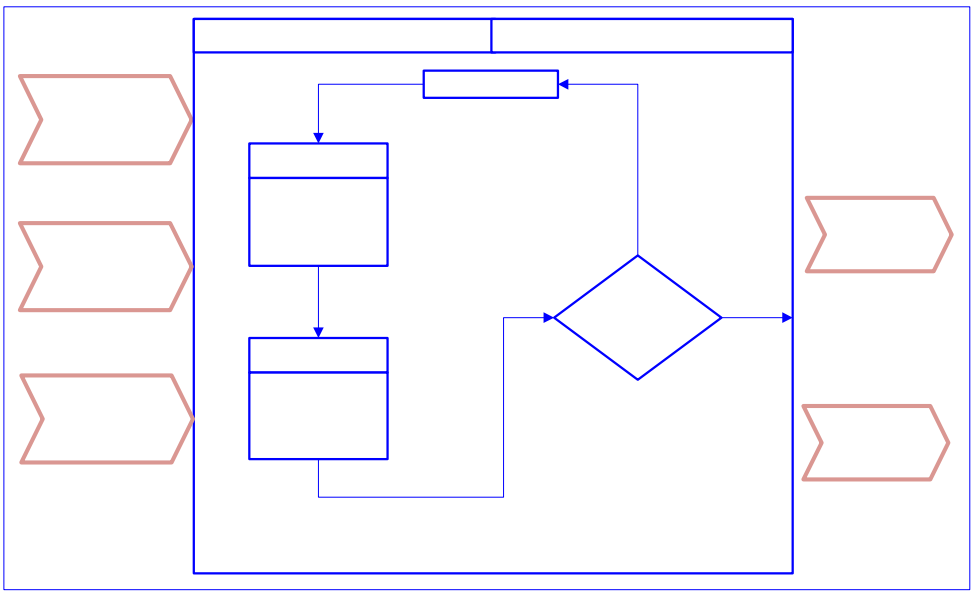


Figura. 3.5.1. Proceso del módulo de Inicio

3.5.1.4 Verificar Viabilidad y Recursos Iníciales del Proyecto

Antes de comenzar con las actividades propias de un proyecto informático, se deben verificar los antecedentes iníciales con que se cuenten, esto para determinar si desde un principio el proyecto es viable, tanto técnicamente, como financiera mente, algunos parámetros a tomar en cuenta en esta verificación son:

- Requerimientos y declaración de intención inicial del proyecto.
- Arquitectura tecnológica de la organización.
- Estándares o metodologías a cumplir
- Prioridad del proyecto.
- Sponsor del proyecto.
- Visión táctica y estratégica de la compañía.
- Recursos financieros o centros de costos disponibles.

El análisis inicial de estos antecedentes nos permitirán verificar y validar que el proyecto a iniciarse cuenta con las autorizaciones, recursos y es parte importante de la estrategia de la organización, ya que estadísticamente hablando muchos proyectos son comenzados sin justificación técnica o financiera, sinautorización ni sponsor,ni estar alineado con la estrategia de la organización. Estos proyectos se caen tempranamente, gastando valiosos recursos que muy bien se pudieron haber utilizar para otros proyectos de mayor peso e importancia.

3.5.1.5 Verificar Antecedentes Iníciales del proyecto

Una vez verificada la viabilidad del proyecto se tomaran todos los antecedentes posibles, tales como:

- Procesos de Negocios del área involucrada en el proyecto
- Procedimientos
- Mapas mentales
- Requerimientos de alto nivel o iniciales

- Casos de negocio ojustificación del proyecto.
- Análisis inicial de costo / beneficio

Esto permitirá construir el Acta de Constitución del Proyecto, este documento es el que determina la formal autorizaciónel proyecto ante la alta administración y compromete a todos los involucrados del proyect en suministrar todos los recursos para que este se cumpla en Plazos, Costos, Alcance y Calidad. En el Anexo A de Técnicas y Herramientas,daremos a conocer un ejemplo de este tipo de documento.

3.5.1.6 Salidas

- Acta Constitución del Proyecto
- Informe de estimaciones iníciales
- Documento o presentación de Kick-Off del proyecto

3.5.1.7 Lista de Verificación (Checklist)

En la tabla 3.5.1se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Está bien definido e identificado el Sponsor del proyecto y cual serán los mecanismos de financiamiento del mismo?				
2	¿Están todos los Stakeholder o grupos de interés identificados?				
3	¿Existe un plan de gestión de alcance inicial del proyecto?				
4	¿Están todos los antecedentes iníciales, procesos de negocio, documentación inicial, etc. adecuadamente documentado y disponible para su análisis inicial?				
5	¿Se cuenta con la información mínima para la elaboración y constitución del proyect?				
6	¿Se verifico que le proyecto tenga un caso de negocio o alguna otra documentación costo/beneficio, que lo justifique ante la alta administración?				

7	¿Está contemplada una reunión de Kick-Off o inicial del proyecto, con los principales involucrados del mismo?				
8	¿Está bien definido e identificado el Sponsor del proyecto y cual serán los mecanismos de financiamiento del mismo?				

Tabla. 3.5.2. Lista de Verificación del proceso de planificación

3.5.1.8 Métricas

- Completitud de los antecedentes iníciales:

$$\frac{\text{Cantidad de antecedentes agregados}}{\text{Cantidad total de antecedentes suministrados}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

3.5.1.9 Involucrados

- Gerente de Proyecto.
- Comité del Proyecto.
- Representante o Líder de Calidad (QA, QC)

3.5.2 PLANIFICACION

3.5.2.1 Objetivo

- Determinar qué recursos estarán disponibles para producir y mantener el softwareasociado al proyecto.
- Determinar cuándo y cómo se incurrirá en costos de personal, tiempo de procesamiento,etc.
- Medir el avance del desarrollo del proyecto.

3.5.2.2 Entrada

- Plan de desarrollo (de software) del proyecto.
- Estimación del proyecto y método utilizado para realizarla.
- Descripción del proceso de desarrollo a utilizar en el proyecto.

3.5.2.3 Proceso

En la figura 3.5.2 se detalla gráficamente, el proceso a desarrollar durante este módulo:

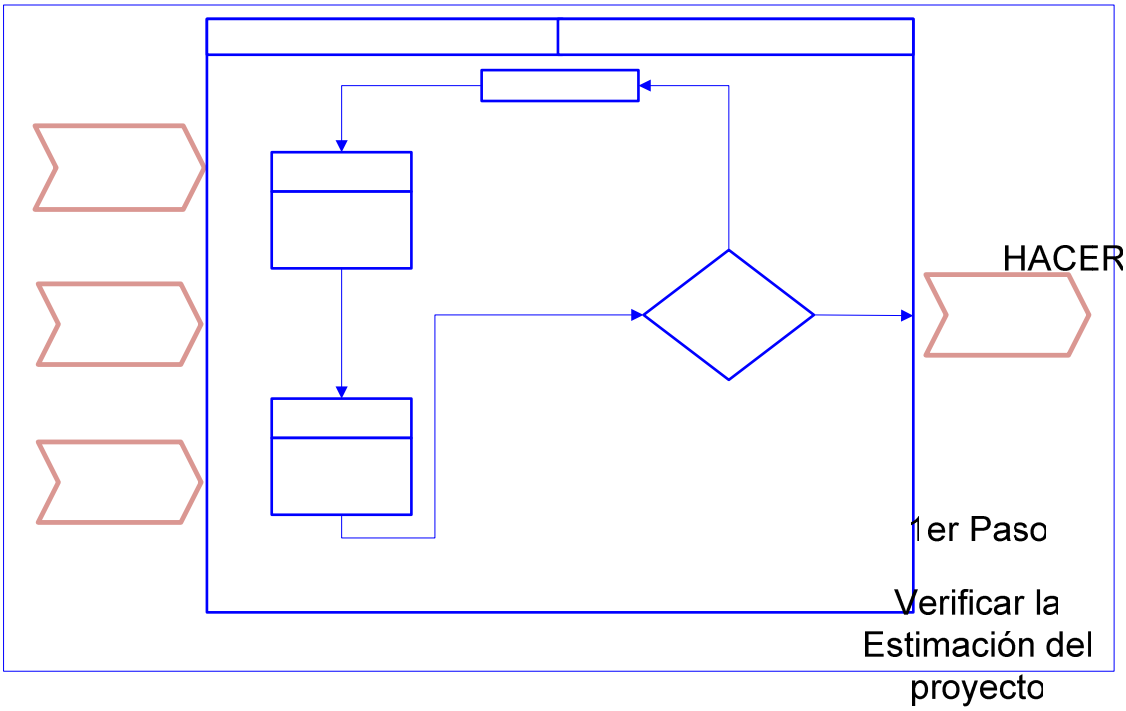


Figura. 3.5.2. Proceso del módulo de planificación

3.5.2.4 Verificación de la Estimación del Proyecto

Muchos proyectos de software son esencialmente innovadores, pero una ineficazestimación del sistema, puede llevar a un exceso de costos. Estimar costos de software esun proceso complicado porque el desarrollo del proyecto es influenciado por un largonúmero de variables, muchas de ellas son subjetivas, no cuantificables e interrelacionadasen forma compleja.

Una estimación inapropiada de costos puede dañar más a la calidad del proyecto desoftware que a cualquier otro factor.

Si la estimación es incorrecta, el equipo de proyecto hará lo imposible para coincidir con laestimación. Todo esto provoca un aumento de costos de mantenimiento, insatisfacción delcliente, esfuerzo adicional en el área del cliente para compensar la debilidad del sistema, ydesanimar al personal del proyecto. La verificación puede aumentar la validez de laestimación. La estimación del software es un proceso de tres partes como se describe acontinuación:

- Validar la sensatez del modelo de estimación.

- Validar que el modelo incluya todos los factores necesarios.
- Verificar la efectividad del modelo estimado de costo.

El detalle de cada uno de estos puntos, se encuentra en el Anexo A de Técnicas yHerramientas.

3.5.2.5 Verificación del Estado del Proyecto

Para conocer el estado del proyecto se sugiere un sistema simple de acumulación depuntos para medir el progreso. Luego este sistema puede compararse con el reportegerencial del proyecto. Si hay una diferencia significativa entre ambos sistemas de medicióndel progreso, el analista de calidad de software puede cuestionar la validez de los resultados producidospor la gerencia de proyecto y/ o sistema de conteo.

El sistema de puntos para realizar la medición durante el desarrollo del software, provee unmétodo objetivo, preciso y eficiente de recolección y reporte del rendimiento de lainformación en el campo de la ingeniería que a menudo carece de visibilidad. El métodoutiliza información basada en ítems de entregables de software y es obtenida como parte delproceso de desarrollo. Los resultados son fáciles de interpretar y pueden presentarse en unavariedad de formatos. El esquema es flexible y puede modificarse para proyectos grandes ychicos. El detalle del mismo, se encuentra en el Anexo A “Técnicas y Herramientas”.

3.5.2.6 Salidas

- Informe de estimación del proyecto
- Informe de estado del proyecto

3.5.2.7 Lista de Verificación (Checklist)

En la tabla 3.5.2 se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿La gerencia del proyecto, está de acuerdo en tener un equipo de Calidad de SW y evaluación de la estimación y estado del plan de desarrollo?				
2	¿El equipo de calidad conoce la estimación del progreso utilizada para el proyecto?				
3	¿El equipo de calidad conoce el método para realizar los informes de estado delproyecto?				
4	¿El equipo de calidad entiende el método utilizado para estimar y calcular?				
5	¿El equipo de calidad ha realizado una buenaverificación para determinar la validez de las estimaciones realizadas?				
6	Si el equipo de calidad está en desacuerdo con lavalidez de las estimaciones realizadas.¿Existe un procedimiento razonablepara manejar tales diferencias?				
7	¿El equipo del proyecto posee un sistema de reportes razonable parainformar el estado del mismo?				
8	¿El equipo de calidad ha establecido que pueden utilizarse los informes de estado de proyecto como guía para la toma de decisiones?				
9	¿Existe algún procedimiento a seguir, si los informes de estado indican que elproyecto está por encima o por debajode las estimaciones?				
10	¿El equipo de calidad ha tenido en cuenta los factores más importantes en la evaluación de laestimación realizada (Ejemplo: tamañodel software, etc.)?				
11	¿El equipo de proyecto ha recibido unacopia del estado del mismo?				
12	¿El equipo de calidad tiene conocimiento de cómo se efectúa la planificación?				
13	¿El equipo de calidad tiene conocimiento de cómo se efectúa la estimación del proyecto?				
14	¿El equipo del proyecto tieneconocimiento del proceso de desarrollo utilizado para construir el software especificado por el proyecto?				
15	¿El plan de proyecto está completo?				
16	¿La estimación del proyecto estátotalmente documentada?				
17	¿El proceso de desarrollo estátotalmente documentado?				
18	¿El método de estimación utilizado parael proyecto, es razonable respecto de las características del mismo?				
19	¿La estimación efectuada es razonable como para completar el proyecto según lo especificado en el				

	plan?				
20	¿El equipo del proyecto tiene un métododefinido para determinar e informar elestado del mismo?				
21	¿El equipo de calidad, está de acuerdo con que el estado informado coincide con el estadoactual del proyecto?				

Tabla. 3.5.2.Lista de Verificación del proceso de planificación

3.5.2.8 Métricas

- Eficacia de la estimación de la duración:

$$\frac{\text{Duración actual del proyecto}}{\text{Duración estimada del proyecto}}$$

- Eficacia de la estimación del esfuerzo:

$$\frac{\text{Esfuerzo actual del proyecto}}{\text{Esfuerzo estimado del proyecto}}$$

- Eficacia de la estimación del Costos:

$$\frac{\text{Costo actual del proyecto}}{\text{Costo estimado del proyecto}}$$

3.5.2.9 Involucrados

- Gerente de Proyecto.

- Comité del Proyecto.
- Representante o Líder de Calidad (QA, QC)

3.5.3 REQUERIMIENTO

3.5.3.1 Objetivo

- Determinar si los requerimientos expresan las necesidades del usuario.
- Verificar la separación de los requerimientos entre obligatorios y opcionales, además de su clasificación entre Funcionales y No-Funcionales.
- Verificar que cada requerimiento:
    - Esté debidamente documentado, reflejando las necesidades del usuario.
    - Se encuentre dentro del alcance definido del sistema en cuestión.
    - Se encuentre dentro de los límites del presupuesto para el sistema en cuestión.
    - Se encuentre bien definido (en forma completa), de manera de evitar cambios posteriores.
- Determinar que los requerimientos documentados puedan ser "implementables" (posibles de ser analizados, diseñados, codificados) y "verificables" (pasibles de serverificados y validados) en fases posteriores.

3.5.3.2 Entrada

- Minutas y documentación levantada en reunión con los usuarios.
- Documento de especificación de requerimientos del usuario.
- Descripción del proceso de desarrollo a utilizar en el proyecto.

3.5.3.3 Proceso

En la figura 3.5.3 se detalla gráficamente, el proceso a desarrollar durante este módulo:

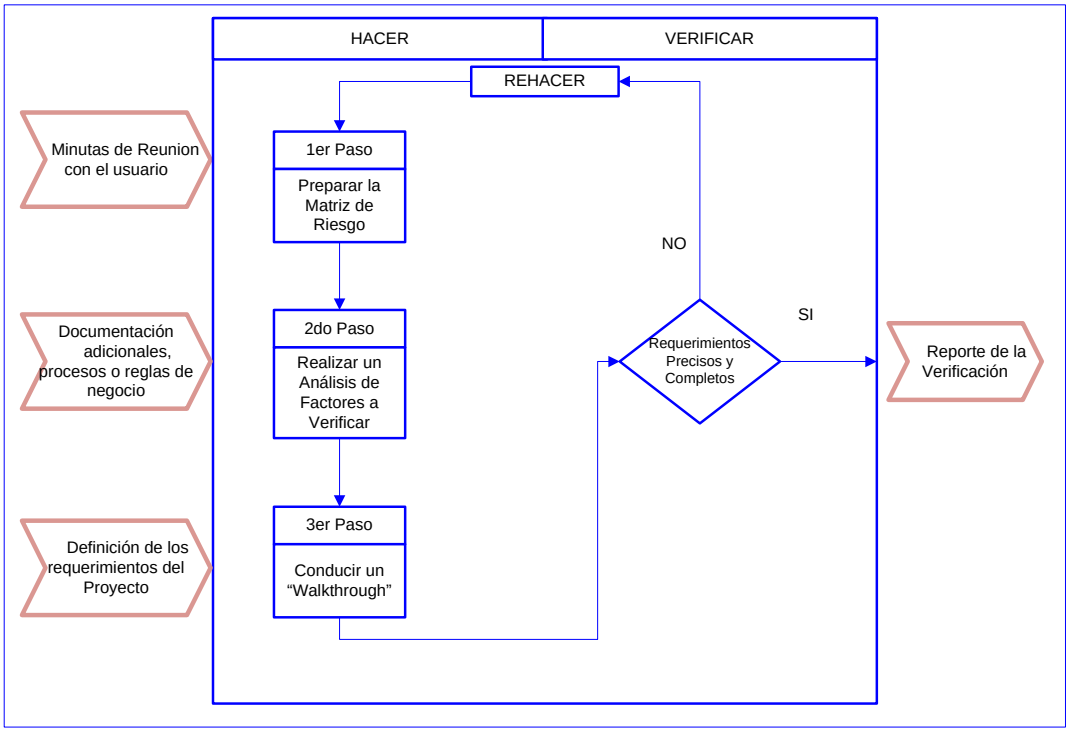


Figura. 3.5.3. Proceso del módulo de Requerimientos

3.5.3.4 Preparar la Matriz de Riesgo

La Matriz de Riesgo es una herramienta diseñada para evaluar el control en los sistemasde información.

Uno de los mayores beneficios de la matriz de riesgo es, no sólo la identificación de losmismos, sino también qué es lo que el sistema debe hacer para gestionar cada uno de ellos. Enprincipio, la matriz de riesgo es una herramienta de diseño, pero puede ser usada comoherramienta de verificación.

En forma ideal la matriz de riesgo comienza en la fase de requerimientos, se desarrolla enla fase de análisis y se termina en la fase de diseño. La ejecución de la matriz de riesgo esun proceso de cinco pasos, los cuales se encuentran detallados en el Anexo A "Técnicas yHerramientas."

3.5.3.5 Realizar un Análisis de Factores a Verificar

Debe llevarse a cabo un proceso para estimar las incumbencias (“concerns”) asociadas ala fase de requerimientos del desarrollo del sistema. Debe incluirse un programa deverificación para cada factor a considerar (son 15), cubriendo cada fase del proceso dedesarrollo. Para cada factor hay un programa de verificación que tiene en cuenta ciertasconsideraciones las cuales se encuentran detalladas en el Anexo A “Técnicas y Herramientas”

El programa de verificación enumera aquellos criterios, que aseguran al equipo de calidad de Software, que la magnitud de esa preocupación es mínima. A estoscriterios debe responder el equipo de calidad. También se debe realizaruna verificación suficiente para evaluar si el equipo de proyecto ha manejadoadecuadamente cada criterio de verificación.

Los factores a verificar en la etapa de requerimientos son:

- o Requerimientos compatibles con la metodología (Factor de prueba: Metodología).
- o Funcionalidad de las especificaciones definidas (Factor de prueba: Precisión)
- o Usabilidad de las especificaciones (Factor de prueba: Facilidad de uso)
- o Mantenimiento de las especificaciones (Factor de prueba: Mantenibilidad)
- o Necesidades de portabilidad (Factor de prueba: Portabilidad)
- o Interfaces del sistema (Factor de prueba: Acoplamiento)
- o Criterios de performance (Factor de prueba: Performance)
- o Necesidades operativas (Factor de prueba: Facilidad de operación)
- o Tolerancia (Factor de prueba: Fiabilidad)
- o Reglas de autorización (Factor de prueba: Autorización)
- o Requerimientos de integridad de archivos (Factor de prueba: Integridad de archivos)

- Recuperación ante fallas en los requerimientos (Factor de prueba: Auditoría)
- Impacto de fallas (Factor de prueba: Continuidad de procesamiento)
- Nivel de servicio deseado (Factor de prueba: Nivel de Servicio)
- Permisos y accesos (Factor de prueba: Seguridad)

3.5.3.6 Conducir un "Walkthrough[31]" de Requerimientos

De los procesos de revisión, la inspección o recorrido (walkthrough) de requerimientos es el menos estructurado y el más propenso a la creatividad. Por lo tanto éste se convierte en un proceso de revisión que complementa los objetivos de la fase de requerimientos. El objeto de este proceso de revisión es crear una situación en la cual un grupo de gente conlas habilidades adecuadas pueda ayudar al equipo en el desarrollo de las soluciones delproyecto. El "walkthrough" intenta usar la experiencia y juicio del equipo de revisión como unsoporte en el proceso de desarrollo.

Este método se implementa como un proceso de cinco pasos ejecutados en secuencia, loscuales se encuentran detallados en el Anexo A "Técnicas y Herramientas", los cuales son:

- Establecer reglas básicas
- Seleccionar el equipo / Notificar a los participantes
- Presentación del proyecto
- Preguntas / recomendaciones
- Reporte final

El tiempo destinado a cada paso dependerá del tamaño de la aplicación que se estérevisando y el grado de asistencia deseado del equipo de "walkthrough".

3.5.3.7 Salidas

- Matriz de riesgo

[31]Walkthrough, término que indica inspección en terreno o caminatas para determinar defectos en fases tempranas del desarrollo

- Análisis de factores de verificación.
- Informe de la inspección (“walkthrough”) confeccionado.
- Resumen de la inspección (“walkthrough”) confeccionado.
- Informe de estado del proyecto.

3.5.3.8 Lista de Verificación (Checklist)

En la tabla 3.5.3 se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Los requerimientos definidos son verificables?				
2	¿El usuario está de acuerdo con el requerimiento definido?				
3	¿Los desarrolladores entienden los requerimientos?				
4	¿Los desarrolladores saben que deben mantener la trazabilidad de los Requerimientos?				
5	¿El requerimiento definido coincide con los objetivos del proyecto?				
6	¿Los requerimientos fueron clasificados en funcionales y no funcionales, además de ser priorizados?				
7	¿Se identificaron los riesgos del proyecto?				
8	¿Se siguió un proceso razonable en la definición del requerimiento?				
9	¿El proceso de control de requerimientos, es adecuado para minimizar los riesgos del proyecto?				
10	¿Durante el proceso de control de requerimientos, se ha llevado a cabo un “walktrough”?				

Figura. 3.5.3Lista de verificación del proceso de Requerimientos

3.5.3.9 Métricas

- Estabilidad de los requerimientos:

$$\frac{\text{Cantidad de requerimientos iníciales}}{\text{Cantidad total de requerimientos}}$$

- Madurez de los requerimientos :

$$\frac{\text{Cantidad de requerimientos cambiados}}{\text{Cantidad total de requerimientos}}$$

- Completitud de los requerimientos:

$$\frac{\text{Cantidad de requerimientos agregados}}{\text{Cantidad total de requerimientos}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

3.5.3.10 Involucrados

- Gerente de Proyecto.
- Grupo de análisis de requerimientos.
- Líder o Analista de Calidad de software (QA, QC)

3.5.4 ANALISIS

3.5.4.1 Objetivo

- Describir la funcionalidad del producto que va a ser entregado.
- Definir el ambiente en el que va a operar.
- Definir la confiabilidad y el compromiso de mantenimiento.
- Identificar necesidades de soporte u operación, como hardware, software, etc.
- Definir para cada requerimiento, qué prueba se va a ejecutar para asegurar que secumpla con el mismo.

3.5.4.2 Entrada

- Documento de especificación de requerimientos revisado
- Documento de especificación funcional y no funcionales.

3.5.4.3 Proceso

En la figura 3.5.4 se detalla gráficamente el proceso de este módulo:

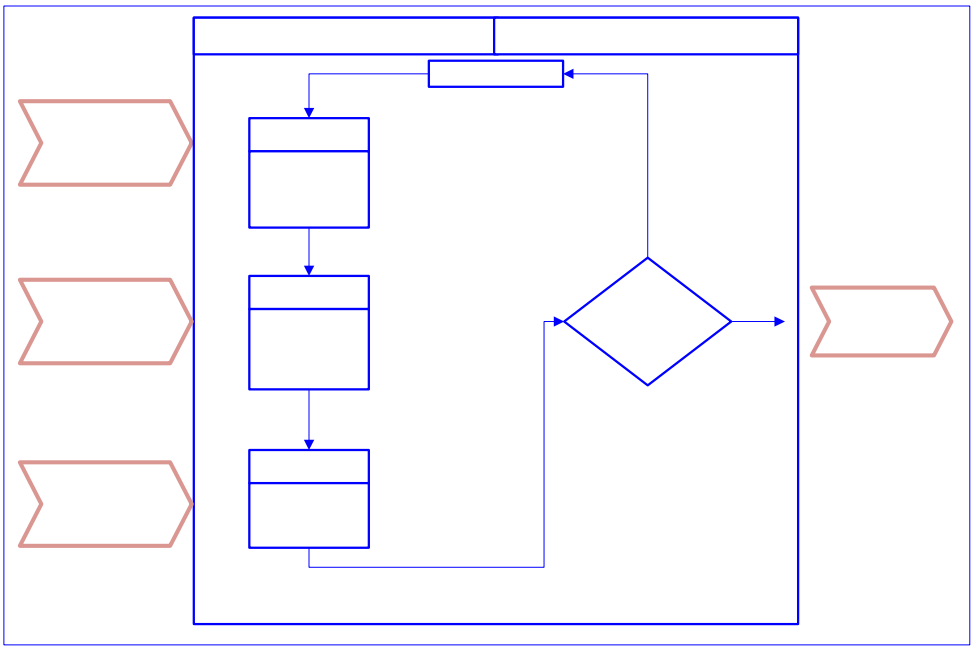


Figura. 3.5.4. Proceso del módulo de Análisis

3.5.4.4 Analizar la Especificación Funcional

La especificación funcional, debe ser vista como un proceso de representación, a fin depoder llegar a una exitosa implementación de software. Algunos principios para especificar,podrían ser:

- Separar requerimientos funcionales de implementación o no funcionales
- Desarrollar un modelo del comportamiento deseado del sistema, que comprenda datos yla respuesta funcional del sistema a los estímulos del ambiente.
- Establecer un contexto en el cual el software opera especificando la forma en la cualotros componentes del sistema interactúan con el software.
- Desarrollar un ambiente en el cual el sistema opere y reaccione a los estímulos delmismo.
- Crear un modelo cognitivo en lugar de un diseño o un modelo de implementación. Elmodelo cognitivo describe el sistema como es percibido por los usuarios de esacomunidad.
- Reconocer que una especificación tendrá varios niveles de detalle y profundidad.

Verificar que la especificación funcional cumpla, con al menos, los siguienteslineamientos:

- El formato de la representación y contenido debería ser relevante para el problema.Debe llevarse a cabo, un desarrollo general para los contenidos de las especificaciones.
- La información contenida en la especificación debe ir de lo general a lo particular (estaranidada). Las representaciones deberían revelar capas de información a fin de permitir allector que pueda moverse al nivel de detalle que sea requerido. Se deben indicaresquemas de numeración dentro de los diagramas utilizados, indicando el nivel dedetalle que se presenta. Es

importante presentar la misma información a diferentesniveles de abstracción para ayudar a su entendimiento.

- Los formatos diferentes de diagramas a utilizar y otras formas de notación, deberán serrestringidas al mínimo en número y consistentes en su uso. La información confusa einconsistente, sea tanto simbólica o gráfica, perjudica el entendimiento y propicia errores.
- La especificación funcional, debe ser verificable.

Verificar que la especificación funcional cumpla, con al menos, el siguiente contenido:

I. Introducción
    a. Resumen del Sistema
    b. Descripción del producto
    c. Entregables
II. Ambiente
    a. Usuarios
    b. Servicios
    c. Físico
    d. Seguridad
III. Descripción de la información
    a. Representación del contenido de la información
    b. Representación del flujo de la información
        i. Flujo de datos
        ii. Flujo de control
IV. Descripción funcional
    a. Partición funcional
    b. Descripción funcional
        i. Narrativa de proceso general
        ii. Restricciones / Limitaciones
        iii. Requerimientos de rendimiento
        iv. Restricciones de diseño
        v. Diagramas de soporte
    c. Descripción de control
        i. Especificación de control
        ii. Restricciones de para el diseño
V. Descripción de situación

a. Estado del sistema
b. Eventos y acciones

VI. Validación y criterios
a. Límites y restricciones
b. Clases de pruebas requeridas
c. Respuesta esperada del software
d. Consideraciones especiales

VII. Cronograma

VIII. Apéndices

3.5.4.5 Analizar Procesos y Reglas de Negocio Adicional

Junto con el análisis de la especificación funcional, se deben tener en cuenta los procesos de negocio, procedimientos, y cualquier otra fuente de información que permita mejorar el conocimiento de la problemática planteada y la solución a proponer, para esto se deben tener en cuenta los siguientes aspectos:

- Comprender a distintos niveles de abstracción los procesos de negocio que serán afectados por el nuevo proyecto, así mismo quien será el encargado de su actualización y cómoserá la interacción con el equipo de proyecto
- Desarrollar resúmenes y entendimiento de los procedimientos más importantes de la organización involucrada en el proyecto.
- Considerar documentación adicional que sea relevante para el proyecto

3.5.4.6 Conducir una Inspección Formal

La revisión de una especificación funcional es conducida por el desarrollador y el cliente.Dado que las especificaciones forman el punto fundamental y a partir del cual se realizará eldiseño y subsecuentemente las actividades de la ingeniería del software o de conocimiento,se deberán extremar los cuidados para conducir esta revisión.

La revisión en una primera instancia es conducida en un nivel macro. En este nivel, losinspectores intentan garantizar que la especificación esté completa, consistente y exacta.

Se tendrán en cuenta los siguientes lineamientos para una revisión detallada:

- Detectar y remplazar términos vagos.
- En el caso de listas de elementos, asegurar que hayan sido incluidos todos.
- Cuando se especifiquen rangos (numéricos, etc.) asegurarse que no contengan valoresasumidos sin ser definidos expresamente.
- Estar alerta de verbos y pronombres ambiguos.
- Identificar afirmaciones que no incluyan certeza.
- Cuando una estructura es descripta en palabras, asegurar que se realice un diagramapara ayudar a comprenderla.
- Cuando sea necesario realizar un cálculo especificado, asegurar que se presente almenos dos ejemplos del mismo.

Con respecto a los cambios en los requerimientos, una vez que los mismos han sido finalizados, el usuario deberá notar que cada uno de ellos es una ampliación del alcance delsoftware y por ende, un cambio en la especificación funcional. Esto incrementará el costodel proyecto y/ o extenderá el tiempo establecido para su finalización.

Aún cuando se realicen las mejores revisiones, cierto número de problemas en laespecificación persiste. La especificación es difícil de verificar, por ende las inconsistenciasu omisiones pueden pasar desapercibidas. Durante la revisión, pueden ser recomendadosmás cambios en la especificación funcional. Puede ser extremadamente difícil estimar elimpacto global del cambio, esto es, cómo un cambio en una función afecta a losrequerimientos en otra función. Esto último deber ser tenido muy en cuenta a la hora de“impactar” dicho cambio.

3.5.4.7 Salidas

- Informe de la Inspección.
- Resumen de la Inspección.

3.5.4.8 Lista de Verificación (Checklist)

La tabla 3.5.4muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Los objetivos y metas del sistema se mantienen consistentes con las políticas de software de la organización?				
2	¿La estructura y el flujo de la información, está adecuadamente definida por el área a la cual compete el problema?				
3	¿Son claros los diagramas? ¿Pueden ser explícitos sin necesidad de ser descriptos o narrados?				
4	¿Las funciones principales del sistema, están dentro del alcance? ¿Cada una de ellas ha sido adecuadamente definida?				
5	¿Es consistente el comportamiento del sistema con la información que debe procesar y las funciones que debe desarrollar?				
6	¿Las limitaciones del sistema son realistas?				
7	¿Ha sido considerado el riesgo tecnológico del desarrollo?				
8	¿Se han considerado requerimientos alternativos de software?				
9	¿Ha sido detallado el criterio de verificación y validación? ¿Los mismos, son adecuados para determinar el éxito del sistema?				
10	¿Existen inconsistencias, omisiones o redundancias en el modelo de información del sistema?				
11	¿El usuario final ha estado involucrado?				
12	¿El usuario revisó el prototipo y/o manual del usuario?				
13	¿Han sido debidamente planificadas las estimaciones de esfuerzo?				

Tabla. 3.5.4Lista de verificación del proceso de Análisis

3.5.4.9 Métricas

- Estabilidad de los requerimientos:

$$\frac{\text{Cantidad de requerimientos modificados}}{\text{Cantidad total de requerimientos establecidos}}$$

- Estabilidad de la funcionalidad:

$$\frac{\text{Cantidad de funciones modificadas}}{\text{Cantidad total de funciones establecidas}}$$

- Corrección de la funcionalidad:

$$\frac{\text{Cantidad de funciones corregidas}}{\text{Cantidad total de funciones establecidas}}$$

- Completitud de la funcionalidad:

$$\frac{\text{Cantidad de nuevas funciones agregadas}}{\text{Cantidad total de funciones establecidas}}$$

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados en análisis}}{\text{Cantidad total de defectos}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos en análisis}}{\text{Cantidad total de defectos}}$$

- Tasa de efectividad de la verificación :

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

3.5.4.10 Involucrados

- Gerente de Proyecto.

- Grupo de análisis de requerimientos.
- Grupo de análisis funcional.
- Líder o Analista de Calidad de software (QA, QC)

3.5.5 DISEÑO

3.5.5.1 Objetivo

- Establecer los factores que conducen a un diseño correcto y preciso
- Conducir revisiones de diseño.
- Conducir inspecciones de los entregables del diseño

3.5.5.2 Entrada

- Comprensión del diseño por parte del equipo de calidad de software
- Entregables derivados del diseño
- Especificaciones de entrada
- Especificaciones de procesamiento
- Especificaciones de archivos
- Especificaciones de salida
- Especificaciones de control
- Diagramas de flujo del sistema
- Requerimientos de software y hardware.
- Especificación de los procedimientos de operación manual.
- Política de resguardo de la información.
- Documento de diseño de alto nivel (Lógico)
- Documento de diseño de detalle (Físico y lógico)

3.5.5.3 Proceso

En la figura 3.5.5 se detalla gráficamente, el proceso a desarrollar durante este módulo:

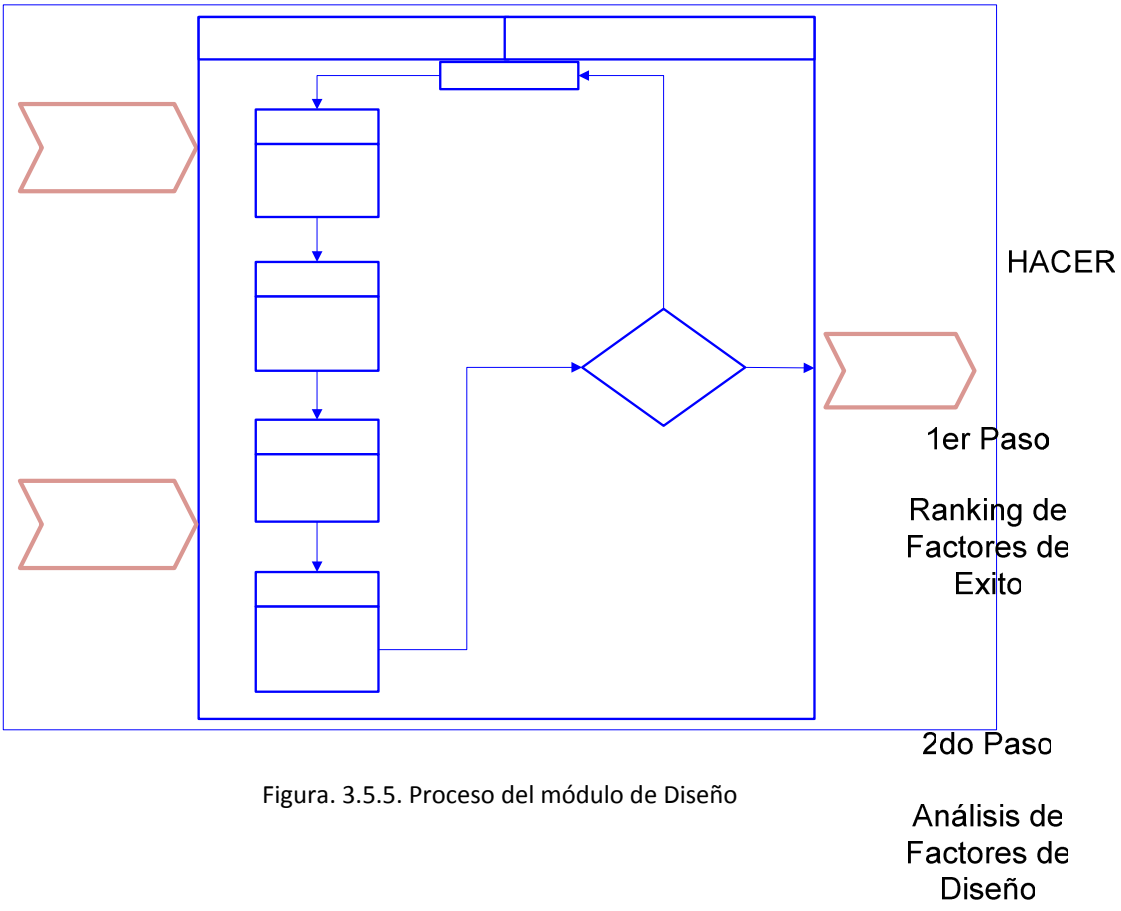


Figura. 3.5.5. Proceso del módulo de Diseño

3.5.5.4 Ranking de Factores de Éxito

El ranking (“scoring”) es una herramienta predictiva que utiliza experiencias en proyectosanteriores. Los proyectos en curso son analizados para determinar los atributos de losmismos y su correlación con el éxito o el fracaso de proyectos o sistemas en particular. Unavez que los atributos correlacionados al éxito o al fracaso pueden ser identificados, tambiénpueden ser usados para predecir el comportamiento del sistema que está en desarrollo.

El concepto es el mismo que el usado para predecir el resultado de las elecciones. Lagente que hace las predicciones procura identificar distritos electorales que muestran unacorrelación histórica alta en los resultados de elecciones anteriores. El conocimiento delregistro de votos de un distrito en especial y los resultados de elecciones previas, nosprovee la base para hacer futuras predicciones. Esos distritos con un registro de grandesganadores pueden servir de muestra, para luego basados en esas muestras, predecir elresultado de una elección antes que los votos sean contados.

Estos tipos de predicciones en sistemas de computación son bien conocidos, pero no hansido bien utilizados, hasta que el concepto de ranking fue desarrollado. Por ejemplo,sabemos que hay una correlación positiva entre la participación del usuario y el sistema decomputación: cuanto más involucrado este el usuario, mayor probabilidad de que el sistemasea exitoso. También sabemos que habrá problemas con el sistema de computación queempuja las actuales corrientes de tecnología. Por ejemplo, el primer sistema en usar lanueva tecnología puede ser identificado como un sistema de alto riesgo, porque la mayoríade los inconvenientes ocurrirá durante la implementación.

Los criterios que se utilizan bajo el concepto de ranking, se describen en el Anexo A “Técnicas y Herramientas”

Al concluir el ranking, el resultado puede ser usado en cualquiera de las siguientes formas:

- Estimar la extensión de la verificación: A mayor riesgo, la gerencia del proyecto puederequerir más verificaciones. Sabiendo que la aplicación es de alto riesgo, se alertará alos gerente del proyecto respecto de la necesidad de tomar las medidascorrespondientes, para reducir el riesgo a un nivel aceptable.
- Identificar módulos a verificar: Dependiendo de la sofisticación del instrumento con elque se obtuvo el ranking, puede identificarse módulos específicos para la verificación.Por ejemplo, si la lógica computacional se sabe que es de alto riesgo, la verificacióndebería evaluar exhaustivamente la exactitud de este proceso.
- Identificar la composición del equipo de verificación: Los tipos de riesgos asociados conel sistema ayudan a determinar la composición del equipo de test. Por ejemplo, si losriesgos tratan más con la tecnología que con la lógica, el equipo de test debería incluirindividuos con conocimientos profundos en esa tecnología.

Al ranking de riesgo se llega a través de un desarrollo matemático, como se muestra en el Anexo A“Técnicas y Herramientas”.

3.5.5.5 Análisis de Factores

El encargado de dirigir la verificación podrá seleccionar los factores de interés apropiados,para ajustar la verificación, teniendo en cuenta los siguientes objetivos generales para lafase de diseño:

- Desarrollar una solución al problema de negocio planteado.
- Determinar el rol de la tecnología para resolver el problema.
- Desarrollar especificaciones para los segmentos manuales y automatizados del sistema.
- Cumplir con el plan de acción, procesos, procedimientos, estándares y regulaciones.
- Definir los controles que reducirán riesgos de la aplicación, llevándolos a un nivelaceptable.
- Completar el proyecto dentro de las restricciones del presupuesto, personal ycronograma establecidos, manteniendo la calidad del mismo.

Los factores que deben analizarse durante la fase de diseño se describen a continuación, ypuede encontrarse más detalle en el Anexo A “Técnicas y Herramientas”: Los factores averificar en la Fase de Diseño son:

- Controles de integridad de datos.
- Reglas de autorización.
- Controles de integridad de archivos.
- Pistas de auditoría.
- Plan de contingencia.
- Método para alcanzar el nivel de servicio requerido.
- Procedimientos de acceso.
- Diseño acorde con la metodología establecida.
- Diseño acorde con los requerimientos establecidos.

- Facilidad de uso.
- Mantenibilidad del diseño.
- Portabilidad del diseño.
- Interfaces de diseño.
- Diseño acorde con criterios establecidos.
- Necesidades de operación.

3.5.5.6 Conducción de la Revisión del Diseño

La revisión de diseño será estructurada usando la misma información básica que lautilizada para llevar a cabo el ranking. El objetivo es identificar de antemano aquellosatributos del diseño que se correlacionan con los problemas del sistema. Entonces durantela revisión del diseño, se investigarán dichos atributos para determinar si el equipo deproyecto los ha tratado apropiadamente.

La revisión del diseño deberá ser conducida por un grupo de individuos con conocimientosen el proceso de diseño. El grupo tendrá la responsabilidad de revisar la integridad yracionabilidad de la aplicación. No es necesario que el grupo conozca la aplicaciónespecíficamente pero sí debe conocer la metodología de la organización y los procesos involucrados o tocados por el proyecto.

La revisión de diseño normalmente es formal y altamente estructurada. Los miembros delequipo intentan determinar si todas las tareas se han realizado apropiadamente. Al final dela revisión de diseño, el equipo emite un reporte indicando sus hallazgos y recomendacionesacerca del proyecto. El equipo de revisión de diseño comprende los siguientes miembros:

- Personal del proyecto.
- Equipo de revisión independiente.

La siguiente es la lista con los pasos de una revisión de diseño, la descripción detalladaestá Anexo A “Técnicas y Herramientas”:

- Seleccionar el equipo de revisión.
- Entrenar los miembros del equipo de revisión.
- Notificar al equipo de proyecto.
- Asignar tiempos adecuados.
- Documentar los datos de la revisión.
- Revisar los datos con el equipo de proyecto.
- Desarrollar recomendaciones de revisión.
- Revisar recomendaciones con el equipo de proyecto.
- Preparar un reporte final.

Una o más revisiones pueden llevarse a cabo durante la fase de diseño. La cantidad derevisiones dependerá de la importancia del proyecto y del lapso de tiempo en la fase dediseño. Así podrán llevarse a cabo revisiones del diseño de alto nivel y del diseño detallado.Cada defecto descubierto durante la revisión del diseño de alto nivel debe documentarse. Seconfeccionará un reporte de defectos. Los defectos deben documentarse, categorizarse,registrarse, presentarse al equipo de proyecto para corregir, y referenciarlo al documentoespecífico en el cual el defecto fue notado. Un ejemplo del formulario para registrar estosdefectos, puede encontrarse en el Anexo A Técnicas y Herramientas.

3.5.5.7 Conducción de la Inspección de los Entregables de Diseño

La Inspección es un proceso por el cual los productos completos pero no verificados, sonevaluados para saber si los cambios especificados fueron implementados correctamente.

Para lograr esto, los inspectores o representantes de calidad examinan los productos antes del cambio, lasespecificaciones de cambio, y los productos luego del

cambio para determinar losresultados. Buscan tres tipos de defectos: Incorrectos (el cambio no se ha hechocorrectamente), faltantes (algo que debería haberse cambiado, no se cambió), y adicionales(algo fue cambiado o agregado sin la intención de hacerlo).

3.5.5.8 Salidas

- Informe de la Inspección.
- Resumen de la Inspección.

3.5.5.9 Lista de Verificación (Checklist)

En la tabla 3.5.5 se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿El equipo de calidad tiene buenos conocimientos acerca del proceso de diseño?				
2	Si se han utilizado herramientas automatizadas en la creación del diseño, ¿las conocen los miembros del equipo de calidad?				
3	¿Han recibido los miembros del equipo de calidad todos los entregables de la fase necesarios para llevar a cabo las verificaciones correspondientes?				
4	¿Los usuarios están de acuerdo con que el diseño representa la realidad?				
5	¿El equipo de proyecto cree que el diseño representa la realidad?				
6	¿Han identificado los miembros del equipo de aseguramiento de la calidad los factores de éxito, tanto positivos como negativos, que pudieran afectar el éxito del diseño?				
7	¿Han utilizado los miembros del equipo calidad, talesfactores en la puntuación de las probabilidades de éxito?				
8	¿Entienden los miembros del equipo de calidad los factores relacionados con el diseño?				
9	¿Han analizado los miembros del equipo de calidad los factores de test del diseño para evaluar el potencial impacto sobre el éxito del mismo?				
10	¿Ha sido instruido el equipo de Revisión acerca de loque representa el éxito del Diseño?				
11	¿La Gerencia apoya el uso del proceso de revisión del diseño?				

12	¿El proceso de revisión del diseño fue conducido en un momento apropiado?				
13	¿Son razonables los ítems identificados en el proceso de revisión del diseño?				
14	¿Están de acuerdo los miembros del equipo de la calidadque los ítems identificados necesitan serverificados?				
15	¿La Gerencia apoya la ejecución de inspecciones en el proyecto?				
16	¿Se ha provisto del tiempo suficiente en el cronograma del proyecto para realizarinspecciones?				
17	¿Han sido instruidos los responsables del proyecto acerca de la importancia de la participación en las inspecciones?				
18	¿La Gerencia ve las inspecciones como una parte integral del proceso, en lugarde tomarlo como una auditoría al desempeño de los participantes?				
19	¿Han sido planificados los procesos de Inspección?				
20	¿Han sido identificados los inspectoresy asignado sus roles específicos?				
21	¿Han sido entrenados los inspectores para cumplir con su rol?				
22	¿Se les ha dado a los inspectores los materiales necesarios para cumplir conla inspección?				
23	¿Se les ha dado a los inspectores tiempo adecuado para realizar las tareas habituales que le permitirán cumplir con la preparación y la reunión para el proceso de inspección?				
24	¿Se han preparado adecuadamente los inspectores para la inspección?				
25	¿Han preparado los inspectores una lista de defectos?				
26	¿Se ha programado la inspección convenientemente para todos los inspectores?				
27	¿Han concurrido todos los inspectores a la reunión de inspección?				
28	¿Estuvieron de acuerdo todos los inspectores con lalista de defectos?				
29	¿Fueron identificados los defectos durante la reunión de revisión registrados y entregados al autor?				
30	¿El autor estuvo de acuerdo acerca derealizar las correcciones necesarias?				
31	¿Se ha desarrollado un proceso razonable para determinar si esos defectos han sido corregidos satisfactoriamente?				
32	¿La certificación del moderador ha sido tomada en cuenta para el producto / entregable inspeccionado?				

Tabla. 3.5.5Lista de verificación del proceso de Diseño

3.5.5.10 Métricas

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados en diseño}}{\text{Cantidad total de defectos hasta diseño}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos de diseño}}{\text{Cantidad total de defectos hasta diseño}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

- Estabilidad del diseño:

$$\frac{\text{Cantidad de cambios introducidos en el diseño}}{\text{Cantidad total de cambios en el proyecto}}$$

3.5.5.11 Involucrados

- Gerente de Proyecto.
- Grupo de diseño o representante de alto nivel (o lógico).
- Grupo de diseño o representantede detalle (o físico).
- Grupo de administración o representante de base de datos.
- Grupo de analistas desarrolladores.
- Líder de o representante de Calidad del Software

3.5.6 CODIFICACION

3.5.6.1 Objetivo

El principal objetivo de esta fase es asegurar que las especificaciones de diseño hayansido correctamente implementadas.

3.5.6.2 Entrada

Los entregables que funcionan como entrada al proceso de verificación de codificación sonlos siguientes:

- Especificaciones de programas.
- Documentación de programas.
- Listado de código.
- Programas ejecutables.
- Diagramas de flujo de los programas.
- Instrucciones para los administradores y/o operador.

3.5.6.3 Proceso

La figura 3.5.6 detalla gráficamente, el proceso a desarrollar durante este módulo:

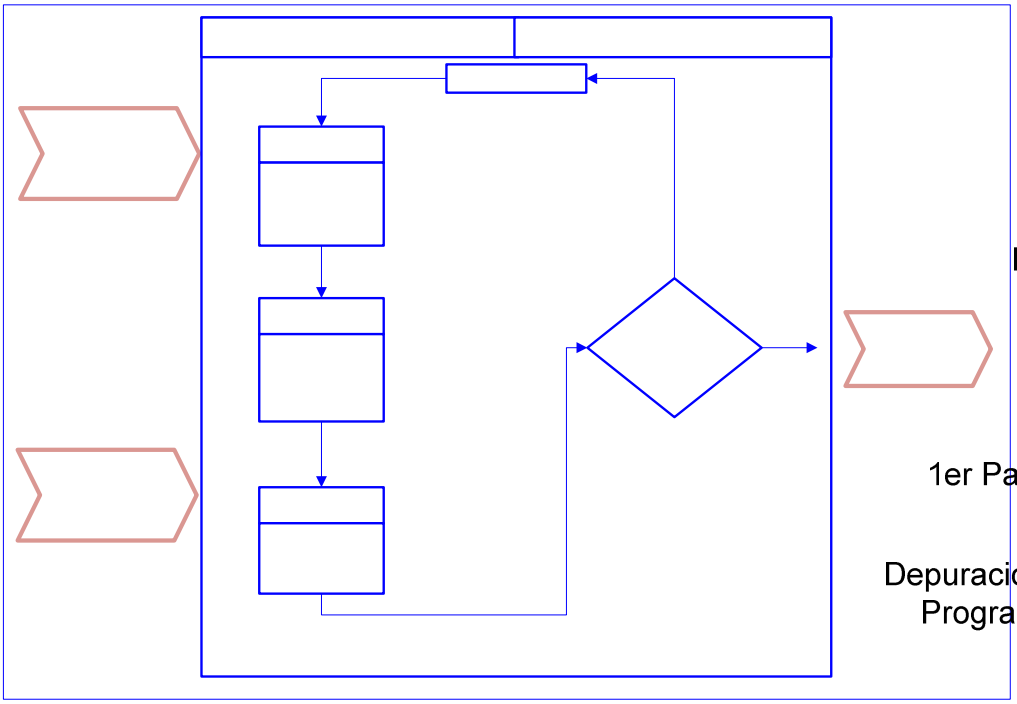


Figura. 3.5.6. Proceso del módulo de Codificación

3.5.6.4 Depuración de Programas

Se utilizarán tres metodologías para la detección y eliminación de los errores en lacodificación:

- Depuración de errores sintácticos.
- Depuración de errores estructurales.
- Depuración de errores funcionales.

El detalle de cada una de ellas está en el Anexo A“Técnicas y Herramientas”

3.5.6.5 Análisis de Factores de Codificación

A continuación, se detallan los factores a analizar,los cuales están especificados en el, Anexo A de “Técnicas y Herramientas”:

- Implementación del control de integridad de información.
- Implementación de reglas de autorización.
- Implementación de control de integridad de archivos.
- Implementación de auditorías de rastreo.
- Plan de contingencia escrito.
- Consecución del nivel de servicio deseado (SLA) para el sistema.
- Implementación de procedimientos de seguridad.
- Cumplimiento del programa con la metodología a adoptar.
- Programa acorde al diseño (Correctitud).
- Programa acorde al diseño (Facilidad de uso).
- Mantenimiento del programa.
- Programa acorde al diseño (Portabilidad).
- Programa acorde al diseño (Acoplamiento).
- Desarrollo de procedimientos operativos.
- Criterio de ejecución del programa (Performance).

3.5.6.6 Conducción de Revisiones de Pares

Se utiliza la revisión por pares, para evaluar la estructura y funcionalidad del programa. Larevisión por pares detecta errores sintácticos, más por observación personal que por elresultado directo del “Walktrough”.

3.5.6.7 Salidas

- La eliminación de todos los errores de codificación utilizando pruebas estáticas.
- Listado de errores encontrados.

3.5.6.8 Lista de Verificación (Checklist)

En la tabla 3.5.6 se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Es considerada la responsabilidad del programador o desarrollador en la verificación y la validación de los programas?				
2	¿El programador o diseñador entiende la diferencia entre testing estático y dinámico?				
3	¿El programa o diseñador podría ser sometido atesting estático como medio primario para remover defectos?				
4	¿El programador o diseñador entiende el proceso funcional o de negocio que generará el código del programa?				
5	¿El programador o desarrollador entiende y usa la técnica de “debugging”?				
6	¿El programador o desarrollador entiende los conceptos implicados, y los tendrá presentes en la verificación de la programación?				
7	¿El programador o desarrollador es verificado usando revisión por pares o inspección de código?				
8	¿El programa será sometido a un test completo antes de ingresar a un nivelsuperior de test?				
9	¿Todos los defectos no cubiertos estánregistrados en detalle?				
10	¿Todos los defectos no cubiertos fueron corregidos antes de ingresar al siguientenivel de verificación?				

Tabla. 3.5.6 Lista de verificación del proceso de codificación

3.5.6.9 Métricas

- Densidad de defectos por componente:

$$\frac{\text{Cantidad de defectos}}{\text{Tamaño del componente}}$$

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados}}{\text{Cantidad total de defectos}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos}}{\text{Cantidad total de defectos}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

3.5.6.10 Involucrados

- Gerente de Proyecto.
- Grupo o representante del diseñador o programador
- Líder de o representante de Calidad del Software

3.5.7 VERIFICACION DEL SISTEMA

3.5.7.1 Objetivo

El objetivo es determinar si el software funcionará correctamente cuando se ejecute. Paraello:

- Se verificará que se haya probado la ejecución del software en un ambiente separado(Alpha y Beta), siendo aproximadamente el mismo entorno operacional en que será ejecutadoluego (producción).
- Las pruebas serán verificadas, una vez terminada su ejecución y aprobación interna.
- Cualquier desviación de los resultados esperados será reportada.

Dependiendo de la severidad y naturaleza de dichos problemas, podrían llegar a realizarsesolicitudes de cambios (RFC) al sistema antes de su puesta en producción.

Los posibles cambios deberán ser evaluados sus impactos en los parámetros principales del proyecto, tales como Tiempo, Costos, Alcances y Calidad.

3.5.7.2 Entrada

- Planes de pruebas:
  - Pruebas en línea (“On-Line”)
  - Pruebas por lotes (“Batch”)
  - Pruebas de interfaces.
  - Pruebas de regresión.
  - Pruebas de integración.
  - Pruebas de “stress”.
- Datos de prueba.

- Resultados de las pruebas.

3.5.7.3 Proceso

La figura 3.5.7detalla gráficamente, el proceso a desarrollar durante este módulo:

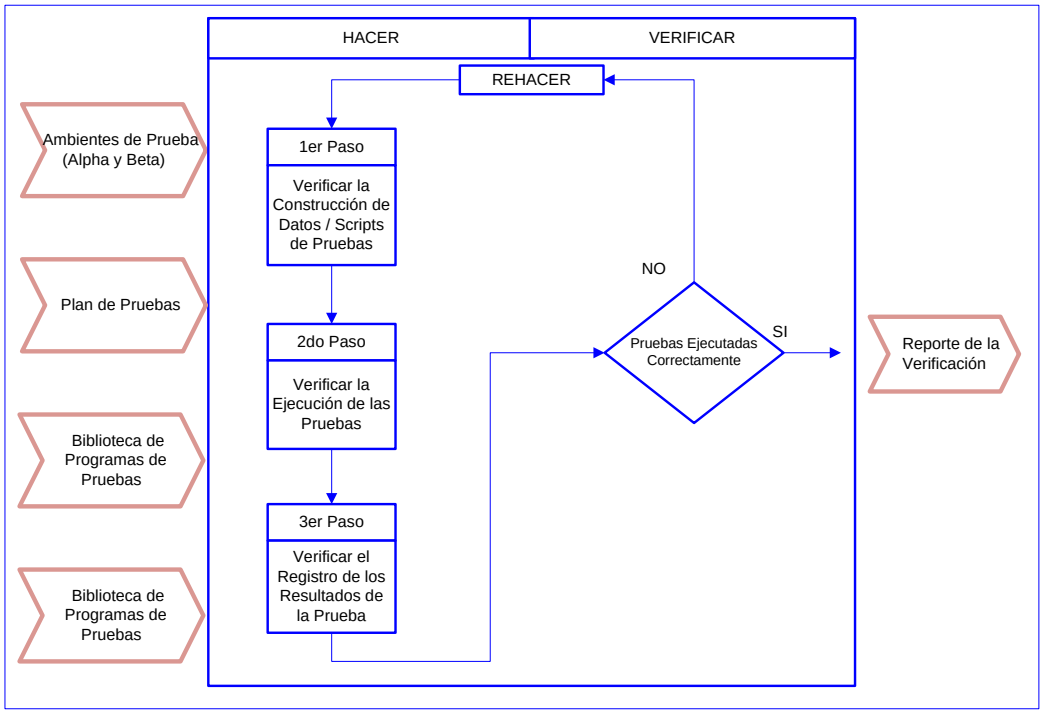


Figura. 3.5.7. Proceso del módulo de verificación

3.5.7.4 Verificar la Construcción de Datos / "Scripts" de Prueba

El concepto de construcción de datos de prueba es, simplemente, la creación de unproceso representativo de la realidad usando transacciones ficticias. De dichos datos setomará una muestra, representativa, para llevar a cabo una verificación de los mismos. Lacorrecta selección de dicha muestra es la clave para el éxito de esta tarea. Dentro de estatarea debe abarcarse la verificación de los siguientes:

Diseño de archivos de prueba.

Estos archivos deben involucrar transacciones normales, transacciones con datosinválidos, y transacciones que puedan hacer incurrir al sistema en alguna situación deexcepción.

Ingreso de los datos de prueba.

Después que las transacciones de prueba estén definidas, se deben introducir en elsistema todos los datos necesarios para que las mismas puedan ser procesadas.

Análisis de los resultados esperados.

Antes de procesar alguna transacción, el equipo de proyecto debe establecer el resultadoesperado, para cada transacción, a probar para poder compararla con el obtenido, luego derealizada la misma.

Prueba de transacciones

Para la prueba mencionada se recomiendan los siguientes 9 pasos :

- Identificar los recursos de prueba.
- Identificar las condiciones de prueba.
- Priorizar las condiciones.
- Seleccionar las condiciones.
- Determinar los resultados esperados.
- Crear transacciones de prueba.
- Documentar las condiciones de prueba.
- Ejecutar la prueba
- Verificar y corregir.

Prueba de volumen

El objetivo de dicha prueba es verificar que el sistema pueda realizar, adecuadamente, lastransacciones cuando sus programas internos o sus límites se vean excedidos. Estaspruebas suelen involucrar grandes volúmenes de datos. Los pasos que se recomiendanpara esta prueba son:

- Identificar las entradas del sistema.
- Identificar las salidas del sistema.
- Llevar cada dato a su máxima expresión tratando de sobrepasar sus límites.
- Documentar las limitaciones.
- Realizar la prueba de volumen.

Creación de casos de prueba

- Determinar el nivel de la prueba: Unitaria, concurrencia, integración, regresión o “stress”.
- Desarrollar los casos de prueba: Un caso de prueba, son todos los pasos que debenseguirse en el sistema para probar dicho caso.
- Ejecutar los casos de prueba: Usar el sistema con los casos desarrollados en el puntoanterior.
- Analizar los resultados.
- Mantener los casos de prueba.

3.5.7.5 Verificar la Ejecución de la Prueba.

Para lograr una prueba efectiva, se debe usar un plan de pruebas creado previamente.Esta etapa es la culminación de un trabajo previo preparado para esta fase. Si esto noocurre, la prueba podría resultar poco efectiva y antieconómica.

Durante esta fase, el equipo de aseguramiento de la calidad, llevará a cabo tanto laverificación de los planes de prueba, como la verificación de la ejecución de los mismos. Todo esto se realizará tomando muestras, representativas, de los planes y sus respectivasejecuciones.

Se verificará que los siguientes tipos de prueba hayan sido llevados a cabo por elequipo de proyecto, durante esta fase:

Prueba manual, de regresión y funcional

La prueba manual asegura que las personas que interactúen con el sistema podránrealizar las operaciones propuestas; la prueba de regresión es la verificación de que lo quese está instalando no afecta a lo que ya estaba instalado; y la prueba funcional apunta a quelos requerimientos del sistema puedan ser ejecutados correctamente, en diversascircunstancias.

Prueba de autorización

En esta prueba deben verificarse las reglas de autorizaciones (permisos) para ejecutar lasdistintas transacciones de la aplicación.

Prueba de integridad

Debe ser verificada durante la prueba, los controles sobre la integridad de los archivos dedatos.

Prueba de pistas de auditoria

La función de pistas de auditoría debe ser probada para asegurarse que para cualquiertransacción, pueda realizársele un seguimiento desde su inicio hasta su fin, para lograr unacorrecta reconstrucción de la misma cuando sea necesario.

Prueba de recuperación

Si el procesamiento debe continuar aunque el sistema no esté operativo, puede llegar atener que provocarse fallas intencionales en el sistema, para poder realizarse la prueba decontinuidad, la cual implica lo antedicho.

Prueba de “stress”

Es la prueba que asegura y cuantifica el nivel de servicio del sistema ante grandesvolúmenes de datos.

Prueba de seguridad

En esta prueba lo que se intenta es violar todos los aspectos de seguridad del sistemaqueriendo obtener como resultado que ninguno de estos falle.

Prueba acorde a la metodología

Todo tipo de prueba deberá ser llevado a cabo de acuerdo a las políticas, estándares yprocedimientos establecidos en la metodología de la organización. Esta última deberíaespecificar el plan de prueba requerido para cada tipo de prueba, las técnicas yherramientas de prueba recomendadas, así como el tipo de documentación requerida a lolargo de la prueba misma. La metodología debería especificar el modo de determinarcuándo una prueba fue exitosa o no.

Prueba funcional

Esta prueba verifica el correcto cumplimiento de los requerimientos del usuario.

Prueba de operación

El éxito de esta etapa dependerá fuertemente de la habilidad de las personas que utilicenel sistema, además es importante que esta prueba se haga en un ambiente lo más parecidoposible al de producción.

Inspecciones

En esta prueba lo que se hace es evaluar el código del sistema para verificar ciertas reglasque lo harán fácil de mantener ante supuestas modificaciones. Dichas reglas deberán estarexplicadas, dentro de los estándares de la organización.

Prueba de desastre

En esta prueba lo que se hace es provocar problemas en el ambiente real de la aplicación,ya que es la única manera de ver realmente cómo se comporta el sistema ante situacionesinesperadas.

Prueba funcional y de regresión

Esta fase de la prueba debe verificar la comunicación con los sistemas relacionados.Donde, la prueba funcional verifica la interconexión, de la nueva funcionalidad, y la prueba deregresión, verifica la continuidad del funcionamiento del resto de los subsistemas asociadospreexistentes.

Prueba de performance

Los criterios de performance son establecidos durante la fase de requerimientos e intentanser medidos en la fase de verificación. De no poder realizarse, porque el ambiente necesariopara llevarla a cabo sería muy costoso, se dejará para ser medido en producción.

Prueba de facilidad de operación

Esta prueba generalmente es realizada por el equipo de Operaciones,el cual trataránormalmente con el sistema e intenta que el personal del grupo de proyecto no proveaningún tipo de instrucciones a los usuarios, para que pueda reproducirse el normaldesarrollo de las transacciones, tal como sería en el ambiente de producción.

3.5.7.6 Verificar el Registro de los Resultados de la Prueba

Ya que documentar un problema, es el primer paso para la corrección del mismo, esnecesario llevar adelante una verificación del registro de los resultados de la prueba. Esto sellevará a cabo, tomando muestras representativas de tales resultados, para su análisisposterior.

Se verificará que estén descriptos al menos, los siguientes aspectos, para todos losproblemas encontrados en las pruebas:

- Declaración del problema: Establecer cuál es el problema.
- Criterio: Establecer cómo debería comportarse el sistema.
- Efecto: Es la diferencia entre el problema y el comportamiento esperado.
- Causa: ¿Qué puede haber causado el problema?.

Para lograr una buena documentación de los problemas, es aconsejable seguir estos trespasos:

- Documentar el desvío: Esencialmente el usuario compara “lo que es” con “lo que deberíaser”, y cuando es identificada una desviación en este sentido, corresponde comenzar adesarrollar la declaración del problema.

- Documentar el (los) efecto(s): Debe evaluarse lo que el problema significa para elsistema. La atención que se le dará al mismo, estará directamente relacionada con lodeclarado en el/ los efecto/s.
- Documentar la o las causas: La o las causas son la o las razones subyacentes para la condición. En algunos casos la causa puede ser obvia a través de los hechos. Pero enotros, puede ser necesaria una investigación para encontrarla.

3.5.7.7 Salidas

- Muestra de planes de prueba verificados.
- Muestra de transacciones de prueba verificadas.
- Muestra de resultados de la ejecución de dichas transacciones verificados.
- Desvío de los resultados esperados documentados.

3.5.7.8 Lista de Verificación (Checklist)

En la tabla 3.5.7 se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Todos los pasos fueron realizados como se especificaron?				
2	Si los pasos no fueron realizados como se especificaron. ¿Se afectaron recursos adicionales para resolver defectos durante la etapa de ejecución?				
3	¿Se estableció un ambiente de prueba apropiado para realizar la prueba del software?				
4	¿Se entrenó analistas de calidad en las herramientas de verificación que serán utilizadas durante esta etapa?				
5	¿Se fijó un tiempo adecuado para esta etapa?				
6	¿Se fijaron los recursos adecuados para esta etapa?				
7	¿Los métodos para crear datos de prueba son apropiados para el sistema?				
8	¿Fueron creados los datos de prueba necesarios para probar adecuadamente el software?				

9	¿Fueron programadas todas las técnicas de verificación indicadas en el plan de prueba para ser ejecutadas durante este paso? (Ej. Prueba de regresión)				
10	¿Fueron determinados los resultadosesperados de las pruebas?				
11	¿Se ha establecido el proceso por el cual se determina el desvío entre los resultados esperados y los actuales?				
12	¿Se han documentado los resultados esperados y los actuales cuando existeuna diferencia entre ellos?				
13	¿Se determinó el potencial impacto de una desviación? (Ej. Problema en la verificación)				
14	¿Se ha establecido un procedimiento para asegurar las acciones / resoluciónapropiada de los defectos?				

Tabla. 3.5.7 Lista de verificación del proceso de verificación.

3.5.7.9 Métricas

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados hasta verificación}}{\text{Cantidad total de defectos hasta verificación}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos hasta verificación}}{\text{Cantidad total de defectos hasta verificación}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

- Antigüedad media de defectos pendientes:

Total de antigüedad de defectos pendientes al fin de un período
Cantidad total de defectos pendientes al fin de un período

3.5.7.10 Involucrados

- Gerente de Proyecto.
- Representante del grupo de análisis de requerimientos.
- Representante del grupo de análisis funcional.
- Representante del grupo de diseño.
- Representante del grupo de analistas desarrolladores.
- Grupo de testeo.
- Líder o representante de la Calidad del Software

3.5.8 VALIDACION DEL SISTEMA

3.5.8.1 Objetivo

Describir los procedimientos para identificar los criterios de aceptación para el ciclo de vidade los productos y validarlos. La aceptación final no solo tiene en cuenta que elproducto de software resuelva adecuadamente los requerimientos de los usuarios, sinotambién, reconocer que el proceso de desarrollo fue adecuado. Como parte del proceso deciclo de vida, la aceptación del software permite:

- Detección temprana de los problemas del software.
- Preparación apropiada de los medios para realizar la prueba.
- Consideración temprana de las necesidades del usuario durante el desarrollo delsoftware.

3.5.8.2 Entrada

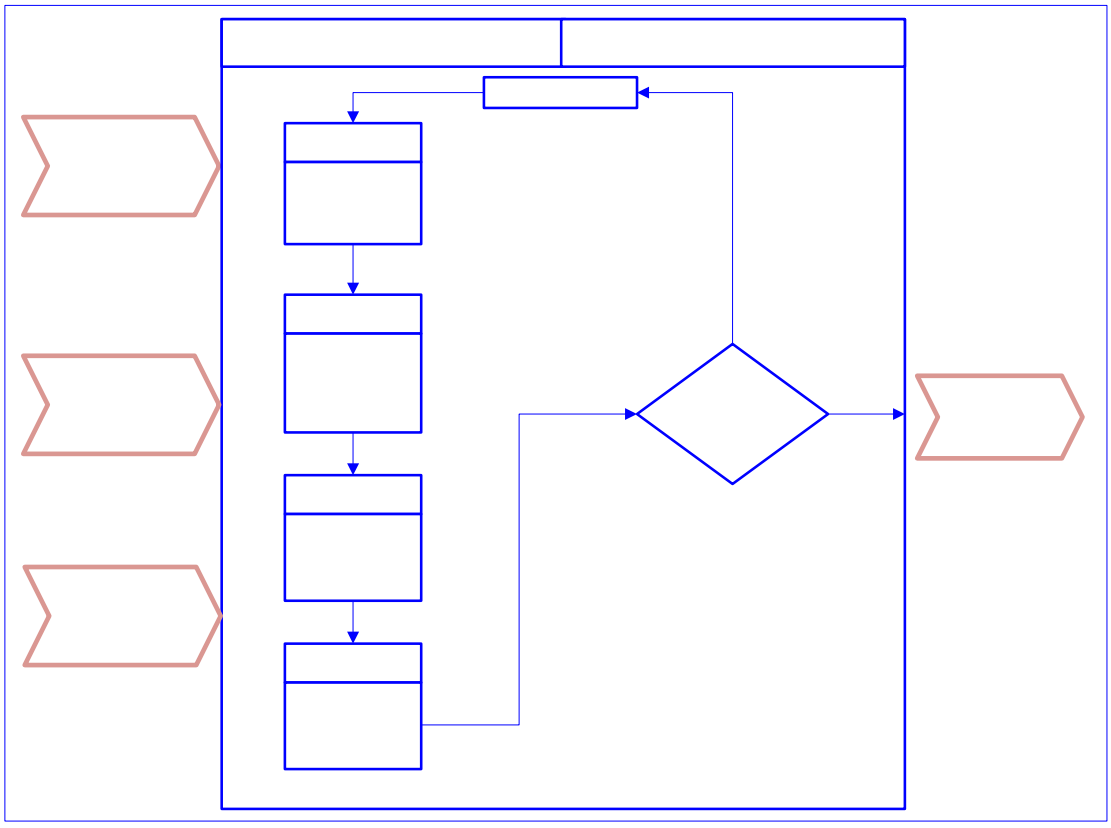
Las entradas dependen del alcance asignado a la validación del sistema (prueba deaceptación). Estas suelen ser:

- Software probado: Una vez realizada la verificación de las piezas de software secomienza con la prueba de aceptación.
- Plan de Prueba de Aceptación:
    - Pruebas en Línea (“On-Line”).
    - Pruebas por Lotes (“Batch”).
    - Pruebas de Interfaces.
    - Pruebas de Integración.
    - Pruebas de “stress”.
    - Pruebas de conectividad.
    - Pruebas de servidores.

- Lista de defectos no resueltos: Quizás no sea prudente esperar hasta que todos losdefectos reportados sean corregidos antes de empezar las pruebas de aceptación. Eneste caso, los encargados de efectuar las pruebas de aceptación reciben una lista noresuelta de defectos, la cual les permitirá conocer los procesos incorrectos y así enfocarla atención en los criterios principales de aceptación antes de la prueba

3.5.8.3 Proceso

La figura 3.5.8detalla gráficamente, el proceso a desarrollar durante este módulo:



HACER

Figura. 3.5.8. Proceso del módulo de Validación

Planes de Prueba de Aceptación

1er Paso

Verificar los Criterios de Aceptación

3.5.8.4 Verificar los Criterios de Aceptación Definidos

El usuario deberá definir el o los criterios que el software deberá cumplir para ser aceptado.Los requerimientos de aceptación, que el sistema debe cumplir, pueden ser divididos en lassiguientes cuatro categorías:

- Requerimientos Funcionales: Referido a las reglas de negocio que el sistema debeejecutar.
- Requerimientos de Performance: Referido a los tiempos y recursos de operación.
- Requerimientos de Interfaz: Referido a la relación humano-máquina, máquina-módulo.
- Requerimientos globales de Calidad del Software: Referidos a la obtención de los límitespara los factores o los atributos tales como la fiabilidad, comprobación, exactitud, yusabilidad.

Estos criterios deberían estar enunciados y documentados durante la fase derequerimientos, para ser refinados y detallados en las fases siguientes. Se procederá asolicitar la recopilación de estos requerimientos, luego el equipo de calidad, procederá a verificarlos.

Se debería determinar con el usuario, el grado de criticidad de los requerimientos para los ítems, cuyas categorías se presentan en la tabla 3.5.8.1:

CATEGORIA	ITEM'S
Funcionalidad	Consistencia interna de documentos, código, y entre las fases Trazabilidad (posibilidad de seguimiento) de la funcionalidad Verificación adecuada de la lógica Preservación de funcionalidad en el ambiente productivo
Performance	Análisis de factibilidad de requerimientos de performance. Herramientas de simulación. Análisis de desempeño en el ambiente productivo.

Interfaz	Documentación de la interfaz. Complejidad de la interfaz. Planes de prueba e integración de la interfaz. Ergonomía de la interfaz. Prueba de la interfaz en el ambiente productivo.
Calidad global del Software	Cuantificación de medidas de calidad. Criterios para la aceptación de los productos de software. Suficiencia de documentación y normas de desarrollo de sistemas. Criterios de Calidad para la Prueba Operacional.

Tabla 3.5.8.1Ítems de requerimientos para determinar su criticidad

3.5.8.5 Verificar el Desarrollo del Plan de Aceptación

El primer paso para completar la aceptación del software es el desarrollo simultáneo de unplan de aceptación, un plan de proyecto y de los requerimientos del software que aseguranque las necesidades del usuario están representadas correcta y completamente. Estedesarrollo simultáneo proveerá una visión general que asegurará que todo los recursosnecesarios, estarán incluidos en el plan de proyecto.

La tabla 3.5.8.2detallan los puntos a verificar dentro de un plan de aceptación típico:

CATEGORIA	ITEM'S
Descripción del proyecto	Tipo de sistema. Ciclo de vida definido por la metodología de la organización. Usuarios del sistema. Principales tareas que debe satisfacer el sistema. Principales interfaces externas del sistema. Uso normal esperado. Potenciales usos indebidos. Riesgos. Restricciones. Estándares y Normas.
Responsabilidades del usuario	Organización y responsabilidades para las actividades deaceptación del sistema. Recursos y cronograma de requerimientos. Requerimientos del recurso.

	Requerimientos para soporte automatizado, datosespeciales, y entrenamiento. Normas, estándares y convenciones. Actualización y revisión de los planes de aceptación.
Procedimientos administrativos	Informes de anomalías. Control de cambios. Resguardo de información.
Descripción de aceptación	Objetivos para el proyecto. Resumen de criterios de aceptación. Principales actividades de aceptación y revisiones. Requerimientos de información. Tipos de decisiones de aceptación. Responsabilidad de las decisiones de aceptación.

Tabla 3.5.8.2 Puntos a verificar dentro de un plan de aceptación típico

3.5.8.6 Verificar la Ejecución del Plan de Aceptación

El objetivo de este paso es verificar el cumplimiento de los criterios de aceptación en elproducto entregado. Esto se puede llevar a cabo a través de revisiones que involucran,verificar, productos provisorios y entregables desarrollados parcialmente, a lo largo delproceso de desarrollo.

El criterio de aceptación del sistema debería estar especificado formalmente en el plan delproyecto. El plan identifica los productos a ser verificados, el criterio específico deaprobación/rechazo, las revisiones, y los tipos de verificación que se harán durante el ciclode vida completo.

Las decisiones de aceptación necesitan una estructura sobre la cual operar, algunoscomponentes son: contratos, criterios de aceptación y mecanismos formales. La aceptacióndel software deberá referirse a un criterio específico que los productos deben tener para seraceptados. El principal medio para alcanzar la aceptación en el desarrollo de los sistemasde software, es mantener una revisión periódica de la documentación y productos desoftware.

Cuando la decisión de aceptación requiere cambios, se hace necesaria otra revisión paraasegurarse de que los cambios solicitados hayan sido configurados e implementados enforma correcta, y que el efecto en cualquier otra parte sea aceptable.

Las actividades de aceptación, pueden incluir la verificación de las piezas de software. Laverificación de aceptación del software, se vuelve formal, cuando durante el ciclo de vida deldesarrollo, el usuario acepta o rechaza el mismo. Esto es como un requerimiento contractualentre el usuario y el equipo del proyecto. El rechazo, generalmente, significa que se deberárealizar un trabajo adicional en el sistema a fin de hacerlo aceptable para el usuario. Laprueba de aceptación final, es la última oportunidad que tiene el usuario para examinar lafuncionalidad, interfaz, performance y aspectos de calidad antes de la revisión final deaceptación. En ese momento, el sistema deberá incluir el software a entregar, toda ladocumentación de usuario y las versiones finales de otros entregables del sistema.

3.5.8.7 Revisar la Decisión de Aceptación

La aceptación de un sistema generalmente significa que el proyecto se ha completado, conla excepción de alguna contingencia.

Las decisiones de aceptación pueden involucrar alguna de las siguientes situaciones:

- Los cambios solicitados son aceptados antes de pasar a desarrollar la próxima actividad.
- Algunos cambios, deben ser aceptados y llevados a cabo antes de continuar con eldesarrollo del producto, y otros, pueden ser postergados hasta la próxima revisión delproducto.
- El desarrollo puede continuar y los cambios podrán ser aceptados en la próxima revisión.
- No hay cambios solicitados y el desarrollo puede continuar.

Cada una de estas situaciones, será verificada por el equipo de calidad de software, teniendo en cuenta los defectos no resueltos, el cronograma del proyecto, el estadoactual del mismo, los recursos involucrados, etc.

3.5.8.8 Salidas

- Informe de la decisión de Aceptación.

3.5.8.9 Lista de Verificación (Checklist)

La tabla 3.5.8.3 se muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Se incorporó la prueba de aceptación al plan general de pruebas?				
2	¿La prueba de aceptación estáenfocada como un proceso del proyecto en lugar de un paso aislado al final del testing?				
3	¿Fueron seleccionados los usuarios correctos para determinar los criterios de aceptación para el software y/o componentes de hardware?				
4	¿El grupo que define los criterios de aceptación, es representativo de los usos del sistema a ser probado?				
5	¿Esos individuos aceptan la responsabilidad de identificar los criterios de aceptación?				
6	¿Los criterios de aceptación fueron identificados tempranamente como para lograr influir en el planteo e implementación de la solución?				
7	¿Se ha desarrollado y escrito el plan de aceptación?				
8	¿El plan de aceptación es consistente con los criterios de aceptación?				
9	¿Se están verificando los productos intermedios por parte de los testeadores, antes de ser utilizados para la siguiente tarea de implementación?				
10	¿Se han seleccionado las técnicas de prueba apropiadas para el test de aceptación?				
11	¿Los testeadores tienen el nivel de conocimiento necesario para realizar dicha tarea?				
12	¿Se afectaron los recursos necesarios para la realización de la prueba de aceptación?				

13	¿Se destinó el tiempo necesario para la realización de la prueba de aceptación?				
14	¿Se han publicado las opiniones internas de la prueba de aceptación?				
15	¿Fue buena la reacción del equipo del proyecto en lo que concierne a los testeadores en la prueba de aceptación?				
16	¿Se ha tomado la decisión final de aceptación del sistema?				
17	¿Se han identificado los criterios de aceptación críticos?				
18	¿Los requerimientos fueron escritos con el detalle suficiente, de forma tal de poder obtener a partir de ellos las interfaces?				
19	¿Están los responsables de cada interfaz, descritos en el diagrama?				
20	¿Se ha definido un caso de prueba para cada uno de los límites que posee el sistema?				
21	¿Los usuarios del sistema están de acuerdo con los casos definidos? ¿Losconsideran completos?				
22	Se definieron tanto las condiciones normales como las fallidas a probar?				
23	¿Los usuarios están de acuerdo con que los casos de prueba cubren todos los probables escenarios?				

Tabla. 3.5.7 Lista de verificación del proceso de verificación

3.5.8.10 Métricas

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados hasta validación}}{\text{Cantidad total de defectos hasta validación}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos hasta validación}}{\text{Cantidad total de defectos hasta validación}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

- Estabilidad de las criterios de aceptación:

$$\frac{\text{Cantidad de criterios iníciales}}{\text{Cantidad total de criterios}}$$

- Estabilidad de la aceptación:

$$\frac{\text{Cantidad de cambios durante la aceptación}}{\text{Cantidad total de cambios}}$$

3.5.8.11 Involucrados

- Gerente de Proyecto.
- Representante del grupo de análisis de requerimientos.
- Representante del grupo de análisis funcional.
- Representante del grupo de diseño.
- Representante del grupo de analistas desarrolladores.
- Grupo de testeo.
- Usuario Final
- Líder o representante de la Calidad del Software

➢

3.5.9 INSTALACION

3.5.9.1 Objetivo

Realizar una verificación completa de la fase de instalación para nuevos sistemas y/ocambio de versiones. La verificación, incluye, para cada criterio de instalación identificado,una evaluación recomendada y las técnicas y herramientas sugeridas para llevarla a cabo.

3.5.9.2 Entrada

- Plan de instalación.
  - Diagrama de flujo de la Instalación.
  - Documentos y listados de instalación (sólo si es requerida una instalación especial).
  - Resultados de la verificación de instalaciones especiales.
  - Documentación de pasaje del sistema a producción.
  - Instrucciones para los administradores u/o operadores.
  - Instrucciones y procedimientos para los nuevos usuarios.
- Resultado del proceso de instalación.

3.5.9.3 Proceso

La figura 3.5.9detalla gráficamente, el proceso a desarrollar durante este módulo:

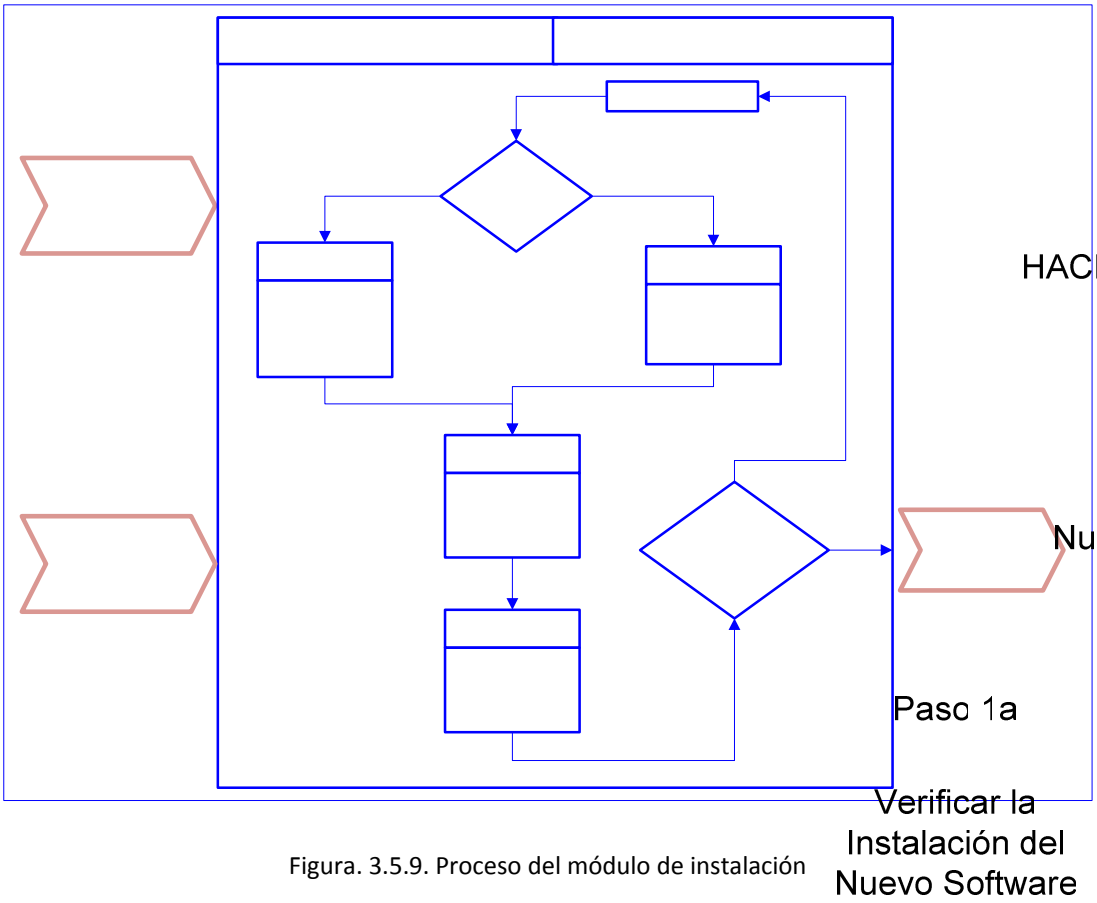


Figura. 3.5.9. Proceso del módulo de instalación

3.5.9.4 Verificación de la Instalación de Nuevo Software

La fase de instalación posee dos dificultades para el equipo de calidad de software. La primera es que la instalación es un proceso separado del resto del desarrollo dela aplicación. Estas funciones no están relacionadas con las necesidades del usuario, perosí con las de instalar en producción una aplicación completa y verificada. La segunda, es que la instalación ocurre en un lapso de tiempo muy corto. Es común que lainstalación dure unas horas. Por lo tanto la verificación debe ser bien planeada y ejecutada.

Es importante que los resultados de la verificación estén disponibles antes de la instalacióncompleta del sistema. El objetivo de la verificación es determinar si la instalación fue exitosa,por lo tanto los resultados deben estar disponibles lo antes posible. Esto significa que losresultados a obtener de la verificación deben estar

explicitados antes de que la verificacióncomience. Los 15 factores a tener en cuenta en la verificación de la instalación son:

- Integridad y exactitud de la instalación.
- Prohibición de modificación de datos durante la instalación.
- Integridad de archivos de producción.
- Registro de pistas de auditoría de instalación.
- Integridad del sistema/versión anterior asegurado (continuidad de procesamiento).
- Recuperación ante fallas de la instalación.
- Control de acceso durante la instalación.
- Instalación acorde a la metodología.
- Instalación en producción de los programas indicados en la fecha asignada.
- Puesta a disposición de instrucciones de instalación.
- Documentación completa (Mantenibilidad).
- Documentación completa (Portabilidad).
- Interfaces.
- Monitoreo de la performance.
- Procedimientos operativos.

3.5.9.5 Verificación de la Instalación de Cambios de Software.

El objetivo principal es instalar el cambio requerido en el tiempo adecuado. A continuaciónse detallan los objetivos específicos de la instalación de cambios:

- Poner a la aplicación modificada en producción.
- Evaluar la eficiencia de los cambios.
- Monitorear la exactitud de los cambios.
- Mantener actualizadas con las últimas versiones disponibles, los ambientes de prueba yproducción.

Son tres las sub-tareas a verificar en la instalación de cambios de software.

a) Verificar si es adecuado el plan de recuperación ante fallas.

Existen varios aspectos de los cambios que impactan en el proceso de recuperación antefallas:

- Agregado de una nueva función.
- Cambio en las funcionalidades
- Uso adicional de programas utilitarios.
- Cambios en períodos de resguardo.
- Cambios en los programas.
- Introducción de un formulario nuevo o revisado.

Los analistas de calidad deben evaluar el impacto en el plan de recuperación, cambio porcambio. Si determinan que el cambio afecta a la recuperación, se debe actualizar el plan.

b) Verificar que el cambio correcto, fue puesto en producción.

El pasaje de la modificación, desde del ambiente de prueba a producción lo debe hacer eldueño del software, previa aprobación del usuario.

El ambiente de producción debería ser capaz de controlar los programas de acuerdo con lafecha de puesta en producción. Cada versión del programa en producción debería seridentificada (etiquetada) de acuerdo a si entra o sale de producción. Debe estar disponible,para cada programa, un historial de los cambios, y así poder proveer pistas de auditoría.

Para verificar que se introdujo el cambio correcto en producción se verificará lo siguiente:

- Que los cambios en los programas hayan sido documentados: El objetivo es contar conun historial, para proveer pistas de auditoría que indiquen cuándo se ha realizado uncambio y por otro impedir cambios no autorizados.

- Que se cuente con un procedimiento para pasajes de versiones del ambiente de pruebaal ambiente de producción.
- El dueño del software decide cuándo una versión va a ser pasada a producción. Estaaprobación da la orden a administración u operaciones para dar aviso que el cambio va a ser instalado.
- Que el procedimiento citado en el punto anterior se cumpla

c) Verificar que las versiones innecesarias hayan sido eliminadas.

Los programas con versiones innecesarias (viejas) deben ser borrados. Esto solo puedehacerse con el debido nivel de autorización. Se debe elaborar un formulario que autorice laeliminación de las versiones antiguas. Este formulario lo debe completar el Gerente o Jefe delproyecto de mantenimiento de software y enviarlo al departamento de administración u operaciones.

El departamento de Administración u Operaciones debe tener un proceso para eliminar versionesinnecesarias, después de recibir la debida autorización.

3.5.9.6 Seguimiento en Producción.

Los sistemas aplicativos son propensos a tener problemas inmediatamente después de introducir una nueva versión[32]. Por eso es necesario monitorear las salidas del aplicativo, unavez instalada la nueva versión en producción.

El personal asignado al mantenimiento de software y el usuario deberán proveer pistas,acerca de dónde ellos creen que podrían ocurrir problemas. El tipo de indicios incluye:

- Investigación de transacciones.
- Clientes.
- Reportes.

[32]Véase la figura 2.3 Curva de Fallos en Capitulo II para mayor detalle de este concepto.

- Archivos.
- Performance.

Estos deben ser documentados mediante el uso de un formulario.

3.5.9.7 Documentar los Problemas.

Los problemas detectados durante el monitoreo de los cambios, deben ser documentados.Además se debe evaluar el riesgo asociado a ese problema.

El reporte de un problema, causado por un cambio en el sistema, permite asociar talproblema a lo específico del cambio. Este reporte debe ser enviado al analista demantenimiento de software para su análisis y posterior corrección.

3.5.9.8 Salidas

- Reporte de recuperación ante fallas.
- Reporte de historia de cambios al software.
- Reporte de puestas en producción.
- Reporte de autorización de eliminación de versiones innecesarias.
- Reporte de notificación de monitoreo de cambios de software.
- Reporte de problemas causados por cambios de Software.

3.5.9.9 Lista de Verificación (Checklist)

La tabla 3.5.9muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	
1	¿Cada cambio es revisado para cada impacto sobre el plan de reinicio/recuperación?				

2	¿Si el cambio impacta a la recuperación, es calculado nuevamente el tiempo de no servicio?				
3	¿Si el cambio impacta a la recuperación, es estimado nuevamente el riesgo del tiempo de no servicio?				
4	¿Los cambios necesarios para el proceso de recuperación son documentados?				
5	¿La notificación de los cambios de la versión de producción fueron documentados?				
6	¿Los cambios de los sistemas de aplicación son controlados por una aplicación de control de cambio de numeración?				
7	¿Existen procedimientos para eliminar versiones antiguas de las bibliotecas de objetos?				
8	¿Existen solicitudes de eliminación para que producción sea autorizado para eliminar programas?				
9	¿Están establecidos procedimientos para asegurar que la versión de los programas sea pasada al ambiente de producción en la fecha correcta?				
10	Si esto afecta a procedimientos de operación, ¿Están notificados los operadores del día en que la nueva versión se pase a producción?				
11	¿Están establecidos los procedimientos para monitorear los cambios de los sistemas de aplicación?				
12	¿Las personas que monitorean reciben notificación de que el sistema de aplicación fue cambiado?				
13	¿Las personas que monitorean los cambios reciben indicios de las áreas consideradas impactadas por el probable problema?				
14	¿Las personas que monitorean el cambio del sistema de aplicación reciben una guía de que acciones tomar si el problema ocurre?				
15	¿Los problemas detectados, inmediatamente después de que el cambio del sistema ocurrió, sondocumentados en un formulario especial para poder vincularlos a un cambio en particular?				
16	¿Se solicita, a las personas que documentan problemas, que documenten el impacto a nivel organizacional que puede implicar el problema?				
17	¿La información acerca de la instalación de cambios de software es recopilada y documentada?				
18	¿El servicio de la dirección realiza revisiones y utiliza la retroalimentación de los datos?				
19	¿El servicio de la dirección periódicamente revisa la efectividad de la instalación de cambios de software?				

Tabla. 3.5.9Lista de verificación del módulo de instalación

3.5.9.10 Métricas

- Eficiencia en la detección de defectos:

$$\frac{\text{Cantidad de defectos detectados hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

- Eficiencia en la remoción de defectos:

$$\frac{\text{Cantidad de defectos removidos hasta instalación}}{\text{Cantidad total de defectos hasta instalación}}$$

- Tasa de efectividad de la verificación:

$$\frac{\text{Cantidad de ítems cubiertos}}{\text{Cantidad total de ítems}}$$

3.5.9.11 Involucrados

- Gerente de Proyecto.
- Representante del grupo de mantenimiento de software
- Representante del grupo de instalación de software
- Representante del grupo administración u operaciones
- Líder o representante de la Calidad del Software

3.5.10 METODOLOGIA, ESTANDARES & PROCEDIMIENTOS

3.5.10.1 Objetivo

La verificación de productos deberá ajustarse a la organización. Esto incluye ajustar elvocabulario y los términos utilizados, en la verificación, para que sean los mismos que losutilizados en la organización, para describir los componentes del sistema, documentos yproductos.

3.5.10.2 Entrada

- Metodología de desarrollo de la organización.
- Estándares de la organización.
- Procesos y Procedimientos de la organización.
- Productos que ingresan a cada una de las fases definidas en el modelo.

3.5.10.3 Proceso

La figura 3.5.10detalla gráficamente, el proceso a desarrollar durante este módulo:

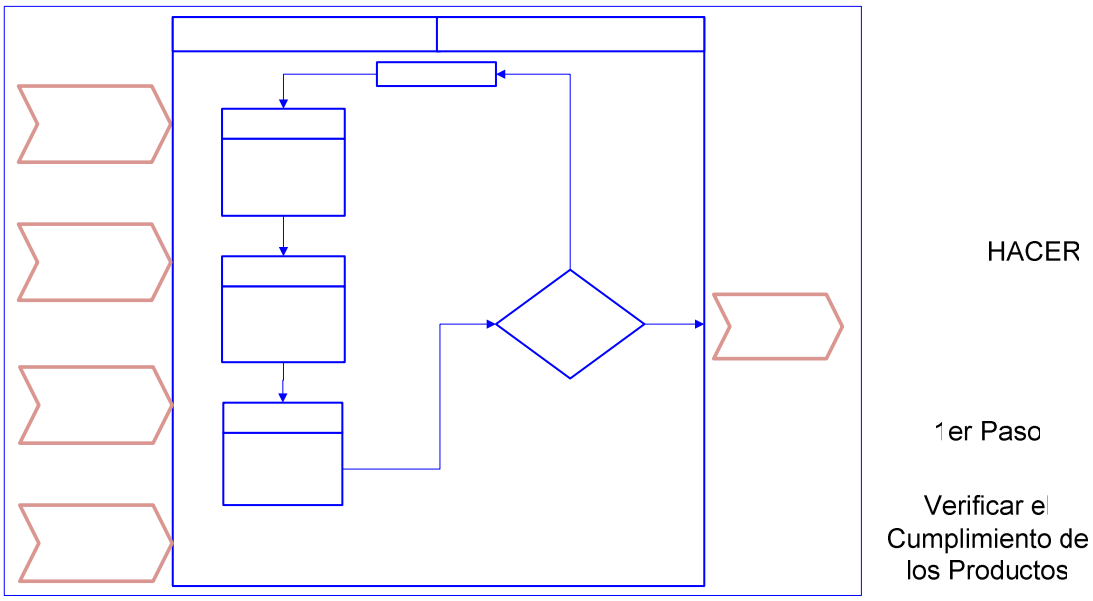


Figura. 3.5.10. Proceso del módulo de metodología, estándares y procedimientos.

3.5.10.4 Verificar el Cumplimiento de los Estándares de Documentación

La formalidad, extensión y nivel de detalle de la documentación a preparar dependerán delas prácticas de la organización y del tamaño, complejidad y riesgo del proyecto. Lo que esadecuado para un proyecto puede ser inadecuado para otro. Para verificar ladocumentación, se procederá a comprobar si está completa, es adecuada y cumple con lasnormas y estándares corporativos.

Esta tarea intenta evaluar el apego a los procedimientos y estándares definidos por lametodología de la organización, evaluando los criterios así establecidos y determinando si elnivel y el alcance de la documentación cumplen con lo requerido.

En el caso que la metodología de la organización, establezca pautadamente ladocumentación necesaria para cada tipo de proyecto según su criticidad, se procederá averificar si la documentación cumple con cada una de esas pautas.

3.5.10.5 Verificar la Integridad de la Documentación

Para evaluar la integridad de la documentación, se usarán los criterios definidos alrespecto, en la metodología de la organización. El equipo de aseguramiento de la

calidad desoftware deberá determinar si cada documento cumple con esos criterios. Si ladocumentación no alcanza a cubrir un criterio, deberá identificarse la deficiencia ydocumentarse.

Otros criterios adicionales, que pueden ser utilizados para evaluar la integridad de cadadocumento, son discutidos en el capítulo de Técnicas y Herramientas, nombrándose acontinuación cada uno de ellos:

- Contenido de la documentación.
- Usuarios del documento.
- Redundancia.
- Flexibilidad.
- Tamaño del documento.
- Combinación de diferentes tipos de documentos.
- Formato.
- Secuencia de contenido.
- Títulos de secciones/párrafos.
- Expansión de los párrafos.
- Diagramas de flujo y tablas de decisión.
- Formularios.

Es aconsejable llevar a cabo una o más verificaciones de adecuación de la documentacióna los estándares de la organización. Tales verificaciones requieren que una persona capaz,no asociada al proyecto, haga una simulación de cambio al sistema basado en ladocumentación actual y el requerimiento de cambio (no deben cambiarse ni ladocumentación real ni los programas). Después que la persona haga el cambio, alguienfamiliarizado con el proyecto debe evaluar si se ha hecho correctamente. Esta verificaciónrevelará si:

- La documentación es entendible para las personas ajenas al proyecto

- Una persona ajena puede utilizar la documentación para hacer un cambiocorrectamente, y lo puede hacer de manera eficiente y efectiva.

3.5.10.6 Verificar el grado de actualización de la Documentación

La documentación que no esté actualizada no será útil. Si la metodología de laorganización ya establece algún método para verificar el grado de actualización de ladocumentación, dicho método será usado por el grupo de verificación durante la misma.Caso contrario, se propone que el equipo de verificación de la documentación utilicecualesquiera de las siguientes cuatro técnicas de verificación (que se detallarán en el anexo A “Técnicas y Herramientas”) para validar la actualización de ladocumentación:

- Utilizar la documentación existente para llevar a cabo un cambio en la aplicación
- Comparar el código fuente de los programas con la documentación
- Verificar que la documentación esté vigente
- Verificar la actualización de los documentos con el usuario final

Las anteriores técnicas podrán aplicarse sobre los documentos completos o sobre partesde los mismos o sobre una muestra de la documentación del proyecto.

3.5.10.7 Salidas

- Reporte de documentación verificada

3.5.10.8 Lista de Verificación (Checklist)

La tabla 3.5.10muestra la lista de verificación o checklist, que sirve para revisar los aspectos más importantes de este proceso.

NUM.	ITEM	RESPUESTA			COMENTARIOS
		SI	NO	N/A	

1	¿Existen estándares para ladocumentación del sistema?				
2	¿Los miembros del equipo de prueba están informados sobre las intenciones y contenidos de esos estándares?				
3	¿Los estándares son adaptables a sistemas de varios tamaños, de manera que el tamaño del proyecto no sea directamente proporcional con al tamaño del documento?				
4	¿Se les entregó al equipo decalidad de software una copia completa de la documentación del sistema correspondiente a esta verificación?				
5	¿El equipo de calidad de software ha medido las necesidades de la documentación de acuerdo a los criterios establecidos por la metodología de laorganización?				
6	¿El equipo de calidad de software ha determinado qué documentos deben revisarse?				
7	¿Las personas del proyecto están de acuerdo con las sugerencias del equipo de calidad de software en cuanto a las revisiones de la documentación?				
8	¿El equipo de calidad de software determinó la completitud de los documentos usando los criterios detallados en la metodología de la organización?				
9	¿ El equipo de calidad de software ha utilizado el proceso de Inspección para determinar la completitud de la documentación del sistema?				
10	¿ El equipo de calidad de Software ha determinado el grado de actualización de la documentación del proyecto?				
11	¿ El equipo de calidad de software ha desarrollado un documento detallando las deficiencias de la documentación respecto de los estándares de la organización?				
12	¿Se ha asegurado el equipo de calidad de software de que las deficiencias descritas están documentadas?				

Tabla. 3.5.10Lista de verificación del módulo de metodología, estándares y procedimientos

3.5.10.9 Métricas

- Documentación vigente:

Cantidad de documentación vigente no actualizada

Cantidad total de documentación vigente

- Revisión de documentos

Cantidad de documentos revisados

Cantidad de documentos a revisar

3.5.10.10 Involucrados

- Gerente de Proyecto.
- Responsables de elaboración de documentos
- Grupo de Metodología, Estándares y Procedimientos
- Líder o representante de la Calidad del Software

IV. CONCLUSION

Aunque pocos profesionales pondrían en duda la necesidad de aplicar los conceptos de calidad en el desarrollo de software,muchos no están interesados en establecer funciones de aseguramiento y control de la calidad formal dentro de una organización o de un proyecto en particular. Las principalesrazones de esta aparente contradicción son las siguientes:

- Los responsables del desarrollo se resisten a hacer frente a los costos extras inmediatosy les cuesta ver los beneficios a largo plazo.
- Por desconocimiento, muchos profesionales creen que ya están haciendo todo lo quehay que hacer con respecto al aseguramiento y control de la calidad en la organización o en los proyectos que participan.
- Pocos saben dónde situar lasfunciones de Aseguramiento y Control de Calidad dentro de la organización o proyecto.
- Todos quieren evitar cierto nivel de burocracia, que la actividad propia de un ente de calidad de software, tiende a introduciren el proceso de ingeniería del software.

En el presente trabajo, se ha expuesto un enfoque práctico para la aplicación de un modelode calidad del software en una organización o proyecto en particular, procurando sostener la hipótesisque **lacalidad del producto final dependerá de que también están afinados los procesos productivos que lo sustentan**.

Aunque basado en la metodología de mejora continua de los procesos CMMI y PMI, este modelo de calidad es independiente de lametodología de desarrollo que alguna organización utilice. Lo anterior se debe a la generalidad y amplitud con la cual este modelo se ha desarrollado, es aplicable a fases comunes de cualquier metodología

ya que no obliga a suaplicación rígida, sino que permite su adaptación a esa metodología y a la propia realidad dela organización o proyecto.

Asimismo, el enfoque presentado constituye un complemento ala metodología de desarrollo que ya utiliza una organización y no obliga a profundos cambiosen la misma. Por el contrario, si la organización o un el proyecto no utiliza una metodología, incentiva lautilización de una.

4.1 Entregables

Según lo expuesto anteriormente los principales entregables de este trabajo son:

A. Modelo de calidad de software dinámico, modular y flexible, aplicable a una organización o proyecto de desarrollo de software en particular.

B. Herramientas para el líder de proyecto y su grupo que le permita potenciar las habilidades duras (técnicas) y blandas del equipo.

C. Ejemplos de Plantillas, Listas de Validación y Verificación, Métricas, Matriz de Riesgos, Criterios de pruebas e inspecciones, Ranking de Factores de Éxito, etc.

4.2 Beneficios

Algunos de los principales beneficios esperados de la aplicación constante y rigurosa del modelo de calidad propuesto, mediante el enfoque presentado, son los siguientes:

- El software tendrá menos defectos latentes, como consecuencia, se reducirá el esfuerzoy el tiempo durante las etapas de prueba y mantenimiento.
- Se dará una mayor fiabilidad y, por tanto, una mayor satisfacción del cliente.

- Se podrán reducir los costos de mantenimiento (un porcentaje sustancial delos costostotales del software).
- El tiempo y el costo total del ciclo de vida del software disminuirá.

4.3 Problemas

Por el lado negativo diríamos queimplementar un modelo de calidad de software podría resultar problemática por las siguientesrazones:

- Es difícil institucionalizar en organizaciones pequeñas o proyectos menores, en las que no están disponibleslos recursos necesarios para llevar a cabo esas actividades.
- Representa un cambio cultural, y el cambio nunca es fácil.
- Requiere un gasto que, de otro modo, nunca se hubiera destinadoexplícitamente a laingeniería del software o al aseguramiento y control de la calidad.

4.4 Evaluación Costo - Beneficio

A la hora de establecer o implementar el modelo de calidad propuesto en unaorganización, es razonable preguntarse si valdrá la pena, si el costo de su establecimiento yaplicación continua se verá justificado por los beneficios alcanzados.

En el marco de un análisis básico de costo-beneficio, se puede afirmar que la aplicación del modelo de calidad de software en una organización o proyecto será efectiva si se cumple con la siguiente regla:

$$A > B + C$$

Siendo:

VAR	DESCRIPCION
A	Costo de las fallas o defectos que aparecen sin la aplicación del modelo de calidad propuesto. Esto incluye, entre otros, las actividades de reparación, re-trabajo, resolución de quejas del cliente, retorno y reemplazo del producto, soporte y ayuda al cliente, y el pago de multaso sanciones contractuales.
B	Costo de la propia aplicación del modelo de calidad propuesto. Esto incluye, entre otros, los salariosdel equipo aseguramiento y control de calidad y las actividades de planificación, revisiones y auditorias.
C	Costo de las fallas o defectos que no se encuentran con la aplicación del modelo de calidad prepuesto. Estoincluye los mismos puntos que A, pero deberían ser dramáticamente inferiores.

Sin embargo, es importante resaltar que en un análisis más minucioso habría queconsiderar también otros aspectos, tales como:

- Reducciones en los costos de las pruebas y de la integración.
- Reducción del número de cambios en las primeras versiones.
- Reducción en los costos de mantenimientoy otros de no tan fácil cuantificación, tales como:
  - Mejora en satisfacción del cliente.
  - Mejora en la imagen externa de la organización.
  - Mejora en la imagen interna del equipo de ingeniería del software.

V. ANEXO A, TÉCNICAS Y HERRAMIENTAS.

En el siguiente capítulo se complementan las principales Técnicas yHerramientas mencionadas en los módulos del Modelo General de Calidad Propuesto.

Además se describen, sólo a modo de ejemplo, un conjunto de productos entregables (formularios, plantillas, informes, etc.) quese obtienen al aplicar éstas técnicas y herramientas. Lo que se pretende mostrar con estosproductos, son los datos que se deberían registrar para tener una referencia de la correcta aplicación de la misma, más allá del formato utilizado, el soportetecnológico o detalles de implementación.

5.1 Acta de Constitución Del Proyecto (Proceso Inicial)

5.1.1 Ejemplo de un Acta de Constitución del Proyecto

A continuación se mostrará un formato con datos de ejemplo del cómodebería ser el documento

ACTA DE AUTORIZACION DEL PROYECTO (PROJECT CHAPTER)	
Nombre del Proyecto	Estandarización de Procesos y Sistemas de Mantenimiento en Filiales.
Sponsor del proyecto	Mario Orellana , VP de Mantenimiento
Gerente de Proyecto	Alberto Urrutia, Jefe de Proyecto Sénior
Descripción del Proyecto Este proyecto debe analizar los procesos de filiales y homologarlos con los procesos de mantenimiento de casa matriz. También debe implementar los sistemas necesarios para apoyar estos procesos y normalizar los sistemas ya existentes en filiales.	
Necesidad del Negocio: Debido al crecimiento que tendrá la compañía, es necesario trabajar integrados, comunicados, con un lenguaje y sentido común y con procesos eficientes que nos permitan ser más seguros y rentables.	
Justificación del Proyecto: Debido al crecimiento de la compañía se hace necesario trabajar coordinados e integrados con sus filiales con sistemas y procesos comunes que contribuyan a la seguridad y rentabilidad de LAN.	
Requerimientos de los Principales Interesados - Implementar sistemas formales de Matriz en filiales - Unificar operación de mantenimiento en las filiales	
Descripción del Producto/Entregables: - Sistemas y procesos de mantenimiento de Matriz - Documento de Procesos - Plan de Implementación - Manual de Usuario - Instructivos de soporte	

Organizaciones Funcionales y su participación

Gerencia Procesos y Sistemas Implementador del Proyecto

Dirección Filiales USAUsuario Líder

Dirección Filiales Brasil Usuario Líder

Principales Hitos

Reunión de inicio proyecto en Brasil

Reunión de inicio proyecto USA

Implantación en ambas filiales

Reuniones de cierre por filial

Cierre proyecto filiales carguera en Miami , Usa

Cierre proyecto filiales Brasil

Restricciones de la organización, ambientales y externas:

- Idioma
- Relaciones Laborales (Sindicato)
- Resistencia al cambio
- Restricciones de Ancho de Banda

Premisas de la organización, ambientales y externas:

- Los sistemas informales deben ser normalizados para contar con soporte.
- Es un proyecto andamio para el Proyecto principal a ejecutarse el próximo año

Presupuesto:USD 200.000.-

5.2 Estimación del Proyecto

5.2.1 Verificar la Validez de la Estimación de los Costos de Software

Una estimación inapropiada de costos puede dañar más a la calidad del Proyecto deSoftware que a cualquier otro factor, por eso, la verificación puede aumentar la validez de laestimación. La estimación del software es un proceso de tres partes como se describe acontinuación:

- Validar el modelo de estimación.
- Validar que el modelo incluya todos los factores necesarios.
- Verificar la efectividad del modelo de estimación de costos.

5.2.1.1 Validar el Modelo de Estimación

La organización necesita usar un modelo para realizar las estimaciones. El objetivo de estepunto es validar la sensatez del modelo de estimación. Esta verificación deberá desafiar elmodelo usando las siguientes 14 características deseables para la estimación de costos.

Estas características buscan determinar los puntos débiles del modelo:

- El alcance del modelo debe estar bien definido.
- El modelo debería ser ampliamente aplicado.
- El modelo debe ser fácil de usar.
- El modelo debe ser capaz de usar información actual, cuando esté disponible.
- El modelo debe permitir el uso de datos históricos y ajustarlos para una organización enparticular y un tipo de software.
- El modelo debe ser verificado contra un número histórico razonable de proyectos.

- El modelo debe requerir sólo entradas basadas en las propiedades del proyecto, lascuales están bien definidas y pueden ser establecidas con un grado de certezarazonable al momento en que la estimación es llevada a cabo.
- El modelo debe proveer entradas basadas en criterios objetivos.
- El modelo no debe ser hipersensible para criterios de entrada subjetivos.
- El modelo debe ser sensible a todos los parámetros del proyecto que fueronestablecidos teniendo un marcado efecto en el costo y no debe requerir entrada deparámetros no correlacionados con costos.
- El modelo debe incluir estimaciones de cómo y cuándo van a ser necesarios losrecursos.
- El modelo debe producir un rango de valores por la cantidad que ha sido estimada.
- El modelo debe incluir la posibilidad de realizar análisis de sensibilidad, para que elresponsable de la estimación pueda ver la variación de los parámetros seleccionados.
- El modelo debe incluir alguna estimación del riesgo para completar las estimaciones detiempo o costo.

5.2.1.2 Validar que el Modelo Incluya Todos los Factores Necesarios.

Los factores que influencian al costo del proyecto de software deben estar divididos en losque contribuyen en el desarrollo y mantenimiento de la organización y en aquellosinherentes al proyecto de software.

Es importante que los modelos sean usados correctamente y que todos los factores queinfluencian los costos de software sean imputados correctamente. Los modelos puedenproducir resultados incorrectos basados en esa influencia de factores. En primer lugar elfactor puede ser excluido del modelo como resultado de una incorrecta estimación. Ensegundo lugar se puede introducir un factor incorrecto o incompleto al modelo, causandoestimaciones incorrectas de costos de software.

Si un factor no fue incluido, el analista de calidad debe determinar dónde afecta el factorsignificativamente en el costo actual de construir el software. A continuación se describenlos factores influyentes en el costo del software:

- Tamaño del software.
- Porcentaje de diseño y/o código nuevo.
- Complejidad del sistema de software.
- Dificultad de diseño y codificación.
- Calidad.
- Lenguajes a usar.
- Clasificación de nivel de seguridad del proyecto.
- Tipo de máquina a utilizar (tecnología).
- Utilización de hardware.
- Volatilidad del requerimiento

A continuación se describen los factores influyentes, que dependen de la organización:

- Cronograma del proyecto.
- Personal.
- Ambiente de desarrollo.
- Recursos no atribuibles directamente a aspectos técnicos del proyecto.
- Recursos de computación.
- Honorarios.
- Inflación.

5.2.1.3 Verificar la Exactitud del Modelo de Estimación de Costos.

Se proponen las siguientes cuatro verificaciones para validar las estimaciones producidaspor el modelo de estimación de costos de software:

1) ***Re-calcular lo estimado:*** El analista de calidad puede validar el proceso de estimaciónejecutando el modelo de estimación. El propósito de esto es:

- Validar que los datos de entrada fueron ingresados correctamente.
- Validar que los datos de entrada fueron razonables.
- Validar que la ecuación matemática fue calculada correctamente.

2) Comparar estimaciones realizadas con proyectos típicos: El analista de calidadpuede determinar cuánto tiempo lleva desarrollar proyectos del mismo tamaño ycomplejidad.

El cálculo realizado por el sistema de estimación, es luego comparado con costos actualesde proyectos típicos. Si existiera una diferencia, el analista de calidad, puede evaluar más endetalle, la validación de la estimación.

3) *Razonabilidad de la estimación*: Esta verificación es similar a la del punto anterior, seutiliza la experiencia pasada. El analista de calidad documenta los factores que influyensobre la estimación de costos, documenta el cálculo producido por el sistema de estimacióny luego valida la razonabilidad de esa estimación teniendo en cuenta experienciasanteriores. Es recomendable un mínimo de tres experiencias consultadas a los Gerentes deproyecto. En caso en que uno o más no sienta que es razonable la estimación, la mismadeberá ser rechazada.

4) Redundancia en estimaciones de costo de software: El Analista de calidad debere-calcular la estimación usando otro modelo de estimación de costo. Si la diferencia espequeña, la estimación será confiable. Caso contrario se tendría que realizar unainvestigación adicional.

A continuación, se detallan las fuentes de los modelos de estimación de software:

- Desarrollos internos (propios) de la organización.
- Modelos de estimación incluidos en metodologías de desarrollo de sistemas.
- Paquetes de software para desarrollo de estimaciones de software.
- Uso de puntos de función para la estimación de costos de software

5.3 Estimación del Proyecto

La creciente complejidad de los sistemas, combinado con los requerimientos para diseñosestructurados y modulares, ha aumentado la cantidad de elementos de softwaredesarrollados y entregados. El aumento de elementos, más los tradicionales hitos usadospara medir el progreso han hecho subjetivo y a menudo poco confiable el seguimiento deldesarrollo del software y la predicción del consumo progresivo del tiempo.

Se ha utilizado exitosamente un sistema para seguir el progreso del desarrollo de softwarea través de un esquema de puntos ganados. Los puntos están asignados en cada paso enel ciclo de desarrollo del software de la organización. Los pasos son hitos en los cuales unproducto generado es aceptado. Al aceptar los productos se ganan los puntos que poseenasociados. La proporción de puntos ganados sobre el total de puntos posibles se recopila yasí se determina el progreso logrado. Luego, se generan informes, tabulando la informaciónen una variedad de reportes gerenciales.

El sistema así implementado es flexible y altamente automatizado. Los puntos acumuladosson rápidamente verificados, objetivos, y basados en el estado actual del desarrollo delproyecto. Cálculos simples o comparaciones de los puntos acumulados proveen una medidaprecisa del progreso, del desvío en el cronograma y la predicción de progresos futuros.

5.3.1 Cómo Utilizar el Sistema de Puntos

El sistema de acumulación de puntos, propuesto por W. Perry[33], es una extensióndel sistema de "hitos". En su forma más simple se asume que cada ítem o componente delsoftware pasa por procesos de desarrollo similares y hay una cantidad de hitos claramenteidentificables dentro del proceso.

[33] William E. Perry (2000). Effective Methods for Software Testing. Wiley Computer Publishing.

A modo de ilustración, se desarrollarán 5 componentes y 4 hitos definirán el proceso dedesarrollo. Los hitos pueden representar diseños revisados y aceptados, código revisado ycompleto, resultados de prueba verificados, y componentes liberados. En un caso simple,cada hito para cada ítem de software vale 1 punto. En este caso, pueden ganarse 20puntos. Como parte de cada revisión de diseño, inspección de código, prueba deverificación, o entrega, se cumple el hito y se ganan los puntos correspondientes. Al poneren un archivo una lista con los componentes e hitos logrados (puntos ganados) y crearsimples generadores de reportes, puede adquirirse una medida objetiva, precisa y oportunade los resultados. En la tabla 5,1 se muestra cómo es un reporte simple de estado:

REPORTE DEL ESTADO DEL SISTEMA					
ÍTEM	DISEÑO	CODIFICACIÓN	VERIFICACIÓN	ENTREGA	PUNTOS GANADOS
Componente A	1	1			2
Componente B	1				1
Componente C	1				1
Componente D	1	1	1		3
Componente E	1	1			2
TOTALES	5	3	1	0	9
PORCENTAJE 9/20 = 45= %					

Tabla. 5.1 Reporte simple de estado

Este esquema simplificado funciona bien para un conjunto homogéneo de componentesdonde todos tienen similar complejidad y cada hito representa una cantidad de trabajosimilar. A través de la introducción de "pesos" en los factores, pueden manejarse fácilmente,los componentes de complejidad diversa o los hitos que representan esfuerzos disímilespara completarlos.

El motor del sistema es un archivo de datos y algunos reportes simples. El archivo de datoses simplemente una colección de registros, uno por cada ítem que debe seguirse, quecontiene campos para indicar si se ha alcanzado algún hito. Usualmente es ventajoso incluircampos para la descripción de los ítems, analista responsable, identificación del trabajo,entre otros.

El mantenimiento o actualización del archivo puede ser tan efectivo como modificar registros con un editor de línea o tan complejo como crear un programa interactivo para esepropósito. Deben usarse algunos medios de acceso limitado para restringir modificacionesno autorizadas al archivo. Una vez actualizado el mismo, para incluir una entrada de datos alcomponente en desarrollo se actualizan los campos de estado de los hitos a medida que sevan alcanzando tales hitos. En algunos casos, esto puede ser un proceso manual; una vezque el evento ocurrió y se alcanzó el hito, una persona (autorizada) actualiza el estado en elarchivo. En otras circunstancias, en sistemas más sofisticados, un programa puededeterminar que ha ocurrido un hito (compilación sin errores o prueba exitosa) yautomáticamente actualiza el estado del hito.

Una vez armado el archivo, se escriben los programas generadores de reportes paraimprimir el estado. Para proyectos menores, puede ser suficiente un programa quesimplemente imprime cada registro, suma los puntos ganados y definidos, y calcula elporcentaje de puntos ganados. Los proyectos más grandes pueden necesitar varios reportespara subsistemas diferentes o reportes resumen que enfaticen los cambios ocurridos.

5.3.2 Utilización del Sistema de Puntos como Método de Prueba

El método de puntos para seguir el progreso del software puede ser utilizado por elAnalista de calidad en alguna de las tres formas siguientes:

5.3.2.1 Validar el Progreso Informado.

El uso del sistema de puntos por el equipo de aseguramiento de la calidad, elabora unaevaluación del progreso que puede compararse con los reportes de progreso del Gerentedel proyecto. Si el seguimiento del progreso es aproximadamente el mismo utilizando losdos resultados, el examinador puede validar el sentido de los reportes del proyectoproducidos por el equipo del proyecto. Esto es un anexo de lo que la prueba normalmentehace, pero puede ser muy valiosa desde la perspectiva de la alta gerencia. Nótese que sihay una diferencia significativa en la estimación del progreso del proyecto, la diferenciapuede estar en el sistema de puntos o en el sistema que usa el Gerente del proyecto. Elobjetivo de hacer ambas estimaciones es proveer al Gerente del proyecto y a la altagerencia de mayor seguridad respecto de la validez de los resultados del reporte deprogreso.

5.3.2.2 Planeamiento de la Prueba

El método de puntos para el seguimiento del progreso indica cuándo ocurrirá la prueba. Altener un sistema que permita al equipo de proyecto, seguir el progreso, puede asistirlo paraplanificar el uso de los recursos de prueba. El sistema de puntos no requiere muchosrecursos; no obstante está destinado a asistir al equipo de proyecto, indicándole cuando losprogramas/subsistemas estarán disponibles para verificar.

5.3.2.3 Reportar el Estado de la Prueba

El sistema de puntos también indica cuándo está hecha la prueba y cuando se liberan losmódulos a producción. La información contenida en el sistema de puntos es la misma quenecesitará el equipo de proyecto para reportar los resultados de la prueba.

5.4 Matriz de Riesgos

La matriz de riesgos es una herramienta diseñada para identificar y evaluar riesgos ydeterminar qué es lo que el sistema debe hacer con cada uno de ellos.

A continuación se describe cómo usar la matriz de riesgos. En forma ideal la matriz deriesgos comienza en la fase de requerimientos y se desarrolla y termina en la fase dediseño. La implementación de esta herramienta es un proceso de cinco pasos. Los pasosdeben ser ejecutados en la siguiente secuencia:

5.4.1 Identificación del Equipo de Evaluación de Riesgos

La clave para el éxito de la matriz de riesgos es establecer un equipo evaluador de riesgos,cuya responsabilidad será completar la matriz. El objetivo de completar la matriz, es llevar acabo un control adecuado de requerimientos y diseño para reducir los riesgos a un nivelmínimo aceptable.

El grupo de riesgo, puede ser parte del equipo que realiza la toma de requerimientos oparte del grupo de test, o puede ser un equipo seleccionado específicamente con elpropósito de confeccionar la matriz de riesgo. El grupo debería contar con entre 3 y 6miembros, y al menos poseer las siguientes habilidades:

- Conocimiento sobre el uso de la aplicación.
- Entender el concepto de riesgo.
- Habilidad para identificar controles.
- Estar familiarizado con ambos, riesgos de la aplicación y de los servicios de información.
- Entender el concepto de servicio de información y sistema de diseño.

Los candidatos en el grupo de riesgo deberán al menos incluir a alguien del área usuaria, yalguno o todos de las siguientes áreas:

- Auditor interno.
- Consultor de riesgo.
- Representante del área de procesamiento de datos.
- Representante del área de seguridad.
- Representante del área de operaciones.

5.4.2 Identificación de Riesgos

El primer objetivo del grupo evaluador de riesgos, es identificar los riesgos enfocándose enla aplicación, pero no en los riesgos ambientales. El equipo evaluador de riesgo, puedeutilizar uno de los dos métodos siguientes para la identificación del riesgo:

5.4.2.1 Análisis de Escenarios de Riesgos

En este método el equipo de riesgo delibera sobre los riegos potenciales de la aplicaciónusando sus experiencias, juicio experto y conocimiento del área de aplicación. Es importante tenersinergia, así los miembros del grupo pueden cuestionarse uno a otro para desarrollar unalista completa de riesgos que son compatibles a la aplicación.

5.4.2.2 Lista de Verificación de Riesgos

Se provee al equipo de riesgo de una lista con los riesgos más comunes que ocurren enaplicaciones automáticas. El equipo selecciona de la lista aquellos riesgos aplicables a laaplicación. En este método, el equipo necesita pocas habilidades porque la lista de riesgosprovee el estímulo para el proceso, y el objetivo del equipo es

determinar cuáles de losriesgos de la lista son aplicables a la aplicación. He aquí una lista de riesgos para suidentificación:

- Acceso no controlado al sistema.
- Prácticas de seguridad no eficaces para la aplicación.
- Errores de procedimiento en los servicios de información:
  - Procedimientos y controles.
  - Manipulación de medios de almacenamiento.
- Errores del programa.
- Defectos del sistema operativo.
- Fallas en el Sistema de Comunicación:
  - Fallas accidentales.
  - Actas accidentales.

5.4.3 Establecer Objetivos de Control

Durante la fase de requerimientos, deben establecerse los objetivos de control para cadariesgo. Estos objetivos definen el nivel de aceptación del perjuicio de cada riesgoidentificado. Otra forma de manifestar el nivel de daño aceptable, es el objetivo mensurablede control.

La adecuación del control no puede chequearse hasta que no esté definido el nivel deperjuicio de cada riesgo. Por lo tanto, mientras que la definición de los objetivos de controles responsabilidad del usuario y del proyecto, puede llevar a la formación de un grupo deriesgos para definirlos. Una vez definidos los objetivos de control, se pueden chequear losrequerimientos para determinar si son factibles.

La tabla 5.2 muestra un ejemplo de matriz de riesgo al final de la fase derequerimientos para un típico sistema de Facturación y Distribución. Esta matriz muestracuatro riesgos para el sistema de Facturación y Distribución, y objetivos de control para cadauno de aquellos riesgos. Por ejemplo, uno de los riesgos es que el producto sea enviado,pero no facturado. En esta instancia, el objetivo de control es

asegurar que todos los envíossean facturados. En otras palabras, el nivel de perjuicio aceptable de este riesgo es cero, yel equipo del proyecto debe instalar un sistema que asegure que para cada envío que sedespacha se prepara una factura. Sin embargo, nótese que el siguiente riesgo es que elproducto se facturará con un precio o cantidad errónea y que los controles tienenestablecido un nivel de defecto mayor a cero, como los otros dos riesgos.

RIESGO	OBJETIVOS DE CONTROL
Despachado pero no facturado	Asegurar que todos los envíos estén facturados
Facturado con precio o cantidad errónea	Facturar al precio actual el 99% de los ítems básicos y error permitido menor al 10%
Facturado al cliente equivocado	Reducir facturas perdidas a menos del 1%
Despacho del producto o cantidad equivocada	Despachar los productos y cantidades correctas al 99% de los ítems de base

Tabla. 5.2 Ejemplo de matriz de riesgo al final de la fase de requerimientos

5.4.4 Identificar Controles en Cada Sistema

Durante la fase de diseño, el equipo de riesgo identificará los controles en cada fase delsistema de aplicación para cada riesgo identificado. Los segmentos de sistemas máscomunes son:

- Origen: La creación del documento fuente más la autorización asociada con aquellatransacción original.
- Ingreso de Datos: Transferencia de información a un medio legible por la máquina.
- Comunicación: El movimiento de datos desde un punto del sistema a otro. El movimiento

- puede ser manual o electrónico.
- Procesamiento: Aplicación del sistema lógico a los datos.
- Almacenamiento: La retención de datos, por largos o cortos períodos de tiempo.
- Salida: La traducción de la información de computadora a los medios entendibles yutilizables.
- Uso: Satisfacción de la necesidad del negocio a través de los resultados del sistema deprocesamiento.

El equipo de evaluación de riesgos, determinará qué controles son aplicables a quériesgos y los registrará en el segmento correcto del sistema. Al término del desarrollo de lamatriz de riesgo, el equipo de riesgo hará una estimación para verificar si los controles sonlos adecuados para reducir el riesgo hasta el nivel de aceptación identificado en el objetivode control. Esto verificará la efectividad de los controles al finalizar el proceso de diseño. La tabla 5.3 muestra un ejemplo de matriz de riesgo para sistemas de facturación ydistribución al final de la fase de diseño.

Los mismos cuatro riesgos identificados durante la fase de Requerimientos (tabla 5.2)también se muestran en esta matriz, por lo que los controles están asociados a cada riesgo.En este ejemplo, el riesgo de despachar sin facturar muestra que los controles 1, 2 y 3ayudarán a disminuir ese riesgo (para una matriz real estos controles deben describirse). Lamatriz muestra en qué segmento del sistema de aplicación residen aquellos controles.

Después de identificar y registrar los controles, el equipo de riesgo debe determinar siaquellos tres controles y los segmentos a los cuales pertenecen son los adecuados parareducir el riesgo de despachar sin facturar.

RIESGO DEL SEGMENTO DEL SISTEMA	ORIGEN	INGRESO DATOS	COMUNIC.	PROCES.	ALMAC.	SALIDA	USO
Despacho sin facturación	#1			#2			#6
Facturado con precio o calidad errónea		#6		#7 #8 #9	#10	#11	
Facturado al cliente equivocado				#12 #3	#14	#15	**#16**
Despacho de producto o cantidad errónea	#17	#18		#19 #20		#21	**#22**

Tabla 5.3 - Ejemplo de matriz de riesgo al final de la fase de diseño

5.4.5 Determinar si los Controles son adecuados

La verificación termina cuando el equipo de riesgo determina si los controles son losadecuados para reducir cada uno de los riesgos identificados al nivel aceptable.

5.5 Análisis de Factores (Módulo de Requerimientos)

Se llevará a cabo un proceso para estimar las incumbencias asociadas a la fase derequerimientos del desarrollo del sistema. Debe incluirse un programa de verificación paracada ítem. Hay 15 factores a considerar, que cubrirán cada fase del proceso de desarrollo. Paracada factor hay un programa de verificación que tiene en cuenta ciertas consideraciones lascuales se encuentran detalladas más abajo. El programa de verificación enumera aquelloscriterios, que aseguran al equipo de aseguramiento de la calidad, que la magnitud de esapreocupación es mínima. A estos criterios debe responder el equipo de aseguramiento de lacalidad. También se debe realizar una verificación suficiente para evaluar si el equipo deproyecto ha manejado adecuadamente cada factor de verificación.

A continuación, se detallarán brevemente cada uno de los factores a tener en cuenta:

5.5.1 Requerimientos Compatibles con la Metodología

(Factor: Metodología)

El procedimiento utilizado para definir y documentar requerimientos, debe estar presentedurante la fase de requerimientos. Cuanto más formales sean estos procedimientos, sefacilita el proceso de su verificación. El proceso de requerimientos es un proceso de,análisis, toma de decisiones y registro de requerimientos en una forma predefinida parapoder utilizarse luego en el diseño.

5.5.2 Funcionalidad de las Especificaciones

(Factor: Precisión)

La satisfacción del usuario solo puede asegurarse, cuando se han cumplido los objetivosdel sistema. El cumplimiento de estos objetivos solo puede medirse cuando

éstos seanmensurables. Por ejemplo, los objetivos cualitativos, como la mejora de servicio, no sonmensurables, mientras que sí lo es el pedido de procesamiento en cuatro horas de usuario.

5.5.3 Usabilidad de las Especificaciones

(Factor: Facilidad de Uso)

La cantidad de esfuerzo requerido para usar el sistema y la habilidad necesaria, debendefinirse durante la fase de requerimientos. La experiencia muestra que las aplicacionesdifíciles de usar finalmente no se utilizan, en cambio los sistemas funcionales fáciles de usarson altamente utilizados. Entonces los desarrolladores, deberán crear especificacionesfáciles de usar, incrementando así la facilidad del uso del sistema en sí mismo.

5.5.4 Mantenimiento de las Especificaciones

(Factor: Mantenibilidad)

Debe definirse el grado de mantenimiento esperado, así como también las áreas donde loscambios son muy probables. Las especificaciones determinarán los métodos demantenimiento (Ej.: cambio de parámetros introducido por el usuario) y el intervalo detiempo en el cual se necesitan implementar dichos cambios.

5.5.5 Necesidades de Portabilidad

(Factor: Portabilidad)

La capacidad para operar el sistema en diferentes tipos de hardware, o migrarlo a otraversión, deberá enunciarse como parte del requerimiento. La necesidad de tener laaplicación desarrollada como portable, puede afectar significativamente la implementaciónde requerimientos.

5.5.6 Interfaces del Sistema

(Factor: Acoplamiento)

Se debe definir en forma precisa, la información esperada como entrada desde otrossistemas computadorizados y las salidas a entregar a otros sistemas. Esta definición no soloincluye los tipos de información intercambiada, sino también, el momento en el cual debeestar disponible cada interfaz y el procesamiento que se espera como resultado de cada unade ellas. Otros factores a tener en cuenta al definir las interfaces son: privacidad, seguridady resguardo de la información.

5.5.7 Criterios de Performance

(Factor: Performance)

Debe quedar establecido: la eficacia, economía y eficiencia esperadas del sistema. Estosobjetivos son una parte integral del proceso de diseño. Cuando no son alcanzados, lainsatisfacción del usuario está garantizada. Como producto final de la fase derequerimientos, se realizará el análisis del costo-beneficio del factor performance, calculadopara la aplicación.

5.5.8 Necesidades operativas

(Factor: Facilidad de Operación)

Las consideraciones operativas deben definirse durante la fase de requerimientos. Esto esespecialmente importante en sistemas de aplicación manejados por el usuario. Los procesosa seguir para operar el sistema, en otras palabras, los procedimientos necesarios paraprocesar transacciones, deben ser lo más simple posible. También deben considerarseprocedimientos de operación por excepción y monitoreo centralizado.

5.5.9 Tolerancia

(Factor: Fiabilidad)

Debe definirse la confiabilidad esperada de los controles del sistema. Por ejemplo, la fasede requerimientos determinará los requerimientos de control para la

precisión de lafacturación, el porcentaje de órdenes que necesitan procesarse dentro de las 24 horas, etc.La tolerancia de facturación puede afirmar que se procesarán las facturas con tolerancia ±1% de los precios de los productos actuales.

Si no se establecen estas tolerancias, no hay base para asignar y medir la confiabilidad delsistema por período de tiempo. Si no se define el nivel de defectos esperados, normalmentese espera cero defectos. Los controles para lograr cero defectos son antieconómicos.Usualmente es más económico, que surja una cantidad mínima de defectos en el procesopero que sean detectados y mensurables.

5.5.10 Reglas de Autorización Definidas

(Factor: Autorización)

Deben especificarse los métodos de autorización (niveles de seguridad) para asegurar quelas transacciones se procesen de acuerdo con la intención que tiene la organización,respecto del sistema.

5.5.11 Requerimientos de Integridad de Archivos

(Factor: Integridad de Archivos)

Se necesita especificar los métodos para asegurar la integridad de los archivos. Estonormalmente incluye el conjunto de controles que deben mantenerse dentro del archivo eindependientemente de la aplicación. Los controles deben asegurar que los registros dedetalle sean consistentes con los totales de control para cada archivo.

5.5.12 Recuperación ante Fallas

(Factor: Auditoria)

La recuperación abarca los procedimientos de re armado ante la identificación de unproblema grave o falla. Se debe definir para cada aplicación, las necesidades deprocesos e información, si la recomposición es necesaria, se necesita enunciar el

momentopara su ejecución. Este momento, puede variar según la hora del día y el día de la semana.Estos requerimientos de re armado afectan el tipo y disponibilidad de los datos.

5.5.13 Impacto de Fallas

(Factor: Continuidad de Procesamiento)

La necesidad para asegurar la continuidad de procesamiento depende del impacto de lasfallas. Si la falla causa solo problemas mínimos, puede ser innecesario asegurarse elprocesamiento continuo. Por el contrario, cuando la continuidad de la operación es esencial,es necesario duplicar los equipos y centros de cómputos, para que se pueda continuarprocesando.

5.5.14 Nivel de Servicio Deseado

(Factor: Nivel de Servicio)

El nivel de servicio implica tiempo de respuesta basado en los requerimientos. El nivel deservicio requerido variará sobre la base de los requerimientos. Cada nivel de serviciodeseado necesita estar enunciado; por ejemplo, hay un nivel de servicio para corregir unerror de programación, otro para instalar un cambio y otro para responder a una consulta,etc.

5.5.15 Permisos y Accesos

(Factor: Seguridad)

Los requerimientos de seguridad deben desarrollarse mostrando la relación entre recursosde sistema y humanos. Los requerimientos deben enunciar todos los recursos del sistemadisponibles sujetos a control, y luego indicar quién debe tener acceso a aquellos recursos ycon qué propósitos. Al final del módulo, el equipo de aseguramiento

de la calidad, puedeemitir juicio acerca de la adecuación de cada criterio. El equipo debe emitir uno de lossiguientes cuatro juicios acerca de cada criterio:

- Muy adecuado: El equipo de proyecto ha hecho más de lo normalmente esperado parael criterio.
- Evaluación adecuada: El equipo de proyecto ha hecho el trabajo suficiente para asegurarel control por encima del criterio.
- Evaluación inadecuada: El equipo de proyecto no ha hecho el trabajo suficiente, y debentrabajar más en este criterio.
- No aplicable (N/A): Debido al tipo de aplicación o diseño filosófico del sistema por partede la organización, la implementación de este criterio no es aplicable al sistema que seestá revisando.

5.6 Inspecciones

Las inspecciones y las recorridas “walkthrough” apuntan a asistir a los productores paramejorar su trabajo.

Las inspecciones intentan examinar técnicamente el trabajo y proveer a los productores,una evaluación independiente de aquellas áreas productivas en donde las mejoras sonnecesarias. Las recorridas son generalmente menos formales y están conducidos en unformato más educacional. Las inspecciones, por el contrario, generalmente tienen unformato formal, la asistencia es específica, y la información se refleja en los resultados.

Hay tantos tipos de inspecciones como tipos de productos. Es de mucha ayudainspeccionar los requerimientos antes de comenzar el diseño de alto nivel, el diseño de altonivel antes de comenzar el diseño detallado, el diseño detallado antes de comenzar laimplementación, y la implementación antes de comenzar la verificación. Esto no significaque todos los diseños de alto nivel deben ser inspeccionados antes del comienzo decualquier diseño detallado, pero el diseño específico a realizar debe basarse en el trabajoque ha sido inspeccionado. También es de ayuda inspeccionar los casos verificados y ladocumentación.

Se recomiendan las inspecciones para diseño, codificación, casos de prueba, ydocumentación. Caso contrario, es adecuado un proceso de recorrida menos formal.

5.6.1 Proceso

El proceso de inspección sigue ciertos principios básicos:

- La inspección es un proceso formal, estructurado a través de un sistema de listas de verificación (“checklists”) y roles definidos para los participantes. Provee un instrumentoordenado para implementar un

estándar de excelencia de ingeniería del software o deconocimiento en toda la organización.

- Los estándares y listas de verificación genéricas son desarrollados para cada tipo deinspección y, cuando sea apropiado, se ajustan a las necesidades de un proyectoespecífico. Estas listas cubren el planeamiento de la inspección, la preparación, laconducción, la salida y reportes de normas.
- Los inspectores están preparados de antemano y tienen identificados sus tareas ycuestiones antes de que comiencen la inspección.
- El foco de la inspección estará en identificar problemas, no en resolverlos. Este foco,junto a lo mencionado en el punto anterior, asegura que la inspección pueda manejarsecon una mínima pérdida de tiempo.
- Una inspección es conducida por técnicos para técnicos. Los directivos no seinvolucrarán, aunque serán informados de los hallazgos y las fechas en las cuales seresolverán los problemas identificados.
- La información de la inspección deberá ser ingresada a una base de datos y se lautilizará para monitorear la efectividad de la inspección, seguimiento y conducción de lacalidad del producto.

Mientras mucha gente pueda estar interesada en los resultados de la inspección, elpropósito de la inspección es asistir a los productores para mejorar su trabajo. Esto puedehacerse mejor limitándose a cinco o seis los inspectores. El punto clave de esta limitación esla atención técnica.

El moderador no es el director del trabajo que se está inspeccionando, ni tampoco ningunode los otros participantes. Los moderadores necesitan una base completa de los principios ymétodos de inspección antes de que puedan hacer un trabajo competente. Esto les dará lasherramientas básicas y los ayudará a tener mayor confianza en sí mismos, necesaria paraliderar tal actividad.

Como las inspecciones bien realizadas requieren de una intensa concentración de todoslos participantes, éstos pueden quedar exhaustos. Por esto, las sesiones de inspecciónгeneralmente no deben exceder las dos horas.

También es de gran ayuda asignar algunos inspectores en áreas productivas específicasdurante el proyecto.

5.6.2 Participantes

Los participantes de la inspección, se detallan a continuación:

- El moderador (líder de inspección): Persona responsable para liderar la inspecciónpronta y eficientemente a una conclusión satisfactoria.
- Los productores: Persona/s responsable/s de hacer que se inspeccione el trabajo.
- Los examinadores (inspectores): Generalmente es la gente directamente involucrada yconocedora del trabajo que está siendo inspeccionado.
- El registrador (quien anota): Alguien que registra los resultados significativos de lainspección.

5.6.3 Salidas

Las salidas a obtener luego de realizarse una inspección, se detallan a continuación:

- Informe de Inspección (Figura 5.1)
- Resumen de la Inspección (Figura 5.2)

INFORME DE LA INSPECCIÓN								
PROYECTO					FECHA			
SISTEMA					UNIDAD			
MODERADOR					OFICINA			
TIPO DE REUNION	VISIÓN GENERAL		REINSPECCION		REQUERIMIENTOS		DISEÑO	
	CODIFICACIÓN		VERIFICACIÓN		VALIDACIÓN		OTRA	
# INPECCIONES					DURACIÓN			
# REVISIONES					TIEMPO PREPARACIÓN			
ESTADO	ACEPTADA		CONDICIONAL		REINSPECCIÓN		FECHA	
INSPECTORES								
PRODUCTORES								
REGISTRADOR								
CERTIFICACIÓN DEL MODERADOR							FECHA	
COMENTARIOS								

Figura 5.1 - Informe de inspección

RESUMEN DE LA INSPECCIÓN								
PROYECTO					FECHA			
SISTEMA					UNIDAD			
MODERADOR					OFICINA			
TIPO DE REUNION	VISIÓN GENERAL		REINSPECCION		REQUERIMIENTOS		DISEÑO	
	CODIFICACIÓN		VERIFICACIÓN		VALIDACIÓN		OTRA	
TIPO	DEFECTOS IMPORTANTES				DEFECTOS MENORES			
	FALTANTE	INCORRECTO	NO REQUERIDO	TOTAL	FALTANTE	INCORRECTO	NO REQUERIDO	TOTAL
FUNCION								
INTERFAZ								
DATOS								
LOGICA								
ENTRADA/SALIDA								
RENDIMIENTO								
MANTENIMIENTO								
ESTANDARES								
DOCUMENTACIÓN								
FACTORES HUMANOS								
SINTAXIS								
OTROS								
TOTALES								

Figura 5.2 - Resumen de la inspección

5.7 Ranking de Factores de Éxito (Módulo de Diseño)

El ranking (“scoring”) es una herramienta predictiva que utiliza experiencias en sistemasanteriores. Se analizan los sistemas existentes para determinar los atributos ocaracterísticas de los mismos y su correlación con el éxito o el fracaso de cada aplicación enparticular. Una vez que los atributos correlacionados al éxito o al fracaso pueden seridentificados, también pueden ser usados para predecir el comportamiento del sistema queestá en desarrollo.

Los lineamientos que se utilizan bajo el concepto de ranking, se describen a continuación:

- Muestreo: Los atributos que se utilizarán corresponderán a una muestra de todos los atributos abarcados en la implementación de un sistema.
- Alta correlación positiva. Los atributos elegidos tendrán que mostrar una alta correlación positiva en el pasado con cualquier éxito o fracaso durante la automatización de una aplicación. Estos atributos no deberían ser intuitivos sino, que deberán ser atributos para los cuales pueda demostrarse que la ausencia o presencia de los mismos muestre una alta correlación con respecto al resultado final del proyecto.
- Facilidad de uso. Para ser efectivo, el proceso de ranking debe ser lo más simpleposible.
- Desarrollar el ranking de riesgo. El ranking para cada atributo debe determinarse en unformato mensurable de modo que el ranking de riesgo total pueda ser desarrollado paracada aplicación. Esto indicaría el grado de riesgo, el área de riesgo, y también unacomparación de riesgos entre las diferentes aplicaciones.

Al ranking de riesgo se llega a través de un desarrollo matemático, como se muestra acontinuación.

RIESGO DE LA CARACTERÍSTICA	# DE CARACTERÍSTICAS EVALUADAS	FACTOR DE MULTIPLICACIÓN	FACTOR PARA EL RANKING
Alta	3	3	9
Media	9	2	18
Baja	12	1	12
TOTAL			39

Figura 5.4 - Desarrollo Matemático para Obtener un ranking de riesgo

Para usar esta tabla, el número de atributos o características calificadas con alto, medio ybajo, deberá ser totalizado, y los totales colocados en la columna “# (cantidad) deCaracterísticas Evaluadas”.

Luego a cada número se lo multiplica por el factor de multiplicación y así obtener el rankingde riesgo. Por ejemplo, el número total de características definidas con alto riesgo debemultiplicarse por tres. Luego los tres números se suman para llegar a totalizar el riesgo total,el cual puede utilizarse para comparar entre los diferentes sistemas, o bien, respecto de unestándar definido por la organización. cuanto más alto es el puntaje, mayor será el riesgo, ya menor puntaje menor riesgo.

5.8 Análisis de Factores (Módulo de Diseño)

Los factores que deben analizarse durante el módulo de diseño se describen acontinuación:

5.8.1 Controles de Integridad de Datos

La integridad de datos va desde la identificación de riesgos, hasta la decisión gerencial deaceptar esos riesgos, establecida en términos de la cantidad de pérdidas aceptables. Loscontroles de integridad de datos se diseñan a partir de la tolerancia a tales riesgos, loscuales se encuentran especificados.

5.8.2 Reglas de Autorización

Las autorizaciones en los sistemas pueden ser manuales y/o automáticas. Losprocedimientos para autorización manual deben especificarse durante la fase de diseño. Losmétodos de autorización automática deben especificarse más detalladamente que losmanuales, por el hecho de que no se pueden dejar librados a criterios personales según lasituación que se presente.

5.8.3 Controles de Integridad de Archivos

La integridad de los archivos estará asegurada por los métodos de identificación dearchivos, controles automáticos y controles de integridad de archivos mantenidosindependientemente. Las especificaciones para este proceso integrador de tres partes debedeterminarse durante la fase de diseño.

5.8.4 Pistas de Auditoria

Estas pistas proveen la capacidad para rastrear transacciones desde su origen hasta loscontroles. Además, las pistas de auditoría se usan para consolidar el procesamiento detransacciones individuales y recuperar la integridad de operaciones luego de su pérdida.Frecuentemente los entes gubernamentales especifican las pistas de auditoría quenecesitan retener para cada tipo de información que manipulan. Estas necesidades deinformación, deben definirse durante la fase de diseño.

5.8.5 Plan de Contingencias

El plan de contingencias esquematiza las acciones a tomar ante la aparición de problemas.Este plan deberá incluir los métodos manuales a seguir cuando las aplicacionesautomatizadas no estuvieran en operación, así como también, las consideraciones de lugarfísico. Las especificaciones del plan de contingencia deben llevarse a cabo durante la fasede diseño.

5.8.6 Método para Alcanzar el Nivel de Servicio Requerido

La fase de requerimientos define el nivel de servicio a obtener durante la operación de laaplicación. El método para alcanzar ese nivel de servicio es desarrollado durante la fase dediseño. Esto involucra el desempeño del sistema y su habilidad para satisfacer lasnecesidades de los usuarios a lo largo del tiempo.

5.8.7 Procedimientos de Acceso

La seguridad en los sistemas automatizados es llevada a cabo predefiniendo quién puedetener acceso a qué recursos y para hacer qué. Esto estará indicado por los perfiles deseguridad definidos. Durante la fase de diseño, deberán desarrollarse los procedimientos,las herramientas y las técnicas necesarias para crear e implementar los perfiles deseguridad necesarios.

5.8.8 Diseño Acorde con la Metodología

El proceso de diseño de sistemas debe ejecutarse y documentarse de acuerdo con lametodología establecida por la organización. Los procedimientos estándares de diseñoaseguran la facilidad de comprensión por todos los miembros del equipo capacitados yentrenados en esa metodología y al mismo tiempo ayuda a asegurar que se cumpla enforma completa el proceso de diseño.

5.8.9 Diseño Acorde con los Requerimientos

El diseño del sistema es una transformación de los requerimientos de los usuarios enespecificaciones más detalladas. Durante cualquier transformación, pueden ocurrir malosentendidos o malas interpretaciones. Entonces, deben tomarse las medidas necesarias paraasegurar que el diseño cumpla sus objetivos y esté focalizado a cumplir con losrequerimientos definidos.

5.8.10 Facilidad de Uso

Cuanto el producto final sea más fácil de usar, habrá mayor probabilidad que el usuario loutilice y que las transacciones sean procesadas correctamente. Deben considerarse lashabilidades necesarias para cada puesto de trabajo y la motivación en el mismo, de todaslas personas usuarias del sistema.

5.8.11 Mantenibilidad del Diseño

El costo de mantener una aplicación normalmente excede el costo de desarrollo. Identificaraquellos aspectos del sistema que son los más factibles de cambiar y construir aquellaspartes del sistema para facilidad del mantenimiento es un aspecto importante del proceso dediseño. La mantenibilidad del diseño puede cambiar significativamente dependiendo de lafrecuencia esperada de los cambios introducidos.

5.8.12 Portabilidad del Diseño

Si los requerimientos indican que el sistema debe poder ser transferido desde un ambientede hardware a otro, o una versión de software de base a otra, el diseño debe tener encuenta e incorporar estos aspectos de portabilidad. Cuando el hardware y software futurosson inciertos, el diseño debe ser lo más general posible y no intentar sacar ventaja de losaspectos o facilidades que brindan el hardware y software existentes.

5.8.13 Interfaces de Diseño

Deben ser identificadas y especificadas y documentadas las interfaces con otrasaplicaciones. Las especificaciones de las interfaces deben también considerar usosalternativos de la información de la aplicación.

5.8.14 Diseño Acorde con Criterios Establecidos

El estudio de costo/beneficio realizado durante la fase de requerimientos no provee unaestimación de gran precisión. La estimación de la fase de requerimientos puede estarafectada en más o menos el 50%. Durante la fase de diseño, la estimación del desempeñopuede ser definida mejor por lo que se puede hacer una mejor predicción y así lograr elcumplimiento del criterio de desempeño establecido. Una guía útil a ser utilizada, es que laprecisión de la estimación del criterio de desempeño al final de la fase de diseño puedevariar en ±10%.

5.8.15 Necesidades Operacionales

El sector de operaciones deberá identificar los requerimientos de procesamientos futurospara preparar el manejo de tales requerimientos cuando el sistema se implemente. Cuantomás grandes sean los requerimientos de procesamiento, mayor

será la necesidad deinvolucrar al sector de operaciones en la consideración de las alternativas de diseño.

5.9 Revisión del Diseño

El objetivo es identificar aquellos atributos del diseño que se correlacionan con losproblemas del sistema. Entonces durante la revisión de diseño, se investigarán dichosatributos para determinar si el equipo de proyecto los ha tratado apropiadamente.

El equipo de revisión de diseño comprende los siguientes miembros:

- Personal del proyecto: El personal del proyecto puede conducir su propia revisión deldiseño. Normalmente la persona asignada con la responsabilidad de la revisión delproyecto no es la misma que realmente diseñó el sistema; sin embargo el que revisapuede tener responsabilidad parcial en el diseño. Esto requiere que las personas duranteel proceso de revisión acepten roles y responsabilidades diferentes a los que han tenidoen el proceso de diseño. Debido a probables vínculos con el diseño real del sistema,tener una lista de verificación de revisión de diseño como herramienta evaluadora,normalmente cumple una valiosa función para el/los que revisan.
- Equipo de revisión independiente: Los miembros de este equipo de revisión no sonmiembros del proyecto que está siendo revisado. Pueden ser de otros proyectos o delequipo de aseguramiento de la calidad. Esta manera de operar provee un mayor gradode independencia para conducir la revisión ya que no habrá conflicto de intereses entrelos roles de diseñador y del que revisa. Por otro lado es frecuentemente difícil para losinspectores ser críticos unos con otros, especialmente en situaciones donde el querevisa pueda eventualmente trabajar para la persona que está siendo revisada.

La siguiente es una guía en la conducción de una revisión de diseño:

- Seleccionar el equipo de revisión: Los miembros del equipo de revisión debenseleccionarse antes del proceso de revisión propiamente dicho.
- Entrenar los miembros del equipo de revisión: Las personas que conducirán la revisióndeben estar entrenadas sobre cómo dirigir la revisión. Mínimamente significa revisar elchecklist y explicar el objetivo e intención de cada asunto. También es aconsejableentrenar a las personas involucradas en relaciones interpersonales y así realizar larevisión en un ambiente armonioso.
- Notificar al equipo del proyecto: El equipo del proyecto debe estar notificado de larevisión con varios días de antelación cuando comienza la revisión y su responsabilidaddurante la misma. Es importante organizar formalmente la revisión para que estén todospresentes.
- Asignar tiempos adecuados: La revisión debe conducirse formalmente y taneficientemente como sea posible. Aún cuando la misma gente que diseñó la revisión, ladirija, las relaciones interpersonales y los efectos de la sinergia de la revisión puedenproducir muchos efectos positivos si se asigna el tiempo suficiente para permitir unainteracción apropiada.
- Documentar los datos de la revisión: Deben registrarse todos los hallazgos realizados enla revisión. Normalmente esto puede llevarse a cabo en el checklist, a menos que loscomentarios sean muy extensos. En cualquier caso, los datos deben hacer referencia apreguntas específicas del checklist que cubrían.
- Revisar los datos con el equipo de proyecto: La precisión de los datos debe sercomprobada por todas las personas involucradas, y la revisión no debe continuar si estono está hecho. Es mejor hacerlo al final de la revisión que durante el proceso mismo.
- Desarrollar recomendaciones de revisión: Basándose en los datos, el equipo de revisiónhará sus recomendaciones para corregir cualquier situación de conflicto. Estasrecomendaciones son parte importante del proceso de revisión.

- Revisar recomendaciones con el equipo de proyecto: El equipo de proyecto debe ser elprimero en recibir las recomendaciones y tener la oportunidad de aceptar, modificar orechazar las recomendaciones.
- Preparar un reporte: Se debe preparar un reporte para documentar los hallazgos, y lasacciones tomadas o a tomar acerca de las recomendaciones. Este reporte puede o noenviarse a altos niveles gerenciales, dependiendo de las reglas establecidas en larevisión por la organización. Sin embargo es importante tener un registro formal delproceso de revisión, qué se encontró y las acciones tomadas respecto de lasrecomendaciones.

La cantidad de revisiones dependerá de la importancia del proyecto y del lapso de tiempoen la fase de diseño. Así podrán llevarse a cabo revisiones del diseño de alto nivel y deldiseño detallado.

5.9.1 Revisión del Diseño de Alto Nivel

Cada defecto descubierto durante la revisión del diseño lógico o de alto nivel debedocumentarse. Se confecciona un reporte de defectos para llevar registro de los mismos,incluyendo tipo y categoría de los defectos. La descripción de cada defecto se registra bajouna columna de Faltante, Erróneo o Adicional. Al final de la revisión del diseño lógico, losdefectos se resumen y totalizan. La figura 5.4 muestra un formulario de registro dedefectos en la fase de diseño de alto nivel:

REGISTRO DE DEFECTOS EN LA FASE DE DISEÑO DE ALTO NIVEL				
CATEGORÍA	DEFECTO			
	FALTANTE	ERRÓNEO	ADICIONAL	TOTAL
LA INFORMACIÓN NO HA SIDO DEFINIDA ADECUADAMENTE				
DEFINICIÓN DE ENTIDAD INCOMPLETA				
ENTIDAD INCORRECTA				
ATRIBUTOS DE ENTIDAD INCORRECTOS				
VIOLACIÓN DE NORMALIZACIÓN				
CLAVE PRIMARIA INCORRECTA				
CLAVE FORÁNEA INCORRECTA				
SUBTIPO DE ENDIDAD INCORRECTO				
PROCESO NO DEFINIDO ADECUADAMENTE				
OTROS				
TOTALES				

Figura 5.4 - Formulario de registro de defectos en la fase de diseño de alto nivel

5.9.2 Revisión del Diseño Detallado

Cada defecto descubierto durante la revisión del diseño físico o detallado debedocumentarse. Se confecciona un reporte de defectos para llevar registro de los mismos,incluyendo tipo y categoría de los defectos. La descripción de cada defecto se registra bajouna columna de Faltante, Erróneo o Adicional. Al final de la revisión del diseño físico, losdefectos se resumen y totalizan. La figura 5.5 muestra un formulario de registro dedefectos en la fase de diseño detallado:

REGISTRO DE DEFECTOS EN LA FASE DE DISEÑO DETALLADO				
CATEGORÍA	DEFECTO			
	FALTANTE	ERRÓNEO	ADICIONAL	TOTAL
SECUENCIA O LÓGICA ERRÓNEA				
PROCESAMIENTO INCORRECTO				
LAS ENTRADAS DEL MÓDULO SON INCORRECTAS				
LAS SALIDAS DEL MÓDULO SON INCORRECTAS				
EL MÓDULO NO ACEPTA LOS RANGOS DE DATOS ESPECIFICADOS				
LOS PROCEDIMIENTOS DE RECUPERACIÓN NO ESTÁN / SON INCOMPLETOS / NO SON ADECUADOS				
EL PROCESAMIENTO EN EL MÓDULO DIFIERE DE LO ESPECIFICADO				
VALORES ACEPTADOS ERRÓNEOS / AMBIGUOS				
ALMACENAMIENTO DE DATOS ERRÓNEO / INADECUADO				
VARIABLES FALTANTES O ERRÓNEAS				
OTROS				
TOTALES				

Figura 5.4 - Formulario de registro de defectos en la fase de diseño de alto nivel

5.10 Depuración de Programas

La depuración (“debugging”) permite al programador evaluar la integridad y precisión delprograma, previo a la conducción de revisiones y verificaciones menos económicas. Ladepuración, incluye el diseño detallado y la codificación.

Esta operación puede ser tan extensa o tan reducida como se quiera. La cantidad dedepuraciones a realizar dependerá de:

- El tiempo de espera hasta que se reciba el siguiente programa.
- Cronograma de implementación.
- Recursos de prueba
- Eficiencia de herramientas de test.
- Políticas del área.

La depuración puede ser sintáctica, estructural, o funcional.

5.10.1 Depuración Sintáctica

Las especificaciones y expresiones del programa deben desarrollarse de acuerdo con lametodología definida por la organización y los requerimientos recopilados. El programadorpuede chequear las expresiones y sintaxis para asegurarse que ha escrito el programa deacuerdo con las reglas establecidas. Al chequear la sintaxis se hacen preguntas como:

- ¿La función a realizar, está claramente identificada?
- ¿Las sentencias del programa están identificadas correctamente?
- ¿Las sentencias del programa están construidas utilizando la estructura apropiada?

- ¿Los elementos de datos están apropiadamente identificados?
- ¿Las estructuras de datos son las adecuadas para colocar los valores que seránutilizados en dichas estructuras?

5.10.2 Depuración Estructural

Los problemas de estructura crean un significante número de defectos en la mayoría de lasaplicaciones. Estos defectos ocultan defectos funcionales por lo que su detección se tornadifícil. Las preguntas típicas durante la depuración estructural son:

- ¿Se colocaron todas las sentencias necesarias?
- ¿Se utilizan todas las definiciones de datos en las sentencias definidas?
- ¿Se utilizan todos los elementos de datos definidos?
- ¿Las tablas internas y valores límite están usados de manera que cuando seexcede ellímite se puede continuar procesando?

5.10.3 Depuración Funcional

Las funciones son los requerimientos que el programa ejecutará. Las preguntas cuando sedepura la funcionalidad son:

- ¿El programa ejecutará la función específica en la forma indicada?
- ¿Algunas de las funciones son mutuamente excluyentes?
- ¿El sistema detectará datos incorrectos o ilógicos?
- ¿Se acumulará apropiadamente la información entre diferentes ejecuciones delprograma?

5.11 Análisis de Factores (Módulo de Codificación)

La profundidad de la verificación en el módulo de codificación, depende de cómo se adecua el sistema a las necesidades del usuario, al final del módulo de diseño. A mayor confianza del equipo de verificación, en la adecuación de la aplicación al final del módulo de diseño, habrá menos inquietudes durante el módulo de codificación.

En el módulo de codificación, el equipo de calidad de software debe identificar las inquietudes que sean de mayor interés, y luego desarrollar el proceso de verificación para hacer frente a dichas inquietudes. Al identificarlas, el equipo de calidad, debe tener en cuenta los cambios que han ocurrido en las especificaciones del sistema desde que se produjo la última verificación. Los objetivos que los miembros del equipo de calidad deben considerar en el módulo de codificación, son:

- ¿Los sistemas son mantenibles?
- ¿Se han implementado correctamente las especificaciones del sistema?
- ¿Los programas son compatibles con los estándares y procedimientos?
- ¿Existe un plan de verificación capaz de evaluar los ejecutables o entregables?
- ¿Los programas están adecuadamente documentados?

Las inquietudes (“concerns”) a considerar durante esta tarea se describen a continuación:

- Implementación del control de integridad de información: Es necesario tenerimplementados controles específicos de manera de lograr la precisión en elprocesamiento deseado. Los controles implementados en forma impropia, no alcanzaránel nivel de tolerancia aceptado, y por la mala comprensión del propósito de los controles,serán implementadas

soluciones simplistas cuando en realidad se requieren controlescomplejos para alcanzar los objetivos de control establecidos previamente.

- Implementación de reglas de autorización: Es necesario verificar la implementación dereglas de autorización de manera de dificultar su evasión. Además, las reglas deautorización no deben sólo considerar la ejecución de las reglas si no también tener encuenta los métodos más comunes de evadirlas.
- Implementación de controles de integridad de archivos: Los controles de integridad dearchivos deben implementarse de manera que minimicen la probabilidad de pérdida de laintegridad, debiendo además prevenirla y detectarla oportunamente.
- Implementación de auditorías de rastreo: Es necesario implementar una verificación delas auditorias para facilitar la recuperación de información de las mismas (rastreo). Si laverificación de la auditoria contiene información costosa o demanda mucho tiempo, suvalor disminuye significativamente. Las consideraciones de la implementación incluyen lacantidad de información a recuperar seguida de la facilidad de recuperación, referenciascruzadas de información para la recuperación y el tiempo que la información de laauditoria necesita ser almacenada.
- Plan de contingencia escrito: El plan de contingencia, es una serie procedimientos detalladosperfilando aquellas tareas a ejecutar ante la ocurrencia de problemas, debe describir lastareas preparatorias para que los datos necesarios y otros recursos estén disponiblescuando sea necesario activarse. Abordar el diseño de la contingencia es de poco valor sino se documenta o no estará en manos de la gente que lo utilizará.
- Consecución del nivel de servicio deseado para el sistema: El nivel de servicio deseadopuede solo hacerse realidad cuando los procedimientos y métodos esténimplementados. Uno de los procedimientos que debe establecerse, es el monitoreo delnivel de servicio para asegurarse que cumple con las especificaciones. La inclusión de larutina de monitoreo

provee seguridad en el logro del nivel de servicio a largo plazo, casocontrario, se detectará a tiempo para tomar medidas correctivas.

- Implementación de procedimientos de seguridad: La seguridad es la combinación deprevisión y entrenamiento, más herramientas y técnicas de seguridad. Losprocedimientos que aseguran que tanto las herramientas como las técnicas de seguridadestén disponibles y que trabajen juntas, deben desarrollarse durante la fase decodificación.
- Cumplimiento del programa con la metodología a adoptar: Los procedimientos aimplementar deben asegurar conformidad con estándares, políticas, procedimientos ymétodos definidos por la organización. Si se detecta la no conformidad, se deberíantomar las medidas necesarias para modificar el diseño y así lograr la conformidad.
- Programas acordes al diseño (Correctitud): El cambio continuo de condiciones puedeprovocar que varios miembros del personal del proyecto, ignoren los objetivos del mismodurante la fase de codificación. El argumento es que siempre hay cambios, de maneraque el ajuste a los objetivos del sistema ya definidos, ahora ya no es muy significativa. Elequipo de aseguramiento de la calidad, debe desalentar esta forma de pensar ycontinuamente monitorear la implementación de dichos objetivos. Si no se hanalcanzado los mismos, o deben cambiarse, o bien, debe cambiarse el sistema para tratarde ajustarlos a las especificaciones funcionales ya realizadas de la aplicación.
- Programas acordes al diseño (Facilidad de uso): La implementación de especificacionesdel sistema puede invalidar algunas de los aspectos del diseño respecto de la facilidadde uso, a menos que los mismos se encuentren definidos específicamente. Laprogramación es la traducción de especificaciones de diseño a lenguaje ejecutable por lamáquina. Esto puede entorpecer el logro de la facilidad de uso. La codificación debelograr la facilidad de uso, de igual forma en que se intenta cumplir con el resto de lasespecificaciones funcionales.

- Mantenibilidad de los programas: El método de codificación puede significar más para elmantenimiento que las especificaciones de diseño mismas. Las reglas de mantenimientodeben definirse en parte por los estándares y en parte por las especificaciones delsistema. Además, el programador debe utilizar su juicio y experiencia para desarrollarcódigo altamente mantenible.
- Programas acordes al diseño (Portabilidad): La portabilidad de los programas dependedel lenguaje seleccionado y de cómo se usa ese lenguaje. Las especificaciones debenindicar lo que se hace y no se hace en la programación para lograr su portabilidad, y lacodificación debe apegarse a esas especificaciones de diseño. Si la portabilidad es unapreocupación importante y las especificaciones del programa fallan al definiradecuadamente la portabilidad de la codificación, la misma quedará en manos delprogramador.
- Programas acordes al diseño (Acoplamiento): Las especificaciones de diseño debenindicar parámetros a pasar desde y hacia otros módulos y aplicaciones del sistema.Normalmente es una buena práctica del programador, verificar que las especificacionesdel sistema estén actualizadas antes que las funciones sean codificadas. Esto no soloasegura que el programa se ajusta al diseño, sino también, que las especificaciones de interconexión entre aplicaciones no se han modificado desde que se documentó eldiseño.
- Desarrollo de procedimientos operativos: Los procedimientos deben desarrollarsedurante la fase de programación, de forma de poder operar la aplicación. Durante la fasesiguiente, los programas ejecutables serán operados. Los procedimientos operativosdeben ser consistentes con los requerimientos operacionales definidos para laaplicación.
- Alcance de los criterios de performance por parte de los programas: La creación delprograma provee la primer oportunidad para los usuarios de evaluar si el sistema puedelograr el nivel rendimiento deseado. Una

evaluación temprana del rendimiento, brindaráuna buena oportunidad para hacer ajustes si es necesario.

5.12 Revisiones por Pares

La revisión, por pares, contribuye a construir el código de un programa a través delas revisiones informales, pero sin embargo efectivas, acerca de la funcionalidad del programa.Este tipo de revisión provee un análisis estático que evalúa la estructura y funcionalidad delprograma. Puede detectar errores sintácticos en mayor medida que una simple observaciónvisual, como resultado de un recorrido (walktrhough) de requerimientos.

La revisión, por pares, también puede ser formal. Las formales son tareas integrales en elproceso de programación, mientras que las informales son solicitadas a discreción del líderde programación. Además puede aplicarse a otros productos, no sólo al código de unprograma.

El equipo de revisión por pares debe tener entre tres y seis miembros. Es importante tenerpor lo menos tres miembros para obtener variedad de opiniones. Las personas a considerarpara este equipo serán:

- Programadores (por lo menos dos).
- Especialistas en Testing o similar.
- Operadores.
- Líderes de programación.

El programa de revisiones por pares se realiza ejecutando las siguientes tareas:

5.12.1 Establecer Reglas Básicas para la Revisión

Esto no es necesario para cada revisión, pero es importante tener buenas reglas de base.Entre ellas están:

- Áreas incluidas y excluidas de la revisión por pares. Por ejemplo: Ver si la eficiencia delos programas será incluida.
- Cuándo se usarán reportes.
- Método para seleccionar el líder de la revisión por pares.
- Ubicación de la conducción de la revisión.
- Método para seleccionar productos a ser revisados por pares.

5.12.2 Seleccionar al Equipo de Revisión

Los integrantes del equipo deben ser seleccionados con la suficiente antelación, demanera que puedan organizar su tiempo y estar entrenados para la revisión.

5.12.3 Entrenar a los Miembros del Equipo

Si una persona del equipo no ha participado previamente en el programa de revisión porpares, se la debe entrenar en el proceso. El entrenamiento incluye el entendimiento de lasreglas básicas de esta revisión, preferentemente algún entrenamiento en relacionesinterpersonales acerca de cómo entrevistar y trabajar con gente en el proceso de la revisión,y entrenamiento en los estándares y metodología de programación.

5.12.4 Seleccionar el Método de Revisión

El líder del equipo debe seleccionar el método de revisión. La revisión en sí misma consisteen dos partes. La primera es una explicación general de los objetivos y funcionamiento delprograma. La segunda parte es la revisión de los programas utilizando el métodoseleccionado. Los métodos que pueden ser utilizados para conducir la revisión por paresson cuatro:

- Diagrama de flujo: El programa se explica desde un diagrama de flujo. Esto es másefectivo cuando el diagrama de flujo es producido directamente desde el código fuente.

- Código fuente: La revisión examina cada línea del código fuente para entender elprograma.
- Muestra de transacciones: El líder de programación explica los programas exponiendolos procesamientos típicos que ocurren a partir de una muestra representativa detransacciones.
- Especificaciones de programas: Las especificaciones de programas se revisan como unmedio para entender el programa.

5.12.5 Conducir la Revisión

El líder de programación del proyecto normalmente inspecciona la revisión por pares. Larevisión comienza habiendo hecho el líder de programación, una revisión rápida de lasreglas básicas, explicado los objetivos, y luego conduce al equipo a revisar el procesamientodel programa. El equipo de revisión será libre de preguntar y comentar sobre cualquieraspecto de la explicación del programador y de hacer recomendaciones y sugerenciasacerca del programa. Generalmente, la revisión por pares es dirigida en forma democrática.

El rol del líder del equipo es asegurar que las preguntas y comentarios del equipo seanordenados, asegurar los derechos de hacer preguntas, recomendaciones o parar en unpunto específico si, en la opinión del líder de inspección, no tiene sentido continuardiscutiendo.

5.12.6 Conclusiones

Al final de la revisión por pares, el líder de programación indicará cuándo no tiene máscomentarios que hacer y lo deja en manos del líder del equipo de revisión por pares. El líderdel equipo de revisión por pares, toma el control y resume la información bosquejada de larevisión y presenta las recomendaciones del equipo de revisión. Idealmente, esto se hacecomo actividad grupal, pero algunos equipos de revisión por pares, especialmente cuando elproceso se formaliza, puede querer algún tiempo a solas para discutir entre ellos lo que hanescuchado y lo que van a recomendar. Las

conclusiones y recomendaciones luego sepresentan al equipo del proyecto para su consideración.

5.12.7 Reportes

En el proceso de la revisión por pares, se prepara el reporte documentando los resultados.Sin embargo, esto es opcional y no es una parte esencial del proceso de revisión por pares.

El formato de los informes típicos será similar al que ya se han detallado con anterioridad:

- Informe de Inspección
- Resumen de la Inspección

5.13 Verificación de Documentación

5.13.1 Integridad de la Documentación

Los criterios principales, para la verificación de la integridad de la documentación, surgiránde la metodología y estándares de la organización. A continuación se detallan criteriosadicionales, para verificar la integridad de la documentación:

- **Contenido de la documentación**: Al índice de cada documento le debería seguir unabreve descripción de cada elemento dentro del documento. Cada documento debe tenerun índice de contenidos mínimos obligatorios, para poder determinar si los documentoscontienen toda la información necesaria. Si faltan elementos, hay un defecto potencial enla integridad de la documentación, la cual debería ser notificada.
- **Usuarios del documento**: Cada tipo de documento estará escrito dirigido a un usuario enparticular, el cual podrá ser una persona o grupo, los cuales usarán el documento paraejecutar una función. (Por ejemplo, operación, mantenimiento, diseño y programación).La información deberá ser presentada con la terminología y el nivel de detallesapropiados para el tipo de usuario al cual está dirigido el documento.
- **Redundancia**: Los documentos típicos dentro del proyecto (Ej. Especificación deRequerimientos, Análisis de Factibilidad, etc.) tendrán alguna redundancia. Se deberáincluir material introductorio en cada tipo de documento para proveer al lector de unmarco de referencia, facilitando la interpretación del documento con la menor cantidadde referencias cruzadas hacia otros documentos. La información que debería serincluida en cada tipo de documento diferirá en su contexto y a veces en la

terminología ynivel de detalle, porque intenta ser leído por diferentes usuarios en diferentes puntos delciclo de vida del software

- **Flexibilidad**: La flexibilidad en el uso de los documentos depende de los lineamientosvigentes en la organización.
- **Tamaño del documento**: Cada tipo de documento puede tener un tamaño que va deunas pocas a varias decenas de hojas. La longitud depende del tamaño y complejidaddel proyecto y del juicio del Gerente de proyecto así como el nivel de detalles necesariopara el ambiente en el cual el software deberá desarrollarse y ejecutarse.
- **Combinación de diferentes tipos de documentos**: En ocasiones, es necesario combinaralgunos tipos de documentos bajo una sola portada o producir algunos volúmenes conlos mismos tipos de documentos. Los tipos de documentos que pueden ser combinadosson manuales para el usuario, operaciones, y mantenimiento del programa. Parasistemas de gran envergadura, puede ser preparado un documento para cada módulo.Algunas veces el tamaño del documento puede requerir ser repartido en variosvolúmenes para hacer más fácil su uso. En esos casos, deberán estar separados porsecciones, sub-secciones, etc.
- **Formato**: El formato obligatorio debe ser verificado, y debe recomendarse el uso delmismo.
- **Secuencia de contenido**: En general, el orden de las secciones y párrafos de un tipo dedocumento debe ser el mismo que el que se muestra en el índice de contenidosobligatorios. El orden puede cambiarse si enriquece la presentación, caso contrario,debería ajustarse a los estándares ya definidos por la organización.
- **Títulos de secciones/párrafos**. Estos títulos generalmente son los mismos que losmostrados en la índice de contenidos. Pueden modificarse para reflejar la terminologíadel software a documentar si el significado le agrega valor a la presentación. Se puedenagregar o eliminar secciones o párrafos según sea necesario.

- **Expansión de los párrafos**: La mayoría de los tipos de documentos definidos por lametodología, estándares o lineamientos de la organización tienen párrafos con un títulogeneral y una lista de puntos que pueden discutirse dentro de ese párrafo. La intenciónde esa lista no es establecer la discusión de cada uno de estos puntos, sino sugerir laconsideración de los mismos al escribir el párrafo. Este y todos los otros párrafos puedenexpandirse y subdividirse para llevar a cabo una mejor presentación de la información.
- **Diagramas de flujo y tablas de decisión**: Los gráficos de las soluciones a problemas enforma de diagramas de flujo o tablas de decisión, pueden estar incluidos o ser unapéndice del documento.
- **Formularios**: El uso de formularios específicos depende de las prácticas de laorganización. Parte de la información especificada en los párrafos del índice decontenidos puede registrarse en tales formularios. Si es así, el formulario puedereferenciarse desde el párrafo apropiado. Se requiere usar los formularios queconforman los estándares de la organización.

5.13.2 Grado de actualización de la Documentación

El equipo de verificación de la documentación puede usar cualquiera de las siguientescuatro verificaciones propuestas para validar la actualización de la documentación. Los tiposde verificación posibles se enumeran a continuación:

5.13.3 Utilizar la Documentación Vigente

La actualización puede validarse utilizando la documentación vigente para realizar algúncambio en la aplicación. Esto permite al analista de calidad buscar y confirmar laconsistencia entre varios documentos (por ejemplo, que las especificaciones en eldocumento de diseño del programa, sean las mismas que están en el código actual) ydeterminar si la documentación respalda el funcionamiento del sistema.

5.13.4 Comparar el Código Fuente con la Documentación

Esta verificación usa la versión actual del código fuente de uno o más programas comobase de la documentación. Esta verificación es realizada sobre la base de una muestraelegida al azar de los programas o parte de los mismos y se las verifica contra ladocumentación. El objetivo es determinar si el código es representativo de la documentaciónque se posee del mismo.

5.13.5 Verificar la Vigencia de la Documentación

Las personas que elaboran la documentación deben verificar que esté vigente. Debenhacerse preguntas específicas, como éstas:

- ¿La documentación es 100% representativa de la aplicación en operación?
- ¿La documentación cambia cada vez que se hace un cambio en el sistema?
- ¿Los individuos que cambian el sistema confían en que la documentación es la correcta?

5.13.6 Verificar la Actualización de la Documentación con el Usuario

Los usuarios finales deben preguntarse si la documentación para el sistema estáactualizada. Si los usuarios finales no están familiarizados con la documentación,necesitarán que se seleccione una muestra pidiendo piezas específicas de ladocumentación. La documentación seleccionada debe ser familiar para el usuario final demanera que se les pueda dar partes representativas del documento y pedir que validen siestán actualizadas o no.

VI. GLOSARIO

BUGS

Falla, error o defecto de un software.

DESARROLLO EN CASCADA

Se trata de un modelo lineal, posible en aquellos proyectos donde los requerimientos están clara y altamente definidos. También se conoce como ciclo de vida clásico. Las actividades siguen un flujo en secuencia.

ERP (Enterprise Resource Planning)

Sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa.

PMI (Project Management Institute)

Es una asociación profesional sin fines de lucro dedicada a promover las buenas prácticas en gestión de proyectos

PMBOOK (Project Management Body of Knowledge)

Estándar en la gestión de proyectos desarrollado por el Project Management Institute.

PROCESO

Conjunto de actividades mutuamente relacionas o que interactúan, las cuales transforman elementos de entrada en resultados o elementos de salida.

PROCEDIMIENTO:

Método o sistema estructurado para ejecutar algunas cosas. Acto o serie de actos u operaciones con que se hace una cosa

CLIENTE

El grupo (organización, proyecto o individuo) responsable por aceptar el producto o por autorizar el pago. El cliente es externo al proyecto, pero puede o no ser externo a la compañía.

STAKEHOLDERS

Personas que tienen un fuerte relación o interés, directa o indirectamente, con el proyecto, ya sea porque proporcionan datos al proyecto o porque reciben datos del mismo,

MODELO DE CAPACIDAD DE MADUREZ

Un modelo que contiene los elementos esenciales de procesos eficaces para una o más disciplinas y que describe un camino evolutivo de mejoramiento de procesos inmaduros.

CMMI (Capability Maturity Model Integration),

Modelo de madurez y mejora continúa de procesos de las organizaciones dedicadas a la fabricación y desarrollos de software

NIVEL DE MADUREZ

Grado de mejoramiento de procesos a través de un grupo predefinido de áreas de proceso in el cual todas las metas han sido cumplidas.

NIVEL DE CAPACIDAD

Logro del mejoramiento de procesos dentro de un área de proceso en particular. Un nivel de capacidad está definido por las prácticas genéricas y específicas de cada área de proceso.

DISCIPLINA

Dentro del CMMI, las áreas de conocimiento disponibles para seleccionar un modelo del CMMI (por ejemplo: ingeniería en sistemas, ingeniería de software, etc.). La estructura del CMMI permite que se integren más áreas de conocimiento en un futuro.

USABILIDAD:

Es el nivel con el que un producto se adapta a las necesidades del usuario y puede ser utilizado por los mismos para lograr unas metas con efectividad, eficiencia y satisfacción en un contexto especifico de uso.

PORTABILIDAD:

Conjunto de características que determinan la capacidad del software para ser transferido de un entorno de operación a otro. Se divide en las subcaracteríticas adaptabilidad, facilidad de instalación, coexistencia, reemplazo.

MANTENIMIENTO:

Esfuerzo requerido para implementar cambios en los productos de software ya puestos en producción u operación.

FIABILIDAD:

Grado en que el sistema responde bajo las condiciones definidasdurante un intervalo de tiempo dado.

FUNCIONALIDAD

Grado en que las necesidades asumidas o descritas sesatisfacen.

ASEGURAMIENTO DE CALIDAD

Todas las actividades planificadas y sistemáticasnecesarias para aportar la confianza suficiente en que un producto o serviciocumplirá con unos requisitos dados de calidad.

FACTORESDE LA CALIDAD DE SOFTWARE

Características de un producto de software a través delas cuales se describe y evalúa su calidad.

FALLAS

Manifestación de un defecto en el software, terminación de la habilidad funcionaldurante la ejecución del software

METRICAS DE CALIDAD:

Método cuantitativo utilizado para determinar el valor de unasubcaracterística de un producto de software.

PASOS:

Secuencia de acciones requeridas para desempeñar una función.

USUARIO

Persona que va utilizar el producto de software.

VII. BIBLIOGRAFIA

Fuentes de Internet

- http://www.ii.uam.es/
- www.sei.cmu.edu/CMMI/cmmi.html
- http://www.navegapolis.net/

Libros

- Guía de los Fundamentos de la Dirección de Proyectos, Tercera Edición (Guía del PMBOK®)
- Ingeniería de Software, Enfoque Practico (5ta Edición) Roger S. Pressman.
- CMMI® for Development, Version 1.2
- William E. Perry (2000). Effective Methods for Software Testing. Wiley Computer
- Barry W. Boehm (1981). Software Engineering Economics. Prentice Hall PTR.