UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
EAP. DE INGENIERÍA DE SISTEMAS

Estándares de calidad para pruebas de software

TESIS para optar el Título Profesional de: INGENIERO DE SISTEMAS

AUTORES
Daniel Rolando Valdivia Espinoza
Eduardo Geonias Valdivia Espinoza
ASESOR: Jorge Díaz Muñante

LIMA- PERÚ 2005

DEDICATORIA:

A nuestros padres María y Segundo y a nuestro hermano Miguel por sus apoyos constantes.

RESUMEN

ESTÁNDARES DE CALIDAD PARA PRUEBAS DE SOFTWARE

Bch. Daniel Rolando Valdivia Espinoza
Bch. Eduardo Geonias Valdivia Espinoza

Junio – 2005

Asesor : **Jorge Díaz Muñante**
Grado : **Lic. Ciencias de la Computación**

El presente trabajo se enfoca en las pruebas de software como un proceso de vital importancia para asegurar la calidad del mismo, es por eso que el proceso de pruebas como tal debe estar estandarizado y normado al interior de una organización que desarrolle software propio o para terceros, esta estandarización, cuyo objetivo no es más que conseguir la calidad en el proceso, se logra aplicando un modelo que permita una mejora continúa y un escalamiento progresivo a niveles de excelencia. CMM e ISO son los referentes más importantes, pero existen modelos como TPI y TMM que se enfocan exclusivamente en el proceso de pruebas de software, sobre estos modelos se realizará un análisis que permita al final definir un proceso de pruebas general para cumplir con los objetivos del Nivel 2 de TMM.

Palabras Clave:

Bootstrap, Calidad, CMM, CMM-SW, Pruebas, Software, TMM, TPI.

ABSTRACT

QUALITY STANDARDS FOR TESTS OF SOFTWARE

Bch. Daniel Rolando Valdivia Espinoza
Bch. Eduardo Geonias Valdivia Espinoza

June – 2005

Adviser : **Jorge Díaz Muñante**
Degree : **Lic. Computer Sciences**

The present work is focused in the software tests like a process of vital importance to assure your quality, it is for that reason that the process of tests as such a debit side to be standardized to the interior of an organization that develops own software or it stops third, this standardization whose objective is not more than to get the quality in the process, it is achieved applying a model that allows an improvement it continues and a progressive scaling at excellence levels. CMM and ISO are the relating ones more important, but models exist as TPI and TMM that are focused exclusively in the process of software tests, on these models they will be carried out an analysis that allows at the end to define a process of tests to fulfill the objectives of the Level 2 of TMM.

Keywords:

Bootstrap, CMM, CMM-SW, Quality, Software, Test, TMM, TPI.

INTRODUCCIÓN

En las últimas dos décadas las aplicaciones de software se han hecho cada vez más importantes en las labores cotidianas de las personas independientemente de cualquier factor diferenciante. Con las mejoras en las tecnológicas en los equipos de computación y el auge del uso de Internet ahora las aplicaciones que en sus inicios eran de soporte o ayuda ahora se han convertido en complejas y vitales a todo nivel, sobre todo para las organizaciones es que llegan a ser críticas y soportan en muchos casos la mayor parte de las operaciones de negocio.

Dada la importancia que el software ha adquirido en la actualidad ha dejado ya de ser una actividad artesanal, para convertirse en una actividad programada, planificada y con un ciclo de vida definido lo que otorga un carácter formal, sin embargo uno de los aspectos más descuidados en este gran proceso de desarrollo es el de las pruebas, a pesar de ser el medio por el cual se puede asegurar la calidad del producto de software. El proceso de pruebas no es un proceso menor en el ciclo de desarrollo de software, al contrario ello es tan importante como los demás, por lo que es necesario conocer lo que implica un proceso de pruebas y establecer para el un modelo que asegure que las pruebas aplicadas tengan la calidad suficiente como para asegurar que el producto final tendrá calidad también.

Actualmente los modelos de calidad aplicados al proceso de software como ISO o metodologías de madurez como CMM y CMM-SW no enfocan de manera especial el procesos de pruebas por lo que

1. PLANTEAMIENTO DEL PROBLEMA.

1.1. Definición del Problema

Un problema persistente en el proceso de desarrollo de software es el aseguramiento de calidad del mismo, cumplir con los requisitos del usuario en la mayoría de los casos no es suficiente, para ello hace falta definir un proceso de pruebas serio, uniforme para todos los proyectos, es decir, estandarizado y que incluya las mejores prácticas de pruebas siempre bajo un modelo que asegure la mejora continua.

Lo que se necesita es desarrollar y aplicar un modelo basado en metodologías o procedimientos estándares para el planeamiento, especificación y ejecución de pruebas de software, que permita alcanzar criterios de aceptación estándares para que las organizaciones desarrolladoras de software alcancen un lugar competitivo en la industria.

Tradicionalmente en nuestro medio las actividades de pruebas pasan casi desapercibidas, pues no se crean casos de pruebas que permitan garantizar la calidad del software, la ausencia de una orientación clara en la planificación del proyecto y de políticas organizacionales que apoyen esta proceso debido al desconocimiento o inaplicabilidad de algunos modelos de calidad aumentan el riesgo de producir software que inmediatamente reporta continuos errores o fallas graves. No se trata de sólo invertir más tiempo, lo que se necesita es una metodología que defina actividades y responsabilidades, naturalmente esta tarea no se puede realizar de manera improvisada sino que es un proceso gradual, aquí es donde CMM y todos los modelos desprendidos de el pueden ayudar.

1.2. Justificación e Importancia

La necesidad de garantizar la alta calidad del software ha aumentado las horas de trabajo de los procedimientos de pruebas respecto a los de análisis, diseño y programación. Hoy en día se calcula que la actividades en un proceso de pruebas serio representan más del 50% del costo de un software, ya que requiere un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico. Pese a su enorme impacto en el costo y tiempo de desarrollo, es una fase que muchos de los involucrados aún no consideran.

Es por esto que la intención de esta tesina es rescatar la importancia de las estándares y metodologías para las pruebas de software que van a garantizar su calidad.

1.3. Limitaciones y Alcances

La presente tesina enfoca el análisis en definir la importancia de la calidad en el software, conceptos previos en los primeros capítulos ayudan a aclarar los temas relacionados y justifican la investigación, más adelante se analiza el proceso de pruebas de software como medio de aseguramiento de calidad por excelencia, se realiza un breve estudio de los estándares y modelos de madurez más usados y sus deficiencias para con los procesos de pruebas, finalmente se analizan dos de los modelos específicos para pruebas de software, y a partir del más aceptado de ellos, para este caso TMM, se hace la propuesta de un modelo de procesos de pruebas con el objetivo de alcanzar el Nivel 2 de TMM, en tal sentido el modelo de pruebas planteado satisface todos los objetivos de madurez que plantea este nivel, se asume que los demás procesos relacionados con el ciclo de vida de desarrollo de software también están por lo menos en un nivel 2 de CMM o de CMM-SW o que se tiene un sistema de calidad implementado.

2. OBJETIVOS

2.1. Objetivo General

- Definir un modelo para el proceso de pruebas de software que alcance los requerimientos del Nivel 2 de TMM (Definición de Fase), aplicando las directivas establecidas por este modelo de madurez, y las mejores prácticas de pruebas y aseguramiento de calidad de software.

2.2. Objetivos Específicos

- Justificar un modelo general y apropiado al proceso de pruebas de software.
- Analizar las metodologías particulares aplicadas a los procesos de pruebas de software.
- Justificar la importancia del procesos de pruebas en el ciclo de vida del software, como un medio de aseguramiento de calidad.

3. CONCEPTO DE CALIDAD Y CALIDAD EN EL SOFTWARE.

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 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 W. Shewart y E. Deming, 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, Japón adoptó el concepto de Gestión Total de la Calidad que

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, se enfocó la consecución de la calidad como una medida de competitividad de las organizaciones, la globalización de finales del siglo pasado hizo que las organizaciones hicieran 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 ello la estrategia que adopta una organización es muy importante, el 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.

La Tabla Nº 1 muestra la evolución histórica de los conceptos de calidad en los últimos 100 años con sus características más sobresalientes.

Característica	Inspección (Antes de 1930)	Control Estadístico de la Calidad (1930 – 1950)	Aseguramiento de la Calidad (1950 – 1980)	Gestión Estratégica de la Calidad (1980 – Actualidad)
Interés	Detección	Control	Coordinación	Impacto estratégico
Enfoque	Detección de problemas	Solución de problemas	Prevención de problemas	Competitividad
Énfasis	Uniformidad del producto	Uniformidad del producto con inspección reducida	Ciclos completos de producción	Necesidades de los mercados y los clientes
Métodos	• Calibración • Medición	Herramientas y técnicas estadísticas	Sistemas de Información	• Planeación estratégica • Metas claras • Compromiso de toda la organización
Responsable	Departamento de Inspección	Departamentos de Ingeniería y Manufactura	Todas las áreas	Todas las áreas con fuerte liderazgo de la alta dirección
Orientación y Proyección	Inspección	Control	Desarrollo	Gestión

Tabla Nº 1: Evolución Histórica de los Enfoques de Calidad

3.2. El 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 en inglés) ha publicado hasta hoy varios estándares relacionados con la calidad en general y también de manera particular con la calidad de software, de todas las normas ISO publicadas, ISO 9000 corresponde al estándar de gestión de calidad ampliamente aceptado y que ha sido la base para otras normas más específicas, ISO 9000 [ISO9000, 2000] conceptualiza la calidad como "*el grado en el que un conjunto de características inherentes cumple con los requisitos*" a su vez para la norma ISO 8402 [ISO8402, 1994] calidad se define 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 la expresión "*necesidades expresadas o implícitas*" se refiere a que las necesidades pueden ser definidas por un contrato o son inherentes a la funcionalidad del producto.

Otro estándar muy importante pero aplicable sólo a su región es el Estándar Industrial Japonés (JIS, por sus siglas en inglés) z8110-1981 [JIS, 1981] establece el concepto de calidad como "*La totalidad de características o rendimiento, que puede ser usado para determinar si un producto cumple o no su aplicación prevista o intencionada* ".

3.3. Sistema 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*".

3.3.1. Principios Básicos de un Sistema de Calidad

Se puede decir que actualmente los sistemas de calidad están enfocados en cinco principios básicos, donde todos tiene participación por igual y son imprescindibles para los objetivos de calidad planteados por cualquier organización:

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

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

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

d. ***Comunicación y Reconocimiento;*** los resultados de la aplicación de las políticas de calidad deben ser comunicados a todos los miembros de la organización, los logros particulares y los más destacados deben ser reconocidos y estimulados, de este modo se

crea una conciencia del valor que posee el trabajo en la generación de calidad y a la vez se estimula a realizar las cosas cada vez mejor.

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.

3.4. La Calidad en el Software

Para entender el concepto de calidad aplicado al software es necesario detenerse un instante a entender que es el software como producto, y las implicancias que se desprenden de la manera particular en que es desarrollado.

Para nadie que este 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].

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 [IEEE, 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 Comisión Internacional Electrotécnica (IEC) editaron de manera conjunta la norma internacional ISO/IEC 9126 [ISO/IEC-9126, 1991] que define calidad de software como "*la totalidad de características de un producto de software que le confiere la capacidad de satisfacer necesidades explícitas e implícitas*", al respecto, se debe agregar que las necesidades explícitas son los alcances y los objetivos propuestos por quienes producen y desarrollan el software, por lo tanto son factores relativos a la calidad durante el proceso de desarrollo del software y solamente 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 los desarrolladores, a estas necesidades se les llama comúnmente calidad de uso.

Como aporte adicional en 1993 Pressman [Pressman, 1993] definió la calidad 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*".

3.5. 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 para los que fue desarrollado, pero surge la pregunta: ¿cómo asegurar de manera eficiente si un software cumple con los requisitos explícitos e implícitos para los que fue creado? Y casualmente esa es la principal dificultad que se enfrentan las empresas desarrolladoras de software al intentar asegurar la calidad de sus productos, el software como ya se ha dicho, posee características muy diferentes a los productos comunes, normalmente en la manufactura de cualquier producto una de las medidas de calidad esta dada por la capacidad del producto a

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

A continuación algunas de las formas más usuales de aseguramiento de calidad de software:

3.5.1. Métricas de Calidad de Software

Durante el proceso de evaluación de la calidad por lo general se hace referencia 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 un concepto de métrica muy genérico pero que también es aplicado al software, pero sin embargo las métricas suelen ser muy subjetivas, en la mayoría de los casos sólo miden algunas de las características de la calidad del software y dejan muchas cosas importantes sin ser medidas, suelen ser aplicadas para realizar mediciones que se basan en la codificación del software, cantidad de líneas de código, comentarios, documentación del código, etc.

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

3.5.2. Verificación y Validación de Software

Una aspecto importante a tener en cuenta son las verificaciones y validaciones como parte del ciclo de vida en el proceso de desarrollo de software, estas se utilizan para detectar fallas durantes las etapas que conforman este ciclo, es decir, durante la especificación de los requisitos, en

el análisis y diseño, desarrollo, e implementación. La verificación y validación son en realidad un conjunto de procedimientos y actividades que basados en técnicas específicas y con la ayuda de herramientas aseguran durante el proceso de desarrollo que el software cumpla con el objetivo para el cual es construido.

En cada etapa se realizan pruebas de verificación y validación específicas que buscan detectar cuanto antes los defectos y corregirlos antes de pasar a la siguiente etapa. Estas verificaciones y validaciones al final no son más que pruebas de diferentes tipos acerca de las cuales se hará un acercamiento más detallado en un posterior capítulo.

3.5.3. Revisión de Software

El estándar ISO9000 – 2000 [ISO9000, 2000] indica de manera general que para conseguir los objetivos de calidad se disponen de los siguientes métodos:

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

Tabla Nº 2: Métodos de Revisión para los Objetivos de Calidad

Como se aprecia en la Tabla Nº 2, se definen objetivos de calidad basados en evaluación, verificación, validación y confirmación de cumplimiento con los requisitos, ISO9000 – 2000 define sendos métodos para asegurar su cumplimiento.

4. PRUEBAS DE SOFTWARE.

4.1. Conceptos

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un software. Las técnicas para encontrar problemas en un software son extensamente variadas y van desde el uso del ingenio por parte del personal ejecutor de las pruebas hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad. Pero de nada serviría conocer todas las técnicas de prueba de software, si un programa carece de documentación, el código es confuso, o no se han seguido los pasos para su planificación y desarrollo. El diccionario del IEEE define a las pruebas, caso de prueba y fallo de la siguiente manera:

4.1.1. Pruebas

"*Es una actividad en la cual un sistema o uno de sus componentes se ejecuta en dos o más circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto*" [IEEE, 1990]. Probar, por lo tanto, es el proceso de ejecutar un programa con el fin de encontrar errores o fallas.

4.1.2. Caso de prueba

Es un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular como, por ejemplo,

ejercitar un flujo de un programa o verificar el cumplimiento de un determinado requisito. También se puede "*referir a la documentación en la que se describen las entradas, condiciones y salidas de un caso de prueba*" [IEEE, 1990].

4.1.3. Fallo

"*Un fallo es la incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados*" [IEEE, 1990].

4.2. Procesos de Pruebas

En la actualidad la tendencia es iniciar los procesos de pruebas antes y dentro del proyecto, los especialistas responsables de esta actividad deben estar capacitados adecuadamente para cumplirla a cabalidad.

4.2.1. Iniciarlas antes y dentro del proyecto

Quiere decir que actualmente las especificaciones de pruebas se realizan al mismo tiempo que el diseño de software; lo que se propone es iniciar el análisis de las pruebas junto con el análisis del software, estos sondeos preventivos permitirán ejecutar las pruebas tan pronto como el software esté listo y con ello no sólo descubrir errores, sino evitarlos.

4.2.2. Capacitar a los especialistas responsables de las pruebas

Este punto se enfoca en crear conciencia acerca de la importancia de las pruebas y de la importancia que tiene un equipo de personas

específicamente dedicadas a esta actividad que puedan integrarse a un proyecto y sean responsables de su calidad.

Los objetivos actuales de las pruebas no sólo tienen que ver con corregir errores, sino con prevenirlos influyendo y controlando el diseño y desarrollo del software. Las pruebas deben ser modeladas y basadas en los requerimientos de la aplicación que se ha de construir; por tanto, en las especificaciones de software deben incluirse las especificaciones de pruebas, ambas deberán revisarse en conjunto, y en esta revisión deberá participar un especialista en pruebas.

Por otro lado, se debe reconocer que las pruebas son una especie de administrador de riesgos pues aunque se puede definir qué debe considerarse como un buen resultado, quizás lo obtenido no necesariamente sea el mejor.

Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del costo de un programa, ya que requiere un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, de este modo si el proceso de pruebas requiere mucho más que tiempo y dinero entonces necesita una verdadera metodología la cual exige herramientas y conocimientos destinados a optimizar esta tarea.

4.3. Principios Básicos de las Pruebas de Software

Los principios básicos de pruebas de software a menudo parecen obvios pero por lo general son siempre ignorados, lo cual genera serias consecuencias en el proceso. Entre los principios básicos de pruebas de software tenemos los siguientes:

a. ***La definición del resultado esperado a la salida del programa es una parte integrante y necesaria de un caso de prueba*;** muchos errores se originan por ignorar este principio. Si el resultado esperado

de un caso de prueba no ha sido predefinido, existe la posibilidad de que un resultado erróneo, se interprete como correcto.

b. ***Un programador debe evitar probar su propio programa;*** la detección de errores es un proceso "destructivo". Es por ello que es difícil para una persona adoptar la actitud mental necesaria para demostrar que lo que acaba de hacer está erróneo.

c. ***Un área de programación no debería probar sus propios programas;*** es el caso similar al anterior, aplicado a la conducta colectiva de un grupo.

d. ***Inspeccionar concienzudamente el resultado de cada prueba****;* este es probablemente el más obvio, pero, sin embargo, a menudo dejado de lado. Empíricamente se ha encontrado que muchos de los programadores fracasan en la detección de errores, aún cuando en los resultados de las pruebas eran claramente observables.

e. ***Examinar un programa para comprobar que no hace lo que se supone que debe hacer es sólo la mitad del problema;*** la otra mitad consiste en ver si el programa hace lo que no se supone que debe hacer. Esto significa que los programas deben ser revisados con respecto a efectos colaterales no deseados.

f. ***Evitar los casos de pruebas desechables a menos que el programa sea verdaderamente un programa desechable;*** una práctica muy frecuente es sentarse frente al computador e inventar casos e pruebas. Los casos de prueba deben ser reinventados con cada prueba que se realice, algo que generalmente no se hace, por lo que las pruebas siguientes no son tan rigurosas como la primera vez, esto significa que si se introdujo un error en una parte ya probada, rara vez se detecta.

g. ***No planear el esfuerzo con la suposición tácita de que no se encontrarán errores*;** este error se comete cuando se parte de la suposición de que la prueba es el proceso de mostrar que el programa funciona correctamente.

h. ***La probabilidad de encontrar errores adicionales en una sección del programa es proporcional al número de errores encontrados*;** este fenómeno tiene su base en comprobaciones empíricas, en efecto, los módulos de un programa (igualmente probados) que han presentado más errores tienen una probabilidad más alta de presentar otros errores.

i. ***La prueba de software no es una ciencia exacta*;** si bien existen métodos que ayudan en el proceso de la elaboración de casos de pruebas, ello requiere de todas maneras de una considerable cuota de creatividad.

4.4. Tipos de Pruebas de Software

Las pruebas de software se dividen en dos grupos: las *Pruebas Estáticas* (sin uso de computadoras), son básicamente manuales, utilizadas para la revisión del código y planeamiento de los objetivos y fases de las pruebas; y las *pruebas Dinámicas* (con uso de computadoras), que permiten probar aspectos como la funcionalidad, integración, resistencia al uso concurrente, etc. En la Figura Nº 1 se muestra una estructura jerárquica de los tipos de pruebas de software, a continuación se detallan los objetivos y actividades de cada uno de los tipos de pruebas.

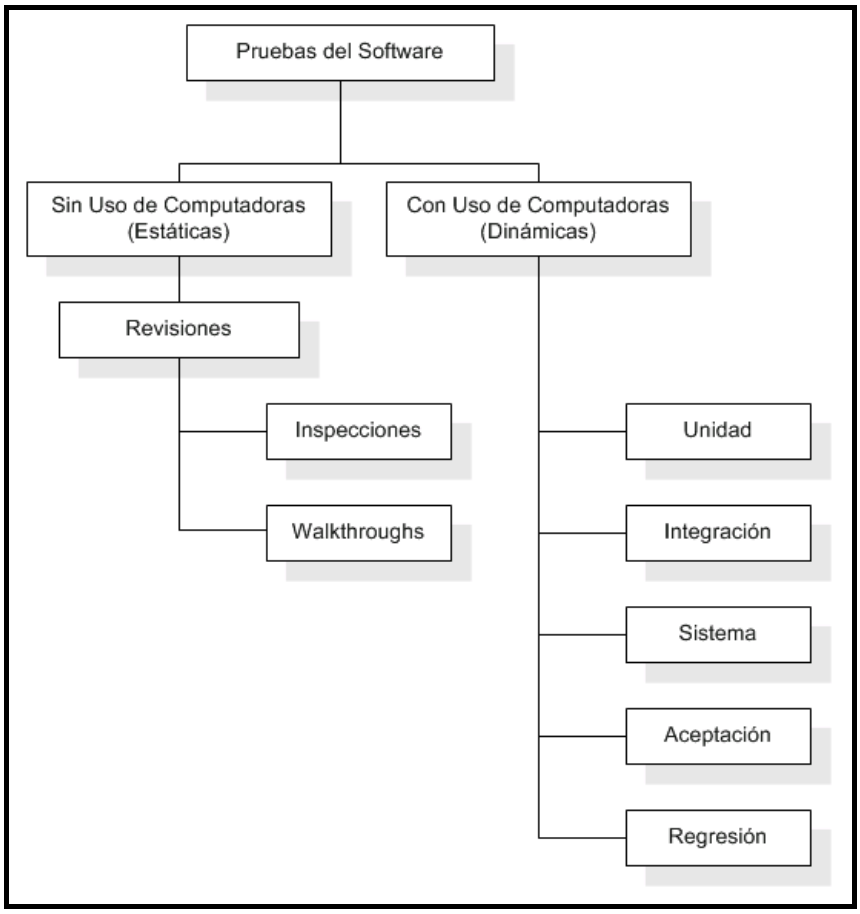


Figura Nº 1: Tipos de Pruebas de Software

4.4.1. Pruebas sin Uso de Computadoras o Estáticas

4.4.1.1. Inspecciones

Una inspección formal es un proceso con objetivos bien definidos, y como todo proceso, está conformado por fases, dichas fases son las siguientes:

a. ***Planificación***; se determina si el producto cumple los criterios para ser inspeccionado, se selecciona el equipo de trabajo, se asignan los roles, se distribuye el material, se coordina fecha, hora y lugar de la reunión de inspección.

b. ***Orientación inicial*;** es una fase opcional. El equipo se familiariza con el tema en cuestión.

c. ***Preparación individual*;** cada miembro del equipo analiza el producto, detectando potenciales defectos que serán luego discutidos.

d. ***Reunión de inspección*;** el equipo revisa el producto para detectar, categorizar y registrar, pero no corregir, los defectos.

e. ***Corrección*;** se corrigen los defectos detectados en el producto.

f. ***Reinspección*;** el proceso recomienza cuando se detectaron gran cantidad de defectos.

g. ***Seguimiento*;** Se determina si los defectos detectados han sido corregidos y se verifica que no se han introducido nuevos defectos.

4.4.1.2. Walkthroughs

Es una técnica más aplicada en control de calidad que en pruebas, consiste en sentar alrededor de una mesa a los desarrolladores y a una serie de críticos, bajo las órdenes de un moderador. El método consiste en que los críticos leen el programa línea a línea y piden explicaciones de todo lo que no está suficientemente claro. Puede ser que simplemente falte un comentario explicativo, o que detecten un error auténtico o que simplemente el código sea tan complejo de entender o explicar que más vale que se rehaga de forma más simple. Para un sistema complejo pueden hacer falta muchas sesiones.

Esta técnica es muy eficaz para encontrar errores de naturaleza local pero falla estrepitosamente cuando el error deriva de la interacción entre dos partes alejadas del programa. Nótese que no se está ejecutando

el programa, sólo se hace un análisis como con una lupa, y de esta forma sólo se ve en cada instante un parte del listado del código fuente.

4.4.2. Pruebas con uso de Computadoras o Dinámicas

4.4.2.1. Pruebas de Unidad

Se focaliza en ejecutar cada módulo (o unidad mínima a ser probada, ejemplo: una clase) lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores. Busca asegurar que el código funciona de acuerdo con las especificaciones y que el módulo lógico es válido.

Características:

- Particionar los módulos en unidades lógicas fáciles de probar, por cada unidad hay que definir los casos de prueba (pruebas de caja blanca).
- Los casos de prueba deben diseñarse de forma tal que se recorran todos los caminos de ejecución posibles dentro del código bajo prueba; por lo tanto el diseñador debe construirlos con acceso al código fuente de la unidad a probar.

- Los aspectos a considerar son los siguientes: rutinas de excepción, rutinas de error, manejo de parámetros, validaciones, valores aceptados, valores límites, rangos, mensajes posibles.

Técnica:

- Comparar el resultado esperado con el resultado obtenido.
- Si existen errores, reportarlos.

4.4.2.2. Pruebas de Integración

Se focaliza en verificar que las interfaces entre las componentes de software funcionan correctamente, así como determinar el enfoque para avanzar desde un nivel de integración de los componentes al siguiente y las acciones a tomar cuando se descubren problemas

Características:

- Identificar errores introducidos por la combinación de programas probados unitariamente.
- Determina cómo la base de datos de prueba será cargada.
- Verificar que las interfaces entre las entidades externas (usuarios) y las aplicaciones funcionan correctamente. Verificar que las especificaciones de diseño sean alcanzadas.
- Determina el enfoque para avanzar desde un nivel de integración de los componentes al siguiente.
- Describe cómo verificar que las interfaces entre los componentes de software funcionan correctamente.
- Decide qué acciones tomar cuando se descubren problemas.

Técnica:

- Utilizar la técnica top-down. Se empieza con los módulos de nivel superior, y se verifica que los módulos de nivel superior llaman a los de nivel inferior de manera correcta, con los parámetros correctos.

- Utilizar la técnica down-top. Se empieza con los módulos de nivel inferior, y se verifica que los módulos de nivel inferior llaman a los de nivel superior de manera correcta, con los parámetros correctos.

4.4.2.3. Pruebas de Sistema

Las pruebas del sistema deben enfocarse en requisitos que puedan ser tomados directamente de casos de uso, reglas y lógica del negocio. El objetivo de estas pruebas es verificar el ingreso, procesamiento y recuperación apropiado de datos, y la implementación apropiada de las reglas de negocio. Este tipo de pruebas se basan en técnicas de caja negra, esto es, verificar el sistema (y sus procesos internos), la interacción con las aplicaciones que lo usan vía GUI y analizar las salidas o resultados. Esencialmente, el encargado de la prueba intenta hacer fallar el programa. Algunos tipos de pruebas de sistema importantes para sistemas basados en software son:

a. ***Prueba de recuperación*****;** es una prueba del sistema que obliga al fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente. Si la recuperación es automática hay que evaluar la corrección de la reinicialización, de los mecanismos de recuperación de datos y del rearranque. Si la recuperación requiere la intervención humana, hay que evaluar los tiempos medios de reparación para determinar si están dentro de los límites aceptables.

b. ***Prueba de seguridad*****;** intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de accesos inapropiados.

c. ***Prueba de resistencia*;** consiste en realizar pruebas que demanden recursos en cantidad, frecuencia o volúmenes anormales. Esto se logra a través de:

- Casos de prueba que requieran el máximo de memoria o de otros recursos.
- Casos de prueba que puedan dar problemas con el esquema de gestión de memoria virtual.
- Casos de prueba que produzcan excesivas búsquedas de datos residentes en disco.

d. **Prueba de rendimiento.** Esta diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Esta prueba se da durante todos los pasos del proceso de prueba, incluso al nivel de unidad, se debe asegurar el rendimiento de los módulos individuales a medida que se realizan las pruebas de caja blanca.

Técnica:

- Ejecutar cada caso de prueba, flujo básico o función utilizando datos válidos e inválidos, para verificar que:

  - Los resultados esperados ocurren cuando se utiliza un dato válido.
  - Los mensajes de error o de advertencia aparecen en el momento adecuado, cuando se utiliza un dato inválido.
  - Cada regla de negocios es aplicada adecuadamente.

4.4.2.4. Prueba de Aceptación

La prueba de aceptación es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción. La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar como el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Basado en esta prueba el cliente determina si acepta o rechaza el sistema.

Estas pruebas están destinadas a probar que el producto está listo para el uso operativo, suelen ser un subconjunto de las Pruebas de Sistema y sirven para que el usuario pueda validar si el producto final se ajusta a los requisitos fijados, es decir, si el producto está listo para ser implantado para el uso operativo en el entorno del usuario.

Técnica:

- Realización de los documentos de planes de prueba de aceptación y especificación de los mismos, basados en los criterios de aceptación del cliente.

- Los casos prueba de aceptación han de ser planificados, organizados y formalizados de manera que se determine el cumplimiento de los requisitos del sistema. Para la realización de estas pruebas se necesita disponer de los siguientes documentos:

    - Especificación de requisitos del sistema.
    - Manual de usuario.
    - Manual de administrador.

4.4.2.5. Prueba de Regresión

En esta prueba se vuelve a probar el sistema a la luz de los cambios realizados durante el mantenimiento o desarrollo de la nueva versión del sistema buscando efectos adversos en otras partes.

Técnica:

- Realizar una nueva corrida de casos de prueba previos.

- Se requiere de políticas para ser creada la prueba de regresión y decidir qué casos de prueba incluir, para probar eficientemente.

- La prueba de viejas funcionalidades es más importante que la de nuevas funcionalidades.

- Aquellos casos de uso (y los casos de prueba asociados) que descubren defectos tempranamente deben ser incluidos en la prueba de regresión.

5. MODELOS Y ESTÁNDARES INTERNACIONALES DE PROCESO DE SOFTWARE.

Dentro de los modelos y estándares de procesos de software que vamos analizar tenemos los siguientes:

- Capability Maturity Model for Software
- Estándar ISO/IEC 15504
- Bootstrap

5.1. CMM-SW

Paulk, Curtis, Chrissis y Weber (1993) definen el modelo de madurez de capacidad (*Capability Maturity Model for Software* – CMM-SW) como un modelo que establece los niveles por los cuales las organizaciones de software hacen evolucionar sus definiciones, implementaciones, mediciones, controles y mejorías de sus procesos de software.

Además, el CMM-SW permite la definición del grado de madurez de las prácticas de gestión y de ingeniería de software de las organizaciones, de esta manera, se puede determinar cómo madura una determinada organización y cuales son las acciones de mejora prioritarias para sus procesos de software.

El enfoque inicial del CMM-SW ha sido el proceso de software. Sin embargo, se puede encontrar aplicaciones del modelo en otros campos como por ejemplo CMM para personas (*People CMM*), CMM para ingeniería de

sistemas (*Systems Engineering CMM*), CMM para la gestión de productos integrados (*Integrated Product Management CMM*) y otros.

5.1.1. Estructura de CMM-SW

El esquema del modelo CMM-SW esta compuesto por cinco niveles de madurez de acuerdo con la capacidad del proceso de software y definidos por los objetivos de los procesos que, cuando se satisfacen, permiten evolucionar al próximo nivel ya que uno o más componentes importantes del proceso de software han sido estabilizados. De esta manera, los niveles de madurez ayudan a las organizaciones a definir prioridades para sus esfuerzos de mejora. En la Figura Nº 2, se presentan los cinco niveles de madurez del proceso de software.

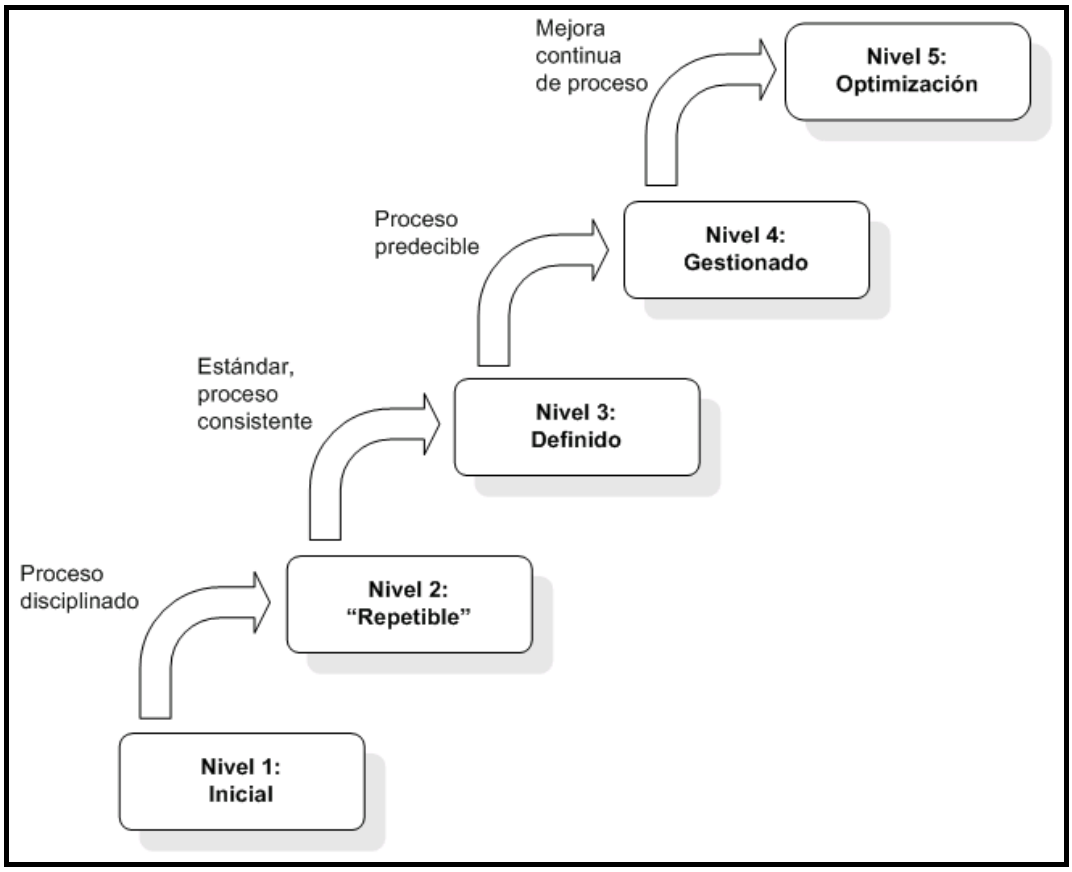


Figura Nº 2: Los niveles de madurez de CMM.

- ***Nivel 1 - Inicial:*** se caracteriza como *ad hoc* o caótico. Pocos procesos son definidos.

- ***Nivel 2 - "Repetible" o repetición:*** se caracteriza como disciplinado. Se establecen procesos básicos de gestión.

- ***Nivel 3 - Definido:*** se caracteriza como estándar y consistente.

- ***Nivel 4 - Gestionado:*** se caracteriza como predicable. Hay una preocupación en la medición detallada de la calidad del proceso de software y del producto.

- ***Nivel 5 - Optimización:*** se caracteriza como mejora continua a partir de la realimentación (*feedback*) cuantitativa.

5.1.2. Componentes de CMM-SW

Cada nivel de madurez tiene una estructura interna (véase Figura Nº 3) compuesta por los siguientes componentes:

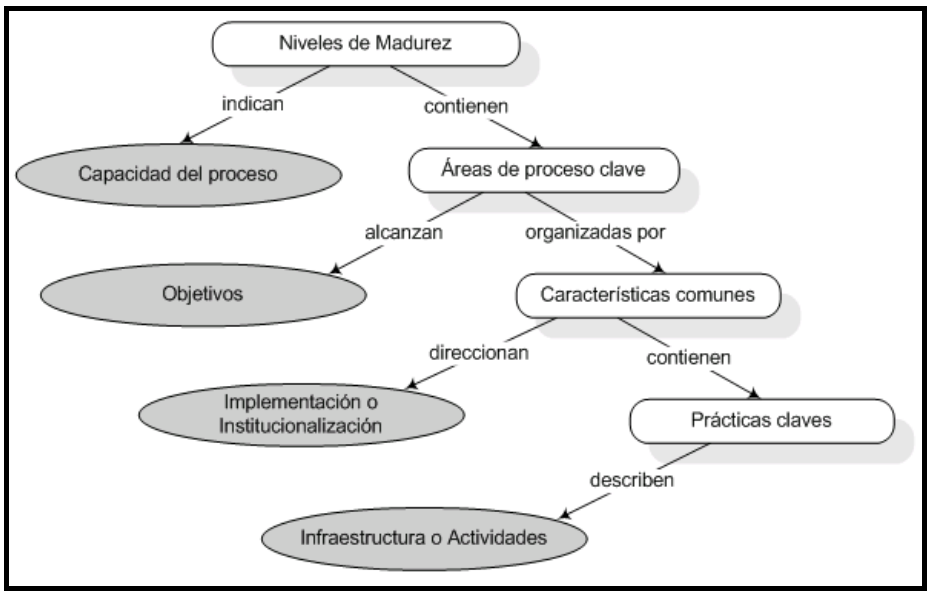


Figura Nº 3: Estructura interna del CMM

5.1.3. Estructura de CMM-SW

5.1.3.1. Nivel de madurez

Representa un indicador evolutivo que permite alcanzar la madurez del proceso de software.

5.1.3.2. Áreas de proceso clave (Key Process Areas - KPA)

Son las subestructuras de cada nivel que indican las áreas a las que una organización debería dirigir su atención con el propósito de mejorar su proceso de software. Se asigna cada conjunto de KPA a un nivel de madurez (excepto al nivel uno), como se muestra en la Figura Nº 4, el CMM-SW no detalla todas las áreas de proceso involucradas en el desarrollo y mantenimiento de software, sino que se enfoca en las áreas clave que contribuyen en la mayoría de las capacidades de proceso.

5.1.3.3. Objetivos

Este componente delimita las KPA a través de la definición de su alcance, límites e intenciones. Los objetivos, por lo tanto, determinan las restricciones que deben ser superadas por la organización para que ésta pueda alcanzar mejores niveles de madurez.

5.1.3.4. Características comunes

Son atributos que indican si la implementación e institucionalización de las KPA son eficaces, repetidas y duraderas, así se presentan cinco características comunes:

- El compromiso en desempeñar las acciones que garantizan el establecimiento del proceso.
- La habilidad en desempeñar dichas acciones.
- Las actividades desempeñadas para implementar las KPA.
- La medición, el análisis del estado y eficacia de las actividades desempeñadas.
- La implementación para la verificación de las actividades desempeñadas respecto a los procesos establecidos.

5.1.3.5. Prácticas clave

Éstas describen la infraestructura y actividades que contribuyen para la implementación e institucionalización efectiva de la KPA.

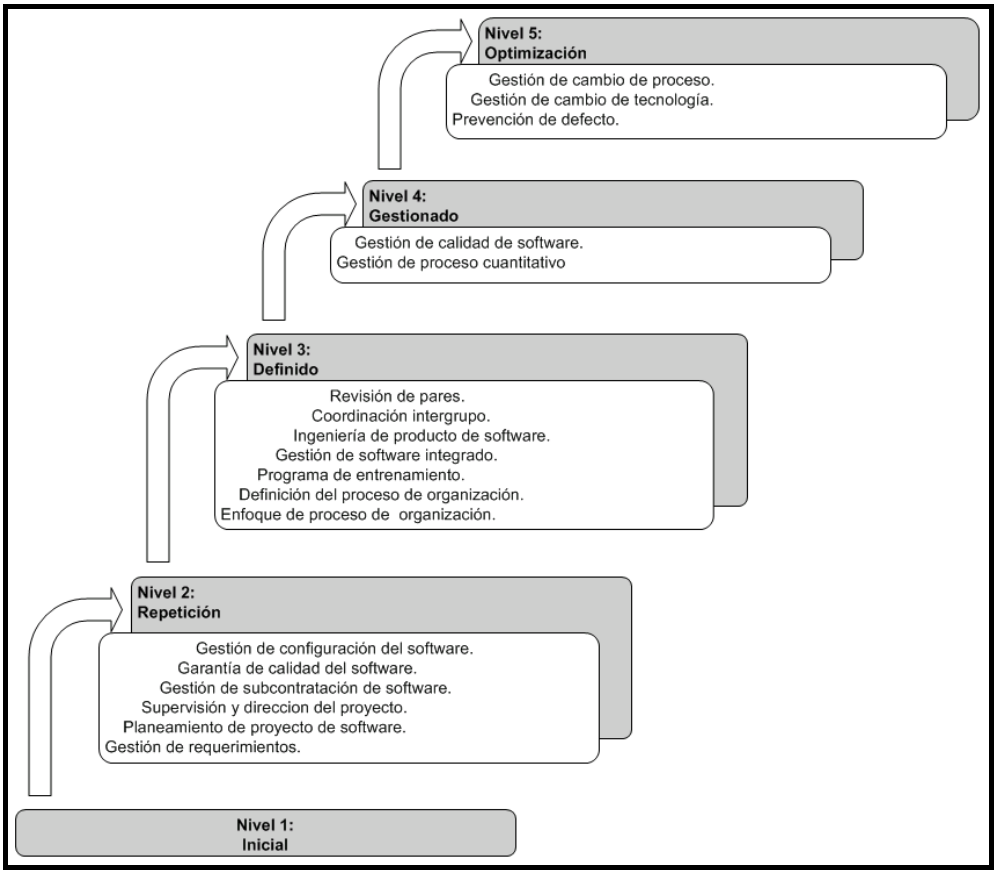


Figura Nº 4: Las KPA representadas en cada uno de los cinco niveles de madurez del proceso de software del CMM

El CMM-SW es un mapa para la definición de pasos que una organización necesita realizar para ir desde los procesos caóticos a procesos de mejora continua. Las evaluaciones basadas en CMM-SW utilizan el cuestionario del *Software Engineering Institute* (SEI) como un instrumento de evaluación. Su enfoque se dirige a temas relacionados con procesos y sus cuestiones se basan en los objetivos de las KPA del CMM-SW.

5.2. Estándar ISO/IEC 15504

La Organización Internacional para Estandarización (*International Organization for Standardization*) y la Comisión Electrotécnica Internacional (*Internacional Electrotechnical Commission*) han presentado el estándar ISO/IEC 15504 (1998), el cual se manifiesta a partir del consenso internacional en la definición de un estándar de dominio público para la evaluación de procesos de software.

El desarrollo del estándar ISO/IEC 15504 se establece a partir del proyecto SPICE5 que tiene como objetivo garantizar una ruta de desarrollo rápida y solicitar las opiniones de los expertos más importantes del mundo.

5.2.1. Estructura del ISO/IEC 15504

El estándar ISO/IEC 15504 consiste en dos dimensiones: la dimensión de proceso y la dimensión de capacidad.

5.2.1.1. Dimensión de proceso

Se caracteriza por los propósitos de proceso (por ejemplo: objetivos de medición esenciales de un proceso) y el resultado esperado del proceso (por ejemplo: la indicación de su finalización exitosa). A esta dimensión se le atribuyen cinco categorías:

- Categoría de proceso cliente-proveedor.
- Categoría de proceso de ingeniería.
- Categoría de proceso de soporte.
- Categoría de proceso de gestión.
- Categoría de proceso de organización.

5.2.1.2. Dimensión de capacidad

Esta dimensión consiste en un conjunto de atributos interrelacionados que ofrecen los indicadores de medición necesaria para gestionar un proceso y mejorar la capacidad de desempeñar un proceso. Esta dimensión compuesta de seis niveles tiene características evolutivas similares a las del CMM-SW (véase Figura Nº 5):

- ***Nivel 0 - Incompleto:*** se caracteriza por un incumplimiento general para lograr el propósito del proceso.

- ***Nivel 1 - Proceso desempeñado:*** se caracteriza por el logro de manera general del propósito del proceso.

- ***Nivel 2 - Proceso gestionado:*** se caracteriza por la identificación de la calidad aceptable con definición de tiempos y recursos. Los productos del trabajo están de acuerdo con los estándares especificados y los requerimientos.

- ***Nivel 3 - Proceso consolidado:*** se caracteriza por la gestión y el desempeño del proceso usando el proceso estándar basado en principios estables de ingeniería de software.

- ***Nivel 4 - Proceso predecible:*** se caracteriza por la consistencia de su desempeño en la práctica con límites de

control definidos para alcanzar sus objetivos de proceso definido.

- ***Nivel 5 - Proceso optimizado:*** se caracteriza por la optimización del desempeño del proceso para encontrar las necesidades de negocio actuales y futuras, y por el grado de repetición que el proceso alcanza encontrando sus objetivos de negocio definidos.

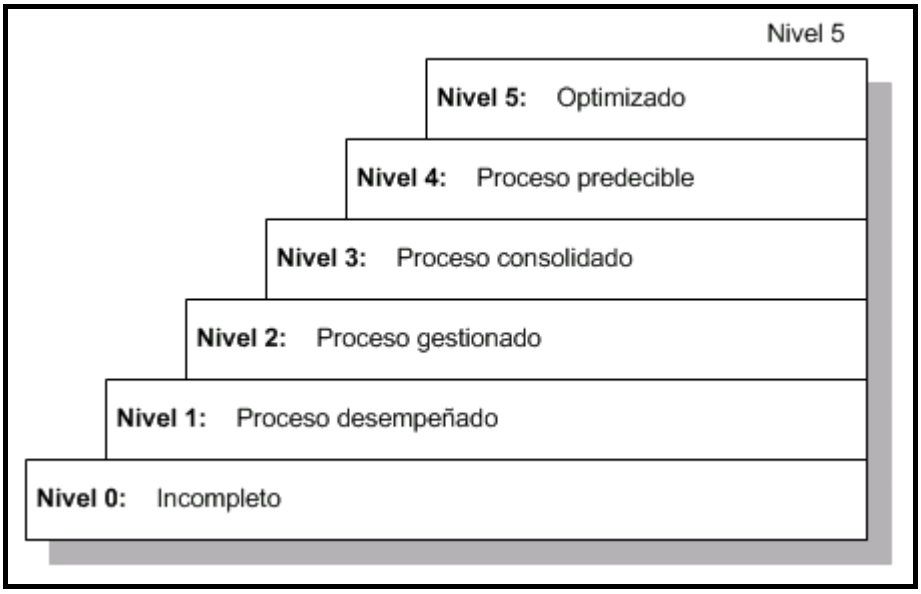


Figura Nº 5: Los niveles de capacidad de proceso según ISO/IEC 15504

5.3. BOOTSTRAP

El Bootstrap es una metodología de evaluación que mejora la capacidad de los procesos de desarrollo de software. Además, el modelo Bootstrap describe el proceso de evaluación, determina donde se encuentra una organización respecto a los niveles de madurez, identifica los puntos fuertes y debilidades de la organización, y ofrece un guía para el proceso de mejora.

Este método es uno de los resultados del proyecto ESPRIT 5441 apoyado por la Comunidad Europea. El Bootstrap ha considerado como punto de partida

varios estándares y metodologías como por ejemplo CMM, ISO/IEC 15504 y los estándares de ingeniería de software (*Software Engineering Standards* - SES) de la Agencia Espacial Europea (*European Space Agency* - ESA). El modelo Bootstrap se basa en la tríada Organización, Metodología y Tecnología (OMT) presentada en la Figura Nº 6.

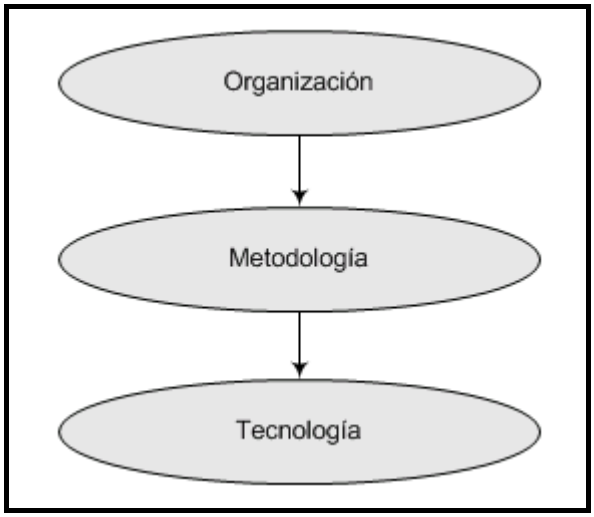


Figura Nº 6: La tríada Bootstrap

5.3.1. Arquitectura del Modelo de Procesos de BOOTSTRAP

Usando la tríada presentada en la Figura Nº 6 como punto de partida, se define una arquitectura en forma de árbol que identifica las categorías de proceso, las áreas de proceso, los procesos y las mejores prácticas, ver la Figura Nº 7 para el detalle de la arquitectura.

En general, se utiliza el proceso de evaluación del método Bootstrap para medir el estado actual de la práctica de desarrollo de software dentro de una organización. La evaluación se basa en un cuestionario que está formado por listas de verificación (*checklists*) de acuerdo con unos atributos clave, de esta manera se calcula la media agregada por atributos clave y consecuentemente el nivel de madurez, basándose en los cinco niveles de madurez del CMM.

El Bootstrap cubre la unidad de producción de software enfocando sus actividades no sólo en la evaluación sino también en el planeamiento de acción.

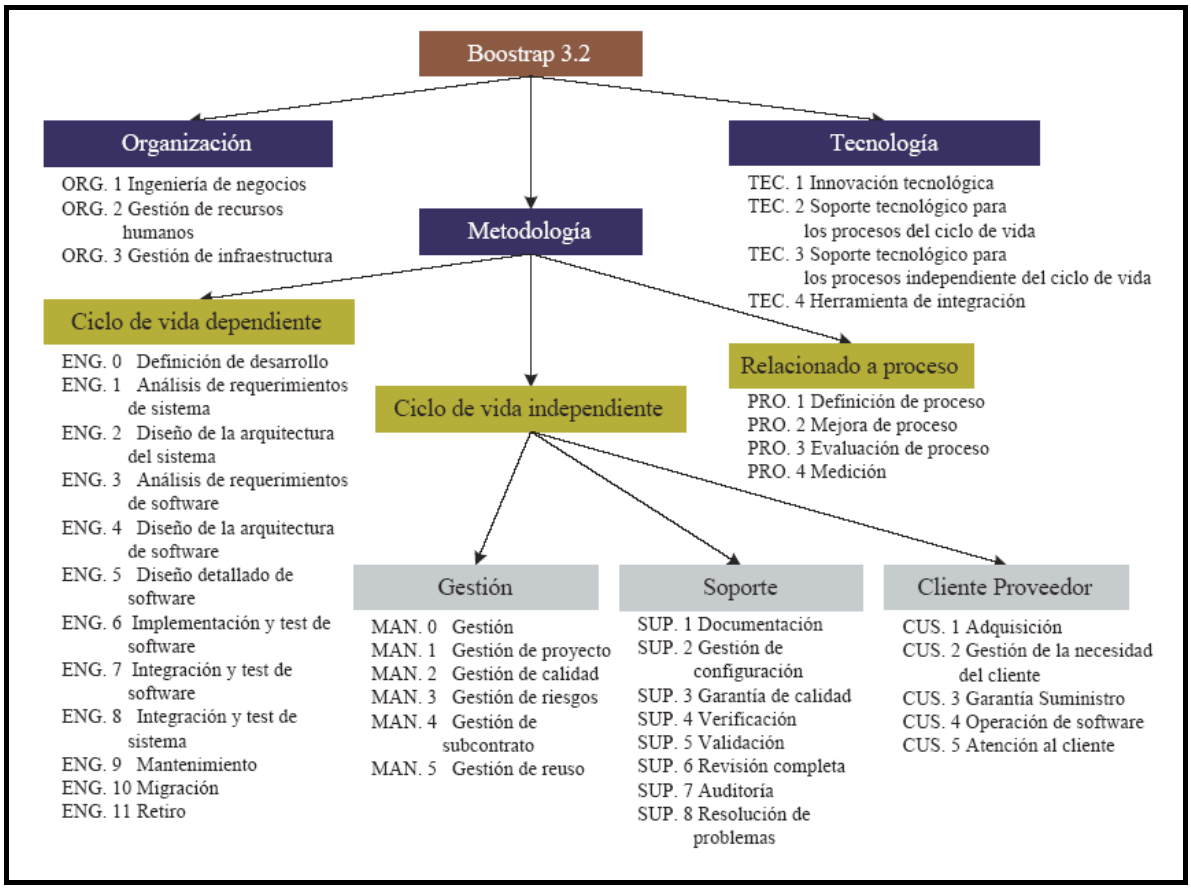


Figura Nº 7: Arquitectura de proceso Bootstrap versión 3.2

6. PROCESO DE PRUEBAS EN LOS MODELOS ESTÁNDARES.

El estándar ISO/IEC 15504 define un esquema integrado para el desarrollo y aplicación de pruebas a lo largo del ciclo de vida del software, así como el modelo Bootstrap. Por otra parte, estos modelos aportan referencia técnica a las pruebas y cómo éstos dan soporte a los demás procesos.

Además el estándar ISO/IEC 15504 y el modelo Bootstrap, bajo un punto de vista operacional, mantienen un entendimiento común con los clientes, ya que el producto o el proceso está de acuerdo con sus requerimientos. Sin embargo, no se identifica una especialización sistemática de los procesos de pruebas (por ejemplo en las pruebas de verificación, validación y usabilidad). En el caso del CMM-SW, no se contemplan adecuadamente los aspectos relacionados con las pruebas en las KPA, tampoco se identifican las prácticas de pruebas como una herramienta o mecanismo de mejora del proceso. De esta manera, en el CMM-SW no han sido identificados algunos conceptos, como por ejemplo el de madurez de las pruebas.

6.1. Modelos Específicos del Proceso de Pruebas

La idea propuesta por diversos investigadores y profesionales de cómo transformar la tarea de pruebas en un proceso de costes efectivos, ha crecido equitativamente respecto a la importancia de dicha tarea. La evolución de los modelos específicos de proceso de pruebas aumenta satisfactoriamente la calidad de dichos procesos y los coloca en una posición relevante dentro de la ingeniería de software.

Como se ha comentado, los modelos y estándares de evaluación de proceso de software ofrecen herramientas para establecer niveles de madurez de desarrollo y mantenimiento del software. Sin embargo, ha sido constatado que estos modelos y estándares no dirigen adecuadamente sus enfoques a los procesos de pruebas. En este sentido se presentan a continuación dos modelos de evaluación y mejora de procesos de pruebas:

- Modelo de Mejora del Proceso de Pruebas (*Test Process Improvement* - TPI).
- Modelo de Madurez del Pruebas (*Testing Maturity Model* - TMM).

7. TPI – TEST PROCESS IMPROVEMENT.

El Modelo de Mejora del Proceso de Pruebas (TPI - Test Process Improvement) se ha desarrollado partiendo del conocimiento y la experiencia de las Pruebas de Control de Software. El modelo TPI es un medio de ayuda para mejorar el proceso de pruebas.

El modelo permite visualizar el nivel de madurez del proceso de pruebas dentro de la organización. Partiendo de este criterio, el modelo ayuda a definir pasos de mejora graduales y controlados. El modelo es visualizado tal como se aprecia en la Figura Nº 8.

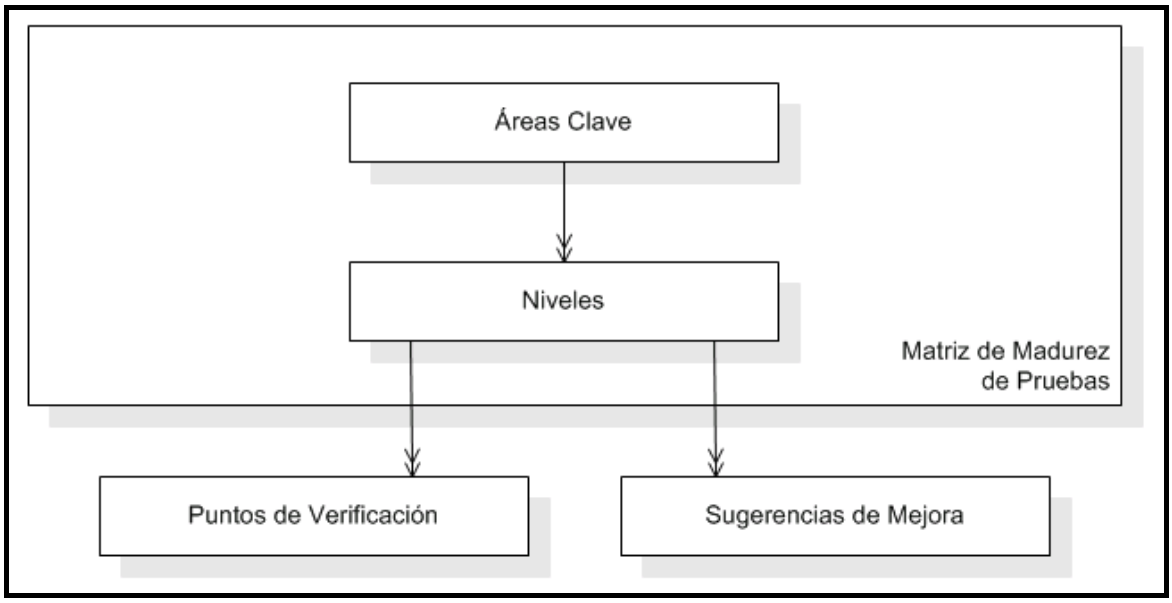


Figura Nº 8: Modelo TPI

7.1. Áreas Clave

En cada proceso de pruebas hay ciertas áreas que necesitan atención específica con el fin de lograr un proceso bien definido. Estas Áreas Clave constituyen la base para mejorar y estructurar el proceso de pruebas. El modelo TPI tiene 20 Áreas Clave. El alcance de la mejora del proceso de pruebas incluye normalmente las pruebas de caja negra, como son las pruebas de sistema y pruebas de aceptación, la mayoría de las Áreas Clave están dirigidas a ello, sin embargo, para mejorar procesos de pruebas más maduros, se debe prestar atención a las actividades de verificación y pruebas de caja blanca, como son las pruebas unitarias y las de integración, estas pruebas se incluyen como áreas clave separadas con el fin de prestar la atención adecuada a estos procesos.

A continuación se muestra una lista completa de áreas clave.

7.1.1. La estrategia de pruebas

Debe estar dirigida a encontrar los defectos más importantes con la mayor rapidez y el menor costo posible. En la estrategia de pruebas se determinan los defectos de calidad descubiertos al realizar las pruebas. Al involucrar más pruebas y medidas de detección se puede definir mejor la estrategia. Las duplicidades u omisiones involuntarias que tengan lugar entre diferentes pruebas pueden ser evitadas coordinando a los probadores y las actividades de pruebas, así como fijando el alcance de la prueba.

7.1.2. Modelo del Ciclo de Vida

Dentro del proceso de pruebas se distinguen algunas fases: planeamiento, preparación, diseño, ejecución y terminación. En cada fase se efectúan algunas actividades, para cada actividad se registran aspectos tales como: objetivo, entradas, proceso, conceptos, dependencias, técnicas y

herramientas, instalaciones y documentación. La importancia de contar con un modelo de ciclo de vida de desarrollo reside en tener un mejor control del proceso de pruebas, dado que las actividades pueden ser planeadas y controladas consistentemente.

7.1.3. Momento de Involucración

Aunque el momento de ejecución de la pruebas se inicia normalmente al concluir el diseño del software, el proceso de pruebas debe iniciarse mucho antes. La involucración temprana de las pruebas en el ciclo de vida del desarrollo de software ayuda a detectar los defectos con mayor antelación y/o con más facilidad, e incluso ayuda a prevenir defectos. Ello redunda en una mejor coordinación entre las pruebas además de reducir enormemente la ruta crítica de las pruebas.

7.1.4. Estimación y Planeamiento

Se necesita estimar y planear para definir las actividades que deben realizarse en cada momento y los recursos que serán necesarios. Estimar y planear son la base para ahorrar capacidad y para coordinar las actividades de prueba y las actividades del proyecto.

7.1.5. Técnicas de Diseño de Pruebas

Una técnica de diseño de pruebas se define como un modelo para derivar casos de prueba de la documentación. El uso de estas técnicas incrementa la visibilidad de la calidad y la cobertura de pruebas y conduce a una mayor reusabilidad de pruebas. En base a una estrategia de pruebas se utilizan diferentes técnicas de diseño de pruebas para obtener la cobertura del código de las partes de software cuyo alcance fue acordado.

7.1.6. Técnicas de Pruebas Estáticas

No todo puede ni requiere ser probado dinámicamente, por ejemplo, ejecutando los programas. A esta forma de validar productos sin ejecutar los programas o evaluar métricas de calidad específicas, se le llama Pruebas estáticas. Las listas de verificación y mecanismos similares son muy útiles en este caso.

7.1.7. Métricas

Las métricas son observaciones cuantitativas. Para el proceso de pruebas, medir el progreso y la calidad del software probado es muy importante, así como también lo son las métricas en estas áreas. Las métricas se utilizan para poder administrar el proceso de pruebas, para poder tener evidencia al momento de expresar una opinión, y también para comparar diferentes sistemas o procesos. Ayudan a contestar preguntas tales como: ¿Por qué el sistema A tiene mucho menos fallos en producción que el sistema B?, ¿Por qué el proceso de pruebas del sistema A puede ser ejecutado con mayor rapidez y con mayor alcance que el proceso del sistema B?.

En la mejora del proceso de pruebas, las métricas son esencialmente importantes para juzgar los resultados de ciertas acciones de mejora. Esto se hace midiendo antes y después de que la mejora sea implantada.

7.1.8. Herramientas de Prueba

La automatización del proceso de pruebas puede efectuarse de muy diversos modos. Como regla, la automatización sirve a alguna de las metas siguientes:

- Menor consumo de recursos.

- Menos consumo de tiempo.
- Mejor cobertura de pruebas.
- Mayor flexibilidad.
- Mayor o más rápida comprensión del estado del proceso de pruebas.
- Mayor motivación del personal de pruebas.

7.1.9. Entorno de Pruebas

La ejecución de la prueba tiene lugar en el llamado entorno de pruebas. Este entorno de pruebas consta de los siguientes componentes:

- Hardware.
- Software.
- Instalaciones de comunicaciones.
- herramientas para la creación y el uso de datos de prueba.
- Procedimientos.

El entorno debe quedar establecido para poder hacer pruebas de forma óptima. El entorno de pruebas influye en gran medida en la calidad, duración y costo del proceso de pruebas. Algunos aspectos importantes del entorno de pruebas son: roles y responsabilidades, control, disponibilidad suficiente y a tiempo, flexibilidad y representatividad de los entornos reales de producción.

7.1.10. Entorno de Oficina

El personal de pruebas necesita oficinas, escritorios, sillas, ordenadores, grabadoras, impresoras, teléfonos, etc. Un arreglo adecuado y a tiempo del entorno influye positivamente en la motivación del personal, y en la comunicación y eficiencia de la ejecución de las tareas de pruebas.

7.1.11. Compromiso y Motivación

El compromiso y la motivación del personal involucrado en las pruebas son condiciones indispensables para lograr un proceso de pruebas maduro. El personal involucrado en el proceso de pruebas incluye no solamente a los miembros del equipo de pruebas, sino también, entre otros, a líderes de proyectos y a los directivos.

El proceso de pruebas debe disponer del tiempo, dinero y recursos suficientes (cuantitativa y cualitativamente) para efectuar una buena prueba. La cooperación y comunicación con los otros miembros del proyecto da como resultado un proceso eficiente y una involucración temprana.

7.1.12. Funciones de Pruebas y capacitación

El personal de pruebas requiere de una cierta preparación. Se requiere una combinación de diferentes materias, funciones, conocimientos y habilidades. Por ejemplo, aparte de tener experiencia específica sobre pruebas, también se requiere conocimiento del sistema que está siendo probado, conocimiento de la organización y conocimiento general de automatización. También es importante contar con ciertas habilidades sociales. Para poder tener estas cualidades, es necesario ofrecer educación y capacitación.

7.1.13. Alcance de la Metodología

Para cada proceso de pruebas se utiliza cierta metodología o acercamiento, que consiste en actividades, procedimientos, estándares, técnicas, etc. Si estas metodologías difieren para cada proceso de pruebas en la organización, o si la metodología usada es demasiado genérica muchas cosas tienen que ser reinventadas una y otra vez. El objetivo de una organización es utilizar una metodología que sea lo suficientemente genérica

como para ser ampliamente aplicable, pero al mismo tiempo que sea lo suficientemente detallada para poder evitar las reinvenciones en cada nuevo proceso de pruebas.

7.1.14. Comunicación

En un proceso de pruebas la comunicación se lleva a cabo de diversos modos, tanto entre los probadores como grupo, como entre los probadores y otros miembros del proyecto, tales como el desarrollador, el usuario final, y el líder del proyecto. Algunos temas objeto de comunicación son la estrategia de pruebas, así como el progreso y la calidad del software bajo prueba.

7.1.15. Informes

Al finalizar cada una de las iteraciones y al completar el proceso de pruebas todos los datos recolectados como producto de las mismas deben ser registrados y comunicados con la finalidad de que puedan ser utilizados en un análisis posterior.

7.1.16. Manejo de Defectos

Aunque el manejo de defectos es de hecho responsabilidad del líder de proyecto, los probadores están altamente involucrados en ello. Una buena administración debería ser capaz de controlar el ciclo de vida de un defecto y crear diferentes informes. Estos informes se usan para prestar asesoramiento bien fundado sobre la calidad del software.

7.1.17. Administración del Testware (elementos de prueba)

El testware o los elementos de prueba deberían ser mantenibles y reutilizables, y por lo tanto, deben ser administrados. Aparte de los elementos de pruebas, también los productos de fases previas, tales como el diseño y la construcción, deben estar bien.

El proceso de pruebas puede verse seriamente interrumpido por la entrega de versiones erróneas de programas. Una buena administración de estos productos incrementa la factibilidad de pruebas y por ende la calidad del software.

7.1.18. Administración del Proceso de pruebas

Con el fin de administrar cada proceso y cada actividad son esenciales los cuatro pasos del llamado círculo de calidad de Deming: Planear, Hacer, Comprobar, Actuar. Un proceso de pruebas bien administrado es de la mayor importancia para efectuar la mejor prueba posible en la a veces turbulenta área de pruebas.

7.1.19. Revisión Estructurada

Revisión Estructurada en este contexto significa validar algunos conceptos tales como el diseño funcional. En comparación con las pruebas, la ventaja de la revisión estructurada es que ofrece la oportunidad de detectar defectos con prontitud. Esto lleva a que los costos en correcciones sean considerablemente más bajos. Además, realizar dicha revisión es relativamente simple dado que ningún programa debe ser ejecutado y no se requiere establecer ningún entorno de pruebas, etc.

7.1.20. Pruebas de Caja Blanca

Una prueba de caja blanca está definida como una prueba de las propiedades internas de un objeto, mediante el conocimiento de funciones internas. Estas pruebas son ejecutadas por los desarrolladores. La prueba unitaria y la de integración son pruebas bastante conocidas de caja blanca. Al igual que la Revisión Estructurada, estas pruebas son efectuadas en las etapas iniciales del ciclo de vida antes de llegar a las pruebas de caja negra. Por otro lado, las pruebas de caja blanca son relativamente baratas porque se requiere menor comunicación y porque el análisis es más sencillo, la persona que detecta los defectos es regularmente la misma persona que hace las reparaciones. Además, se prueban menos objetos.

7.2. Niveles

La forma en que están organizadas las áreas clave dentro de un proceso de pruebas determina la madurez del proceso. Es obvio que no toda área clave tendrá la misma atención ni profundidad: cada proceso de pruebas tiene sus puntos fuertes y débiles. Con el fin de permitir una visión del status de cada área clave, el modelo proporciona Niveles (de A a B a C). Como promedio, hay tres niveles por cada área. Cada nivel superior (donde C es mayor que B, y B es mayor que A) es mejor que su nivel previo en términos de tiempo (más rápido), dinero (menor costo), y calidad (mejor). Al usar niveles podemos evaluar sin ambigüedades la situación prevaleciente del proceso de pruebas. También incrementa la posibilidad de recomendar mejoras graduales.

Cada nivel conlleva el cumplimiento de ciertos requisitos para el área clave. Los requisitos Checkpoints de un cierto nivel también incluyen los requisitos de niveles previos: un proceso de pruebas de nivel B satisface los requisitos de ambos niveles A y B. Si un proceso de pruebas no satisface los requisitos para el nivel A, se considera que se encuentra en el nivel más bajo y, por tanto, en un nivel indefinido para dicha área en particular.

Tabla Nº 3: Características de las Áreas Clave de Proceso para Cada Nivel del TPI

Niveles / Área Clave	A	B	C	D
Estrategia de Pruebas	Estrategia de Pruebas para una sola prueba.	Estrategia de Pruebas combinada para pruebas de caja negra.	Estrategia de Pruebas combinada para pruebas de caja negra y pruebas de caja blanca o evaluación.	Estrategia de Pruebas combinada para todas las pruebas y actividades de evaluación.
Modelo del Ciclo de Vida	Planeamiento, Diseño, Ejecución	Planeamiento, Preparación, Diseño, Ejecución, Terminación.		
Momento de Involucración	Al terminar la especificación	Al inicio de la especificación	Al iniciar la definición de requerimientos.	Al inicio del Proyecto.
Estimación y Planeamiento	Estimación y Planeamiento Elemental	Estimación y Planeamiento fundada estadísticamente		
Técnicas de Diseño de Pruebas	Técnicas Informales	Técnicas Formales		
Técnicas de Pruebas Estáticas	Pruebas en base a entradas	Lista de verificación (checklists)		
Métricas	Estadísticas del Proyecto (producto)	Estadísticas del Proyecto (Proceso)	Estadísticas del Sistema	Estadísticas de la Organización
Herramientas de Prueba	Herramientas de Planeamiento y Control	Herramienta de ejecución y análisis, y de pruebas	Automatización Integral de Pruebas	
Entorno de Pruebas	Entorno administrado y controlado	Pruebas en el entorno mas conveniente	Mejora de entorno a solicitud.	
Entorno de Oficina	Entorno de oficina adecuado y a tiempo			
Compromiso y Motivación	Asignación de presupuesto y tiempo.	Las pruebas integradas en la organización del proyecto.	Ingeniería de Pruebas.	
Funciones de Pruebas y Capacitación	Gerente de Pruebas y Probadores.	Soporte (Metodológico, Técnico, Funcional), Control	Aseguramiento de la Calidad Interno	

Alcance de la Metodología	Específico del Proyecto	Para toda la organización, genérica	Organización, optimización (Investigación y Desarrollo)	
Comunicación	Comunicación Interna	Comunicación del Proyecto (defectos, cambios, control)	Comunicación en la organización	
Informes	Defectos	Progreso (status de pruebas y productos), actividades (costes + tiempo, metas), defectos con prioridades	Riesgo y Consejo, incluyendo estadísticas	SPI consejo
Manejo de Defectos	Administración Interna de Defectos	Administración Extendida de Defectos, Informes Flexibles	Administración de Defectos del Proyecto	
Administración de Testware (elementos de prueba)	Administración y control interno de los conceptos de pruebas	Administración y control externo de la base y de los objetivos de pruebas.	Testware reutilizable	Capacidad de Rastreo: desde los requerimientos hasta los casos de prueba.
Administración del Proceso de pruebas	Planear, Hacer	Planear, Hacer, Verificar, Reaccionar	Comprobar, Reaccionar en la organización	
Revisión Estructurada	Técnicas de Revisión Estructurada	Estrategia de Revisión Estructurada		
Pruebas de Caja Blanca	Ciclo de Vida: Planear, Diseñar, Ejecutar	Técnicas de Diseño de Caja Blanca	Estrategia de Pruebas de Caja Blanca	

7.3. Puntos de Verificación (Checkpoints)

Con el fin de determinar los niveles, el modelo TPI se basa en un instrumento de medición objetiva. Los requisitos para cada nivel están definidos en forma de Puntos de Verificación o Checkpoints, los cuales son preguntas que necesitan ser respondidas afirmativamente para poder calificar a dicho nivel. A partir de las listas de verificación se puede evaluar un proceso de pruebas y se puede determinar para cada área clave el nivel alcanzado. A medida que se considera mejorada cada área clave, los puntos de verificación son acumulables, así, para poder clasificar para el nivel B, el proceso de pruebas necesita responder afirmativamente tanto a los puntos de verificación del nivel B como del nivel A.

7.4. Matriz de Madurez de Pruebas

Después de determinar los niveles para cada área clave se debe dirigir la atención hacia cuáles son los pasos de mejora que hay que realizar. Esto se debe a que no todas las áreas clave y niveles tienen la misma importancia, por ejemplo, una buena estrategia de pruebas (nivel A del área clave “Estrategia de Pruebas”) es más importante que una descripción de la metodología de pruebas utilizada (nivel A del área clave “Alcance de la Metodología”). Además de estas prioridades, existe una dependencia entre los niveles de diferentes áreas clave. Antes de poder recibir estadísticas de los defectos encontrados (nivel A del área clave “Métricas”), el proceso de pruebas tiene que calificar para el nivel B del área clave “Manejo de Defectos”. También hay dependencias entre muchos niveles y áreas clave.

Por lo tanto, todos los niveles y áreas clave están interrelacionados en una Matriz de Madurez de Pruebas. Se ha concebido como una buena manera de expresar las prioridades internas y dependencias entre niveles y áreas clave. El eje vertical de la matriz indica áreas clave, el eje horizontal de la matriz muestra escalas de madurez. En la matriz, cada nivel está relacionado con cierta escala de madurez de pruebas, resultando así 13 escalas de madurez de

pruebas. Las celdas abiertas entre diferentes niveles no tienen significado por sí mismas, pero indican que el lograr una mayor madurez para un área clave está relacionado con la madurez de otras áreas clave. No existe graduación entre niveles, mientras que un proceso de pruebas no esté clasificado enteramente como nivel B, permanecerá en el nivel A.

Escala / Área Clave	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Estratégia de Pruebas		A					B				C		D	
Modelo del Ciclo de Vida		A			B									
Momento de Involucración			A				B				C		D	
Funciones de Pruebas y Capacitación				A			B			C				
Alcanc de la Metodología					A						B			C
Comunicación			A		B							C		
Informes		A			B		C					D		
Manejo de Defectos		A				B		C						
Administración de Testware (elementos de prueba)			A			B				C				
Administración del Proceso de pruebas		A		B								C		
Revisión Estructurada							A			B				
Pruebas de Caja Blanca					A		B		C					
Estratégia de Pruebas		A					B				C		D	
Modelo del Ciclo de Vida		A			B									
Momento de Involucración			A				B				C		D	
Estimación y Planeamiento				A							B			
Técnicas de Diseño de Pruebas		A		B										
Técnicas de Pruebas Estáticas					A		B							
Métricas						A			B			C		D
Herramientas de Prueba					A			B			C			
Entorno de Pruebas				A				B						C
Entorno de Oficina				A										
Compromiso y Motivación		A										C		

Tabla Nº 4: Matriz de madurez de Pruebas

El propósito principal de la matriz es mostrar los puntos fuertes y débiles del actual proceso de pruebas y ofrecer una ayuda a la hora de determinar la prioridad de las acciones de mejora. Una matriz con datos reales ofrece a todos los participantes una visión clara de la situación actual del proceso de pruebas.

Además, la matriz ayuda a definir y seleccionar propuestas de mejora. La matriz se maneja de izquierda a derecha, de tal manera que las áreas claves poco maduras sean mejoradas en primer lugar. Como consecuencia de la dependencia entre niveles y áreas clave, la práctica ha demostrado que las áreas clave con alta escala de madurez, rodeadas de áreas clave con escalas de madurez media o baja no son una buena inversión, por ejemplo, no sirve de mucho tener una gestión de los defectos muy avanzada, si no se usa para realizar análisis e informes. Siempre que no se aparten del modelo, se permiten desviaciones, pero deberán existir razones de peso que lo justifiquen.

7.5. Sugerencias de Mejora

Las acciones de mejora pueden definirse en función de los niveles superiores que se deseen alcanzar. Para alcanzar un nivel superior, los Puntos de Verificación proporcionan gran ayuda, además de éstos, el modelo tiene otros medios de soporte para la mejora del proceso de pruebas: Las “Sugerencias de Mejora”, que son diferentes tipos de ideas o consejos que ayudan a alcanzar un cierto nivel de madurez de pruebas. A diferencia de los Puntos de Verificación, no es obligatorio usar las Sugerencias de Mejora y cada nivel está provisto de varias Sugerencias de Mejora.

7.6. Aplicación del Modelo TPI

El proceso de mejora de pruebas es similar a cualquier otro proceso de mejora. La Figura Nº 9 muestra las diferentes actividades de un proceso de mejora. Estas actividades se van a comentar a continuación, poniendo especial atención a aquellas áreas en las que se puede utilizar el modelo TPI.

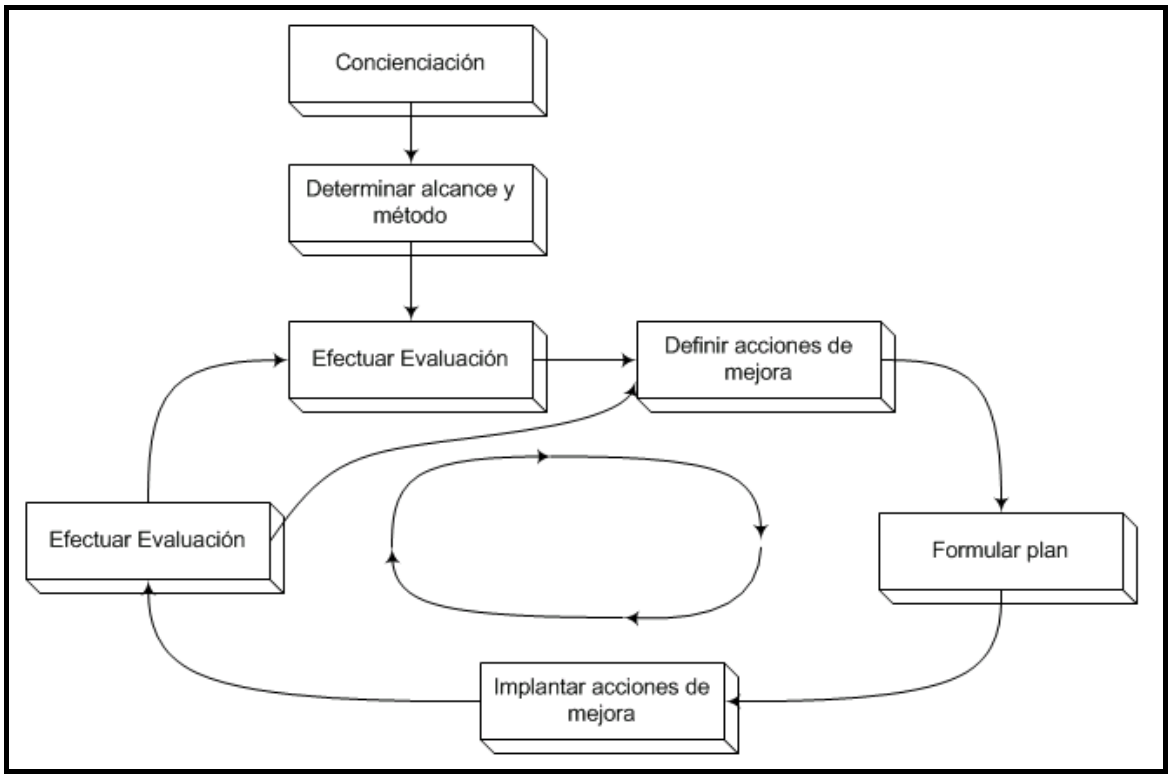


Figura Nº 9: Aplicación del Modelo TPI

7.6.1. Concienciación

La primera actividad de un proceso de mejora de pruebas es tomar conciencia de la necesidad de mejorar el proceso. Hablando genéricamente, la razón para mejorar el proceso de pruebas es que existen una serie de problemas referentes a las pruebas. Hay que resolver dichos problemas y la solución radica en una mejora del proceso de pruebas, esto también implica que las partes acuerdan mutuamente las líneas generales y se comprometan al proceso de cambio.

El compromiso no solo debe lograrse al inicio del proceso de cambio, sino que debe permanecer a lo largo de todo el proyecto. Esto requiere de un esfuerzo continuo.

7.6.2. Determinar el alcance y método

Determinar cuáles son las metas y el alcance de la mejora, esto quiere decir si es que se deben realizar las pruebas con mayor rapidez, a menor costo o mejor, determinar los procesos de prueba susceptibles de mejorar, y establecer de cuánto tiempo se dispone para hacer la mejora y cuánto esfuerzo se esta dispuesto a dedicarle.

7.6.3. Efectuar la Evaluación

En esta actividad se obtiene una evaluación de la situación actual. El uso del modelo TPI es una parte importante de la evaluación porque ofrece un marco de referencia que enumera los puntos fuertes y débiles del proceso de pruebas. Basado en entrevistas y documentación, los niveles por área clave del modelo TPI son examinados al utilizar los Puntos de Verificación, y se determinan cuáles son los puntos que se han alcanzado, y cuáles no se han alcanzado, o cuales se han alcanzado parcialmente.

La Matriz de Madurez de Pruebas se utiliza aquí para ofrecer una visión completa de la situación en que se encuentra el proceso de pruebas. Aquí se mostrarán las puntos fuertes y débiles del proceso de pruebas en forma de niveles asignados a las áreas clave, así como su posición relativa en la matriz.

7.6.4. Definir Acciones de Mejora

Las acciones de mejora se definen a partir de los objetivos de mejora establecidos, así como de los resultados de la evaluación. Estas acciones se determinan de tal manera que sea posible ir mejorando paso a paso.

El modelo TPI ayuda a establecer estas acciones de mejora. Los niveles de las áreas clave y la Matriz de Madurez de Pruebas ofrecen algunas

posibilidades para definir pasos de mejora de modo gradual. Dependiendo de los objetivos, el alcance, el tiempo disponible y los resultados de la evaluación, se puede decidir iniciar mejoras para una o más áreas clave. Para cada área clave seleccionada se puede decidir ir al siguiente nivel o, en casos especiales, incluso a un nivel superior. Adicionalmente, el modelo TPI ofrece un gran número de Sugerencias de Mejora que ayudan a alcanzar niveles superiores.

7.6.5. Formular el plan

Se diseña un plan detallado para implantar las acciones de mejora a corto plazo. En este plan se determinan los objetivos y se indican cuáles son las mejoras que deben implantarse y en qué momento debe hacerse con el fin de lograr los objetivos. El plan se dirige tanto a las actividades relacionadas con el contenido de las mejoras del proceso de pruebas como a las actividades generales necesarias para realizar el proceso de cambio en la dirección adecuada.

7.6.6. Implantar Acciones de Mejora

Se ejecuta el plan, debido a que durante esta actividad, las consecuencias del proceso de cambio tienen un mayor impacto, hay que prestar mucha atención a la comunicación. Sin duda, puede haber cierta resistencia al cambio, por lo que deberá afrontarse y discutirse abiertamente. Se debe medir hasta qué punto se han ejecutado las acciones y si han resultado con éxito. Una forma de hacerlo es mediante la llamada auto-evaluación, donde las personas involucradas revisan su propio proceso de pruebas usando el modelo TPI. Otra parte vital de esta fase es la consolidación. Se debe evitar que las acciones de mejora implantadas sean experimentadas una sola vez.

7.6.7. Efectuar la Evaluación

Se necesita verificar hasta qué punto las acciones implantadas han logrado los resultados esperados. En esta fase el objetivo es ver en qué medida las acciones fueron implantadas con éxito, así como evaluar en qué medida se alcanzaron los objetivos iniciales. A partir de estas observaciones se adoptará la decisión sobre la continuación del proceso de cambio.

8. TMM – TEST MATURITY MODEL.

Hacia Finales de 1996 en el Instituto de Tecnología de Illinois se creó el TMM o Test Maturity Model, el objetivo principal de TMM es conducir las evaluaciones y mejoras de los procesos de pruebas en las organizaciones que desarrollan software, En principio TMM esta guiado por el conjunto de los conceptos básicos que dieron origen al CMM, y esta compuesto por dos componentes principales: un conjunto de niveles de madurez y un modelo de evaluación, de este modo en el TMM se describe la posición que un determinado proceso de pruebas ocupa en la jerarquía de madurez de las pruebas.

8.1. Características Generales

Se debe considerar a las siguientes como características generales del Modelo TMM:

- Es un modelo complementario y compatible a CMM y por consecuencia a CMM-SW.
- Esta basado en una validación de la situación actual del proceso de pruebas a través de reglas claras y objetivas.
- Estimula la mejora continua de los procesos de pruebas de software.
- Es un modelo basado en las mejores prácticas de pruebas de software existentes en el mercado.

Al igual que CMM, el TMM posee 5 niveles, pero eso no quiere decir que exista alguna correspondencia, la Tabla Nº 5 muestra los 5 niveles de TMM con su descripción correspondiente.

Niveles	Descripción
1	Inicial
2	Definición de Fase
3	Integración
4	Gestión y Medición
5	Optimización, Prevención de defectos y Control de calidad

Tabla Nº 5: Niveles de TMM

A pesar de la aparente similitud de los niveles existentes entre el TMM y el CMM la correspondencia entre ambos modelos no es directa, pues la evolución en un modelo TMM suele ser más rápida, sin embargo su evolución sí esta ligada a la evolución en un modelo CMM, la Tabla Nº 6 muestra esta correspondencia en la evolución.

Niveles de TMM	Niveles Correlativos en CMM	Áreas claves de los procesos
1	1	
2	2	• Gerencia de requisitos • Gerencia de configuración • Planeamiento de proyectos de software
3	2	• Garantía de calidad de software • Supervisión de proyectos de software
	3	• Enfoque en los procesos de la organización • Definición de los procesos de la organización • Programas de entrenamiento
4	3	• Revisión por terceros
	4	• Gerencia de calidad de software • Gerencia cuantitativa del proceso
5	5	• Gerencia de evolución de los procesos • Gerencia de evolución de la tecnología • Prevención de defectos

Tabla Nº 6: Correspondencia entre la evolución de los niveles de TMM y CMM

TMM esta diseñado para mejorar los procesos de calidad de pruebas en una organización desde adentro y es usado por:

- Un equipo de aseguramiento interno para identificar la capacidad actual de pruebas.
- La alta dirección para iniciar una mejora en el programa de pruebas.

- Los Ingenieros de aseguramiento de calidad de software para desarrollar e implementar los planes del proceso de mejora.
- Los equipos de desarrollo para mejorar la eficacia de las pruebas.
- Los usuarios y clientes para definir su rol en el proceso de pruebas.

Debido al rol importante de las pruebas en el proceso de desarrollo de software y en la calidad del producto final y porque los modelos existentes tienen serias limitaciones en este sentido es que TMM se presenta como una alternativa de solución, los siguientes puntos respaldan los objetivos que se pretenden conseguir:

- ***Un conjunto de niveles que definen una jerarquía de madurez de pruebas*;** cada nivel representa un escenario en la evolución hacia un proceso de pruebas maduro. El movimiento hacia un nivel más alto implica que las prácticas menores continúan en su lugar.

- ***Un conjunto de objetivos y sub-objetivos de madurez para cada nivel;*** *(excepto para el nivel 1)*. Los objetivos de madurez identifican objetivos de mejora para las pruebas que deben ser definidos para conseguir la madurez en un determinado nivel. Los sub-objetivos definen el alcance, los límites, y la necesidad de logros para un nivel especial. También hay actividades, tareas, y responsabilidades (ATR) relacionadas con cada objetivo de madurez que son necesarios para respaldarlo.

- ***Un modelo de aseguramiento que consta de tres componentes*:**

  1) Un conjunto de preguntas relacionadas a los objetivos de madurez diseñadas para asegurar el actual proceso de pruebas.
  2) Un conjunto de pautas para instruir al equipo de aseguramiento.
  3) Un procedimiento de aseguramiento con los pasos para guiar al equipo de aseguramiento a través de los procesos de pruebas para la evaluación y mejora.

8.2. Requisitos Generales para el Desarrollo de TMM

Los requisitos generales para el desarrollo de TMM son los siguientes:

- El modelo debe ser aceptado por la comunidad desarrolladora de software de la organización y debe estar basado en los principios de ingeniería y prácticas de software.
- En los niveles de madurez más altos se debe ser lo suficientemente flexible como para adaptarse a las futuras mejores prácticas.
- El modelo también debe permitir el desarrollo de la madurez en los procesos de prueba en las fases intermedias, estructuradas para seguir un procesos natural de evolución hacia el nivel superior.
- Debe existir un mecanismo de soporte para el aseguramiento y validación de los procesos de pruebas.

8.3. Influencias en el Modelo TMM

Para satisfacer todos los requisitos del modelo TMM se tomaron los siguientes modelos como bases principales en su desarrollo:

8.3.1. CMM

De la misma manera que el CMM, el TMM usa el concepto de niveles de madurez para probar los procesos de evaluación y mejora. Los niveles de TMM tienen una base estructural como en el CMM, pero se le han añadido unos componentes llamados "Vistas críticas" para incluir los grupos clave necesarios para el crecimiento de madurez del proceso de prueba. Ambos modelos requieren que todas las capacidades cumplidas en el nivel más bajo sean incluidas en nivel inmediato superior. Para soportar el proceso de aseguramiento, el TMM usa el enfoque de evaluación de cuestionario - entrevista del CMM. Además de estas semejanzas estructurales, el TMM es visualizado como un complemento para el CMM, esto es esencial, ya que un

proceso de prueba maduro está en función de la madurez del proceso general, y la inversión de la organización en pruebas puede ser optimizado si las pruebas en algunas áreas de proceso pueden ser llevadas a cabo en paralelo.

8.3.2. El modelo de prueba evolutivo de Gelperin y Hetzel

Otra base de TMM es el modelo histórico de Gelperin y Hetzel (1988), utilizado para fundamentar la diferenciación histórica en los niveles de TMM. El modelo de Gelperin y lo Hetzel describe las fases y los objetivos de prueba a través de los años, así, el período inicial es descrito como "orientado a la depuración", durante este periodo las organizaciones desarrolladoras de software no tenían clara la diferencia entre pruebas y depuración, la prueba era una actividad ad-hoc relacionada con la depuración para eliminar los defectos de los programas, pero en hoy las pruebas han avanzado a un nivel "orientado a la prevención", por lo que las mejores prácticas de pruebas se realizan en el nivel 5 del TMM.

8.3.3. Las prácticas actuales de pruebas en la industria

Una visión general de las prácticas industriales también fue tomada como información importante para la definición de los niveles de TMM.

8.3.4. Las fases evolutivas del modelo mental de pruebas de Beizer

En el desarrollo de TMM también han combinado los conceptos asociados con el modelo evolutivo de Beizer, su influencia está basada en la premisa de que una organización de pruebas madura es construida sobre las destrezas, las habilidades, y actitudes de los individuos que trabajan dentro de ella.

Para el desarrollo de TMM se ha usado un enfoque sistemático sobre la base de las cuatro fuentes ya descritas, permitiendo satisfacer los requisitos. El resultado ha sido que TMM tiene un enfoque de desarrollo orientado hacia un modelo que es:

- Más exhaustivo en su estructura de niveles.
- Soportado por un modelo de aseguramiento adicional siempre que este sea bien definido.
- Bien definido y más fácil de entender y usar.
- Provee la cobertura más grande acerca de todos los aspectos relacionados con las pruebas.
- Mejor para respaldar el crecimiento en la madurez de los procesos de prueba.

8.4. Estructura de Niveles

TMM se caracteriza por tener cinco niveles de madurez para las pruebas con una base en objetivos, sub-objetivos, actividades, tareas, y responsabilidades. Esta base del modelo se muestra en la Figura Nº 10. Cada nivel implica un estado de madurez para una prueba específica, con excepción del Nivel 1 los objetivos de madurez que identifican áreas claves de proceso son mostrados en cada nivel.

Los objetivos de madurez identifican objetivos de mejora de pruebas que deben ser orientados para conseguir la madurez en ese nivel. Para ser colocada en un nivel, una organización debe satisfacer todos los objetivos de madurez de ese nivel. Cada objetivo de madurez es respaldado por uno o más sub-objetivos de madurez, que especifican objetivos menores y definen el alcance, los límites, y logros necesarios para un nivel especial. Los sub-objetivos de madurez son conseguidos a través de un conjunto de actividades, tareas y responsabilidades (ATR), estas son puestas en práctica con el fin de adaptar la organización a un nivel específico.

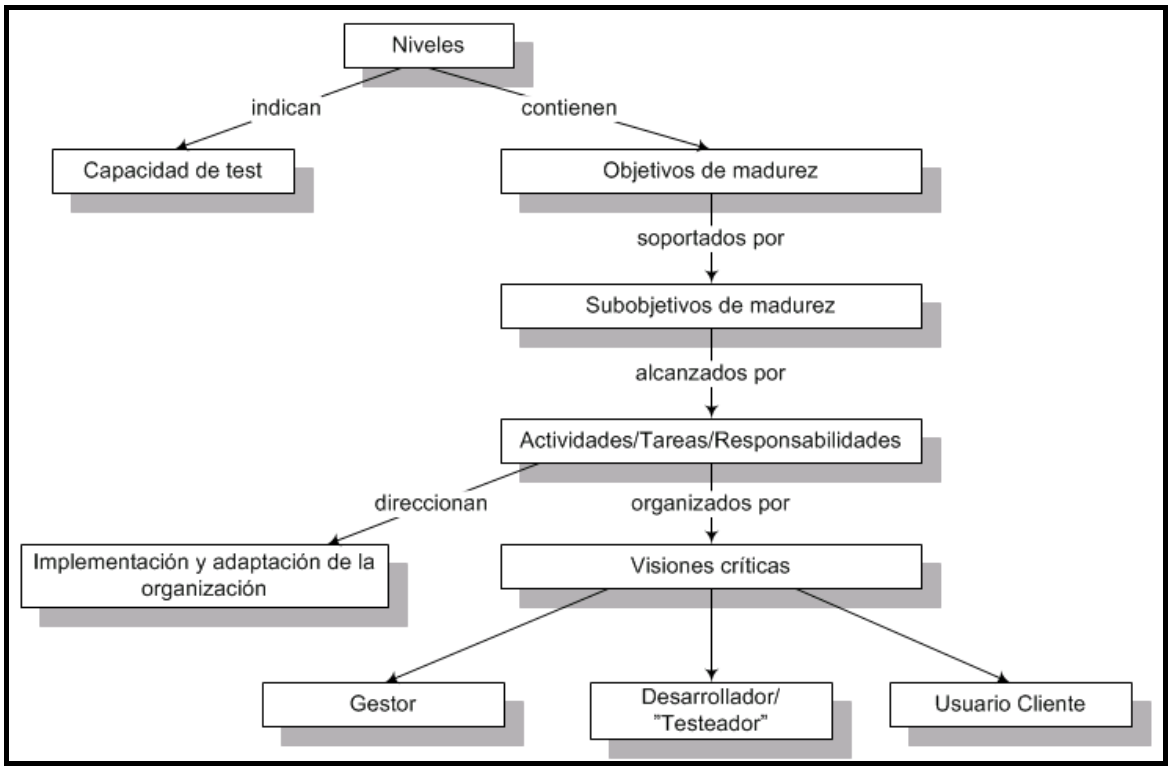


Figura Nº 10: Estructura base del modelo TMM

Las actividades y tareas son definidas como acciones que deben ser llevadas a cabo en un nivel en particular para mejorar la capacidad de pruebas. La responsabilidad para cumplir estas ATR es atribuida a los participantes clave en el proceso de pruebas: directores, desarrolladores/probadores, y usuarios/clientes, en el modelo se les conoce como "Las tres vistas críticas." La vista del director involucra asumir el compromiso y la capacidad de llevar a cabo las actividades y las tareas relacionadas con mejorar la madurez del proceso de pruebas. La vista del desarrollador/probador abarca las actividades técnicas y tareas que se deben realizar. La vista del usuario/cliente es definida como una opinión de ayuda o de soporte. Los desarrolladores/probadores trabajan con grupos de usuarios/clientes en las actividades relacionadas a la calidad y tareas orientadas a las necesidades del usuario. El enfoque que se pretende es solicitar al usuario/cliente su ayuda, consenso, y participación en actividades como análisis de requerimientos, pruebas de funcionalidad, y aprobación de los planes de pruebas.

8.5. Objetivos de Madurez en los Niveles de TMM

TMM provee una secuencia jerárquica de niveles que contienen los objetivos de madurez, sub-objetivos, y ATR que definen el estado de la madurez para pruebas de una organización en un nivel especial, estos niveles identifican áreas en las que una organización debe concentrarse para mejorar su proceso de pruebas. La jerarquía de los objetivos de madurez de pruebas es mostrada en la Figura Nº 11.

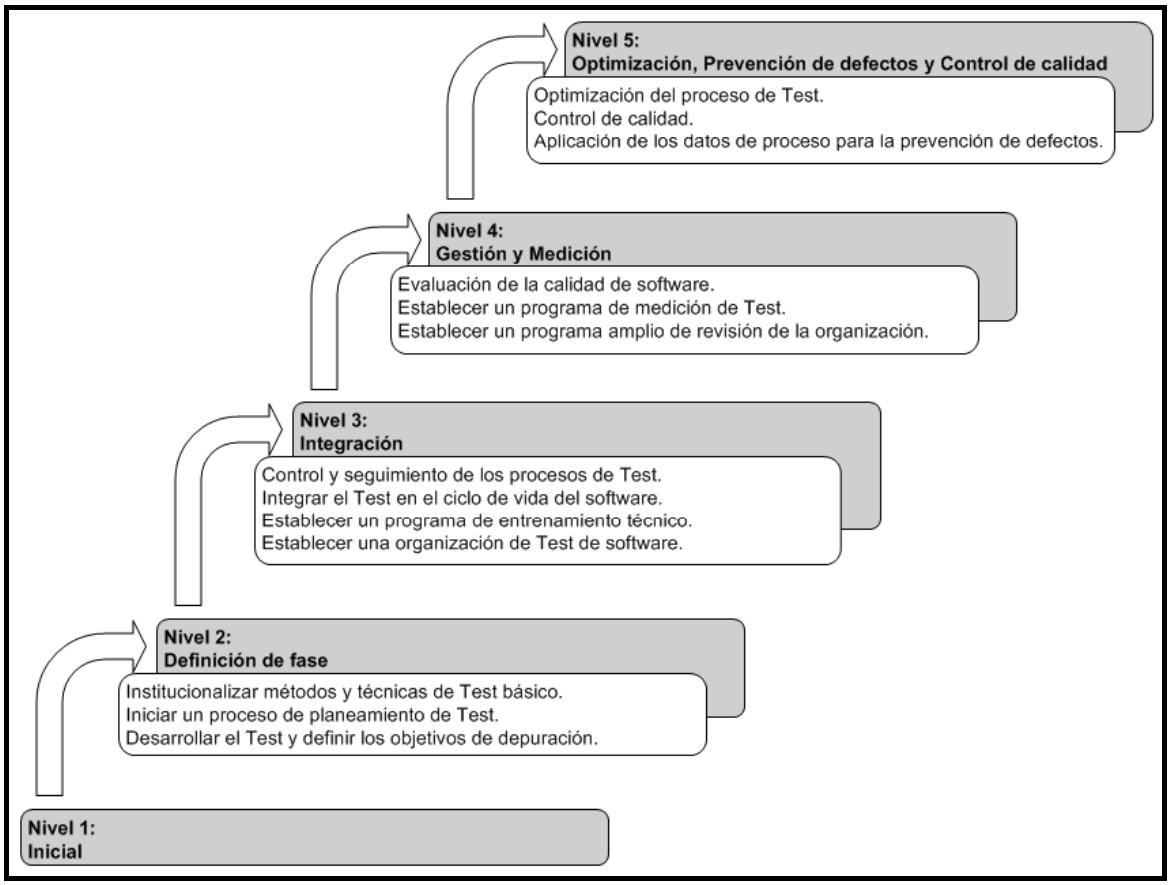


Figura Nº 11: Niveles del modelo TMM y sus objetivos de madurez

8.6. Descripción de los Niveles de TMM

8.6.1. Nivel 2: Definición de Fase

En el Nivel 2 de TMM una organización empieza a abordar los aspectos técnicos y directivos de madurez para realizar pruebas, una fase de prueba es definida en el ciclo de vida del software. Las pruebas son planeadas y apoyadas por técnicas básicas de pruebas y herramientas, y son repetibles en todos los proyectos de software. Aquí las pruebas son separadas de la depuración. Los siguientes son los objetivos de madurez de este nivel:

a. ***Desarrollar pruebas y definir objetivos de depuración;*** para hacer una clara diferencia entre la prueba y la depuración los objetivos, las tareas, las actividades, y las herramientas para cada uno deben ser identificados y las responsabilidades deben ser asignadas. La dirección de la organización debe crear e institucionalizar ambos procesos. Separar estos dos procesos es esencial para la evolución de madurez en las pruebas ya que son diferentes en sus objetivos, métodos, y forma. Realizar pruebas en el Nivel 2 de TMM es una actividad planeada y siempre puede ser dirigida, pero dirigir la depuración, es más complicado, porque es difícil pronosticar la naturaleza de los defectos que existirán y cuánto tiempo tomará el repararlos. Para reducir la imprevisibilidad de estos procesos a menudo causada por las actividades relacionadas con la depuración a gran escala el gerente de proyectos debe destinar el tiempo y los recursos para la localización de los defectos, las correcciones, y las nuevas pruebas, en niveles más altos del TMM esto será facilitado por la disponibilidad de información detallada acerca los defectos y de su corrección obtenida de proyectos anteriores.

b. ***Iniciar un proceso de planificación de pruebas;*** la planificación es esencial para un proceso que será repetido, definido, y administrado. La planificación de las pruebas requiere definir los objetivos, analizar los riesgos y las estrategias, y desarrollar las especificaciones del diseño de pruebas y los casos de pruebas. La planificación de las pruebas también supone documentar los criterios que determinan cuando estas ya se han completado. Además, el plan de pruebas debe abordar la asignación de recursos, la planificación de las actividades de pruebas, y las responsabilidades para hacer pruebas de unidad, de integración, del sistema, y los criterios de aceptación de conformidad con los resultados.

c. ***Institucionalizar métodos y técnicas básicos de pruebas;*** para mejorar el proceso de madurez los métodos y técnica básicos de pruebas deben ser aplicados en toda la organización, para ello deben ser especificados claramente de modo que se sepa en todo momento cuándo y cómo serán aplicados. Los métodos y técnicas básicos incluyen las estrategias de pruebas de caja blanca y de caja negra, el uso de matrices de validación de requerimientos y la división de las pruebas de ejecución basadas en sub-fases como pruebas de unidad, pruebas de integración, pruebas del sistema, y pruebas de aceptación.

8.6.2. Nivel 3: Integración

Las pruebas en el Nivel 3 de TMM se amplían a un conjunto de actividades bien definidas que están integradas en todas fases del ciclo de vida del software. En este nivel la dirección también aporta la formación y la capacitación de un equipo de pruebas de software, éstos son especialistas que son responsables en todos los niveles de las pruebas, y junto a profesionales de aseguramiento de calidad de software se desempeñan como

enlace con los usuarios/clientes para asegurar su participación en el proceso de pruebas. Los siguientes son objetivos de madurez del nivel 3:

a. ***Establecer una organización de pruebas de software;*** ya que las pruebas en si mismas tienen una gran influencia sobre la calidad del producto y dado que las actividades de desarrollo de un software tienen un cronograma ajustado y son realizadas por lo general bajo presión, es necesario tener un grupo entrenado y dedicado exclusivamente a este proceso. El equipo de pruebas constituido en el Nivel 3 de TMM supervisa la planificación de las pruebas, la ejecución y registro de los resultados de las pruebas, el seguimiento de los defectos encontrados, las métricas de las pruebas, la base de datos de pruebas, la reutilización de las pruebas, el afinamiento de las pruebas, y la evaluación.

b. ***Establecer un programa de entrenamiento técnico;*** un programa de capacitación técnica asegura personal experimentado en el equipo de pruebas. En el Nivel 3, el personal es entrenado en la planificación de pruebas, los métodos de pruebas, los patrones, las técnicas y las herramientas. El programa de entrenamiento también prepara al personal para el proceso de evaluación y para asegurar la participación del usuario en las pruebas y las actividades de evaluación.

c. ***Integrar las pruebas en el ciclo de vida del software;*** el equipo técnico debe realizar las actividades de pruebas en paralelo en todas las fases del ciclo de vida del software, este objetivo es crítico para lograr la madurez del proceso de pruebas y la calidad en el producto de software, esto puede lograrse con la aplicación de un modelo de desarrollo que soporte la integración de las actividades de pruebas en el ciclo de vida. Como consecuencia de los esfuerzos de integración, la planificación de las pruebas es iniciada con las primeras fases del ciclo de vida de desarrollo. El usuario es

considerado como un colaborador para el proceso de pruebas y se solicita su ayuda durante varias fases del ciclo de vida.

d. ***Controlar y monitorear los procesos de pruebas;*** monitorear y controlar el proceso de pruebas provee claridad a las actividades asociadas y asegura que el proceso de pruebas siga según lo planeado. El progreso de las pruebas se determina comparando las tareas de pruebas para los productos actuales, las pruebas de esfuerzo, los costos, y las tareas realizadas de acuerdo al plan de pruebas. El soporte para el control y monitoreo viene desde los estándares para las pruebas de los productos, log's de pruebas, pruebas relacionadas a los planes de contingencia, y métricas de pruebas que pueden ser usadas para evaluar el progreso y eficacia de las pruebas. La administración de la configuración de los ítems relacionados a las pruebas también otorgan el soporte adicional para este objetivo de madurez.

8.6.3. Nivel 4: Gestión y Medición

En el Nivel 4 de TMM el proceso de pruebas se hace completamente administrado; es decir es planeado, dirigido, proveído de personal, organizado y controlado. Las mediciones producto de las pruebas son definidas, registradas, analizadas, y usadas por los gerentes de proyecto, por el personal del equipo de aseguramiento de calidad de software, y por los probadores. La definición de una actividad de prueba es ampliada para incluir oficialmente las inspecciones en todas fases del ciclo de vida de software. Cada revisión e inspección sirve como un complemento a las pruebas basadas en la ejecución, todos estos pasos son vistos como procedimientos de control de calidad que pueden ser aplicados para eliminar los defectos del software. Los siguientes son los objetivos de madurez del Nivel 4:

a. ***Establecer un programa de revisión en la organización;*** en el Nivel 3 de TMM una organización integra actividades de pruebas en

el ciclo de vida del software, se hace énfasis es desarrollar planes de pruebas en los inicios del proceso de desarrollo. En el Nivel 4 esta integración se consolida por el establecimiento de un programa de evaluación formal. Todas las revisiones, en forma de inspecciones y validaciones, son consideradas actividades de pruebas y son dirigidas en todas las fases del ciclo de vida del software para identificar, catalogar, y eliminar los defectos del producto de software de manera fácil y efectiva. Pueden utilizarse cualquier medio para la integración de las actividades de evaluación y pruebas.

b. ***Establecer un programa de medición de pruebas;*** un programa de medición de pruebas es esencial para valorar la calidad y la eficacia del proceso de pruebas, asegurar la productividad del personal asignado a las pruebas, y monitorear la mejora del proceso de pruebas. Un programa de medición de pruebas debe ser planeado y dirigido cuidadosamente. Los datos de las pruebas deben ser identificados y registrados y se debe decidir cómo serán usados y por quién.

c. ***Establecer la calidad de evaluación del software;*** la evaluación de la calidad del software en este nivel de TMM relaciona los asuntos de calidad de software con la suficiencia del proceso de pruebas. La evaluación de calidad de software requiere que una organización defina los atributos medibles de calidad y los objetivos de calidad para evaluar cada tipo software. Los objetivos de calidad se relacionan a la suficiencia del proceso de pruebas ya que un proceso de pruebas maduro debe resultar en un software que es, mantenible, seguro, utilizable, portable, y seguro, entre otras características.

8.6.4. Nivel 5: Optimización, Prevención de defectos y control de calidad

En este nivel se realizan pruebas para asegurar que el software satisface su especificación, que es seguro, y que se puede tener contar con un alto nivel de confiabilidad, las pruebas también son realizadas para detectar y prevenir los defectos, esto último se consigue recolectando y analizando los datos de los defectos encontrados.

Debido a que el proceso de pruebas es repetible, definido, dirigido, y medido, puede ser puesto a punto y mejorado constantemente. La dirección de la organización lidera, motiva y respalda la infraestructura necesaria para mejorar el producto y el proceso de calidad constantemente. Los siguientes son los objetivos de madurez en este nivel:

a. ***Procesamiento de los datos históricos para la prevención de defectos;*** las organizaciones maduras son capaces de aprender de su pasado, las organizaciones en el nivel más alto de TMM registran los defectos, analizan los defectos e identifican las causas raíz de los errores, se desarrollan los planes de acción, y se toman las medidas necesarias para prevenir la repetición de defectos. Se debe contar con un equipo de prevención de defectos que es responsable de estas actividades, los miembros de este equipo interactúan con los desarrolladores para aplicar las actividades de prevención de defectos durante todo el ciclo de vida del software.

b. ***Control de calidad;*** en el Nivel 4 de TMM las organizaciones se concentran en evaluar basándose en un grupo de atributos relacionados con la calidad, como conformidad con los requerimientos funcionales, la portabilidad, la interoperabilidad, la usabilidad, y la mantenibilidad. En el nivel 5 las organizaciones usan muestreos estadísticos, las mediciones de niveles de confianza, de fiabilidad, y objetivos de confiabilidad para conducir los procesos de pruebas. El equipo de pruebas y el equipo de

aseguramiento de la calidad de software son los líderes de calidad, trabajan con diseñadores y desarrolladores de software para incorporar técnicas y herramientas para reducir defectos y mejorar la calidad del software. Las herramientas automatizadas soportan la ejecución repetida de los casos de pruebas y la recolección de los datos para el análisis de los defectos encontrados.

c. ***Optimización de los procesos pruebas;*** en el nivel más alto del TMM el proceso de pruebas está sujeto a la mejora ininterrumpida durante el desarrollo de los proyectos y en toda la organización. El proceso de pruebas es cuantificado y puede ser puesto a punto con el propósito de que la evolución de madurez sea un proceso continuo. Una infraestructura organizativa que consta de políticas, estándares, entrenamiento, instalaciones, herramientas, y estructuras organizativas provocan un indiscutible avance en la jerarquía del modelo madurez de TMM. Optimizar los procesos de prueba involucra:

- Identificar las prácticas de pruebas que necesitan ser mejoradas.
- Implementar las mejoras.
- Ajustar el progreso.
- Evaluar nuevas herramientas y tecnologías de pruebas.
- Respaldar la transferencia de tecnología.

9. PROPUESTA DE UN MODELO DE PRUEBAS PARA ALCANZAR EL NIVEL 2 DE TMM.

9.1. Requisitos del Modelo de Pruebas

En el Nivel 2 de TMM (Definición de Fase) uno de los objetivos de madurez más importantes es "Iniciar un proceso de planificación de pruebas", esto requiere que los siguientes sub-objetivos se cumplan:

- Establecer políticas de planificación de pruebas en la organización las cuales deben ser respaldadas por la alta dirección.
- Debe elaborarse una plantilla de planificación y especificación de pruebas la cual debe ser entregada y puesta a disposición de los gerentes de proyecto y desarrolladores.
- Se debe implementar un mecanismo para integrar los requisitos del usuario al plan de pruebas.

Además del cumplimiento de los sub-objetivos del nivel se deben establecer claramente las actividades, tareas y responsabilidades de cada uno de los participantes en las pruebas, para ello se deben utilizar las vistas definidas por TMM, para el Nivel 2 las vistas de cada uno de los participantes describen las siguientes actividades, tareas y responsabilidades:

9.1.1. La Vista de la Dirección

- La alta dirección debe suministrar el soporte, los recursos, y la capacitación para establecer las políticas adecuadas de planificación de pruebas y su puesta en práctica.
- El gerente de proyecto es responsable de asignar las responsabilidades a los desarrolladores para elaborar los planes de prueba y evaluar los componentes de software.
- El gerente de proyecto asegura que los usuarios aprueben el plan de pruebas, aquí se deben registrar los requisitos de los usuarios registrados durante el análisis de requerimientos y durante el análisis y diseño de la solución.

9.1.2. La vista del desarrollador/probador

- Los desarrolladores/probadores proveen lo necesario para el plan de pruebas, ayudando a determinar los objetivos de las pruebas, hacia donde estarán enfocadas, los métodos y procedimientos a usar, y las herramientas que ayudarán a realizar las pruebas.
- Los desarrolladores/probadores son responsables de elaborar las especificaciones de pruebas, los diseños de prueba, y los criterios de aprobación para cada nivel en el plan de pruebas.
- Los desarrolladores/probadores deben validar el ambiente de pruebas, verificando el estado de los equipos que se van a utilizar, y las herramientas de software requeridas para las mismas.

9.1.3. La vista del usuario/cliente

- Si es necesario los usuarios/clientes deben reunirse con los representantes del equipo de desarrollo para establecer los criterios de

aceptación para las pruebas, todas las contribuciones hechas por los usuarios/clientes deben ser registradas.

- Los usuarios describen requisitos funcionales, requisitos de rendimiento, y los otros atributos de calidad como la portabilidad, interoperabilidad, usabilidad, etc. por lo general estos criterios son detallados como parte del análisis de requerimientos y del análisis y diseño pero no esta demás que se validen.

9.2. Propuesta del Modelo de Pruebas

El Modelo propuesto para el Nivel 2 de TMM satisface todos los requisitos especificados, la Figura Nº 12 muestra en detalle el proceso de pruebas propuesto por el modelo.

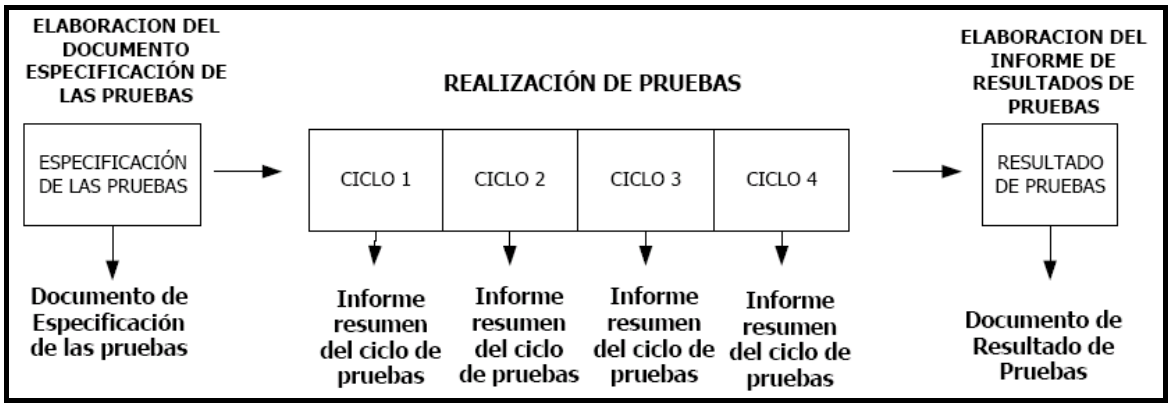
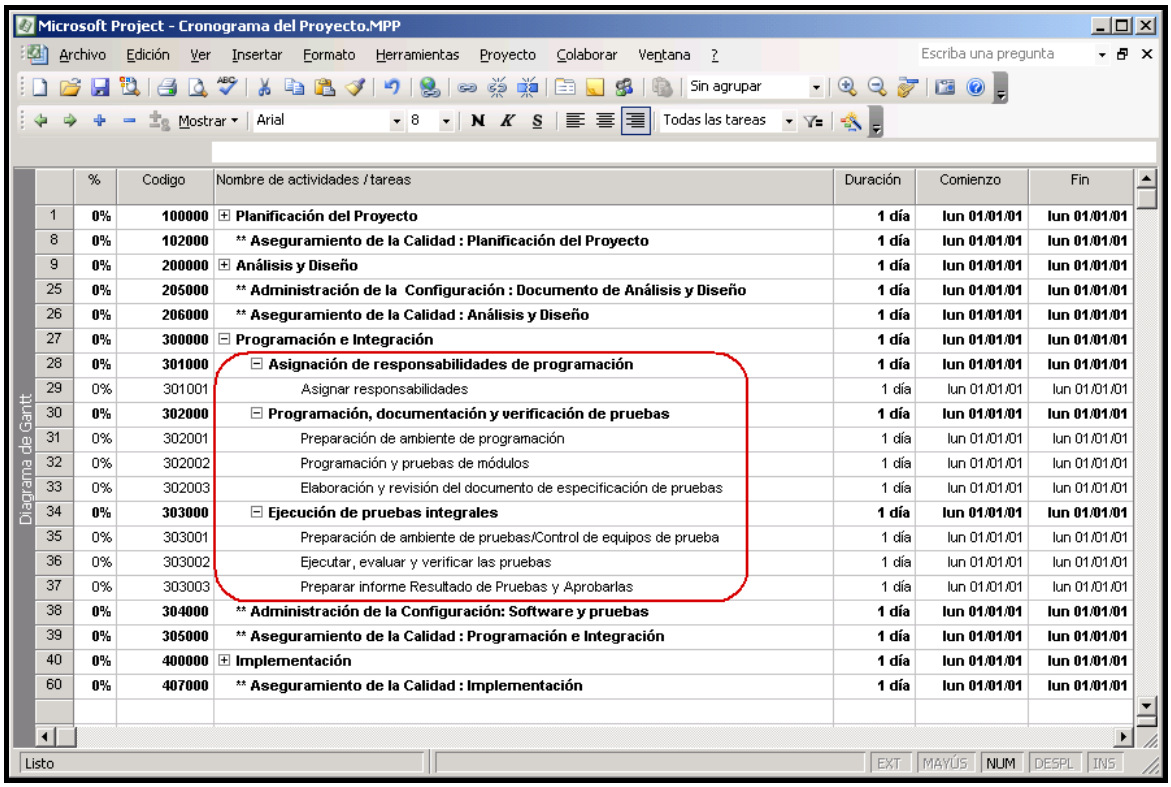


Figura Nº 12: Esquema del Proceso De Pruebas Propuesto

Los procedimientos del modelo de pruebas se cumplen en base a los sub-objetivos propuestos por el Nivel 2 de TMM.

9.2.1. Establecimiento de una Política de Planificación de Pruebas

Las pruebas deben ser planificadas desde el inicio del proyecto, las pruebas se conceptualizan en su fase inicial durante el análisis de requerimientos y posteriormente se afinaran durante el análisis y diseño. El plan del proyecto debe incluir una descripción de las pruebas que se deberán especificar, hay que recordar que dependerán mucho de las características del software, además como parte del cronograma del proyecto deben incluirse los tiempos que se utilizarán para todas las actividades del proceso de pruebas (Ver Figura Nº 13), y se establecerán los ciclos de pruebas a realizar.



	%	Codigo	Nombre de actividades / tareas	Duración	Comienzo	Fin
1	0%	100000	Planificación del Proyecto	1 día	lun 01/01/01	lun 01/01/01
8	0%	102000	** Aseguramiento de la Calidad : Planificación del Proyecto	1 día	lun 01/01/01	lun 01/01/01
9	0%	200000	Análisis y Diseño	1 día	lun 01/01/01	lun 01/01/01
25	0%	205000	** Administración de la Configuración : Documento de Análisis y Diseño	1 día	lun 01/01/01	lun 01/01/01
26	0%	206000	** Aseguramiento de la Calidad : Análisis y Diseño	1 día	lun 01/01/01	lun 01/01/01
27	0%	300000	Programación e Integración	1 día	lun 01/01/01	lun 01/01/01
28	0%	301000	Asignación de responsabilidades de programación	1 día	lun 01/01/01	lun 01/01/01
29	0%	301001	Asignar responsabilidades	1 día	lun 01/01/01	lun 01/01/01
30	0%	302000	Programación, documentación y verificación de pruebas	1 día	lun 01/01/01	lun 01/01/01
31	0%	302001	Preparación de ambiente de programación	1 día	lun 01/01/01	lun 01/01/01
32	0%	302002	Programación y pruebas de módulos	1 día	lun 01/01/01	lun 01/01/01
33	0%	302003	Elaboración y revisión del documento de especificación de pruebas	1 día	lun 01/01/01	lun 01/01/01
34	0%	303000	Ejecución de pruebas integrales	1 día	lun 01/01/01	lun 01/01/01
35	0%	303001	Preparación de ambiente de pruebas/Control de equipos de prueba	1 día	lun 01/01/01	lun 01/01/01
36	0%	303002	Ejecutar, evaluar y verificar las pruebas	1 día	lun 01/01/01	lun 01/01/01
37	0%	303003	Preparar informe Resultado de Pruebas y Aprobarlas	1 día	lun 01/01/01	lun 01/01/01
38	0%	304000	** Administración de la Configuración: Software y pruebas	1 día	lun 01/01/01	lun 01/01/01
39	0%	305000	** Aseguramiento de la Calidad : Programación e Integración	1 día	lun 01/01/01	lun 01/01/01
40	0%	400000	Implementación	1 día	lun 01/01/01	lun 01/01/01
60	0%	407000	** Aseguramiento de la Calidad : Implementación	1 día	lun 01/01/01	lun 01/01/01

Figura Nº 13 : Esquema de un cronograma de proyecto considerando un proceso de pruebas

Estas consideraciones se incluyen y se detallan en el Procedimiento de Planificación del Proyecto y es también una parte importante de la Metodología de Desarrollo de Proyectos, estos documentos deben estar al

alcance de todos los miembros del equipo desarrollador ya sea publicados en un servidor de directorios o en una intranet. Todos los miembros participantes directa o indirectamente en el ciclo de desarrollo de software deben estar enterados y concientes de la importancia de su rol en el cumplimiento de estos procedimientos. De esta manera las funciones principales de los actores participantes en el desarrollo del proyecto con respecto al proceso de pruebas son las siguientes:

- **Gerente del Proyecto**

    - Gestionar y coordinar los recursos adecuados para llevar a cabo las actividades del proyecto de configuración y de evaluación y pruebas.
    - Verificar que la asignación de recursos se realice con oportunidad y calidad requeridas para el proyecto.
    - Elaborar y actualizar el plan de actividades del proyecto, delegar a los responsables las tareas de pruebas que les corresponden y controlar su cumplimiento.
    - Revisar los documentos que se generen los procesos de pruebas.
    - Aprobar las pruebas ejecutadas por el equipo de trabajo designado.

- **Equipo de Trabajo**

    - Elaborar los documentos que se requieren en las diferentes actividades de pruebas del proyecto.
    - Reportar al Gerente de Proyecto los resultados obtenidos y solicitar su aprobación respecto a productos entregables.
    - Documentar continuamente y según sea requerida la ejecución y los resultados a fin de facilitar el control y seguimiento.

En la Tabla Nº 7 se muestra el cuadro de responsabilidades de elaboración y revisión de documentos para el proceso de pruebas y en la

Tabla Nº 8 el cuadro de actividades vs. entregables para el proceso de pruebas.

Proceso	Documento	Responsables	
		Elaborar Documento	Revisar y Aprobar Documento
Proceso de Pruebas	• Documento de Especificación de Prueba	• Equipo de Trabajo	• Gerente de Proyecto
	• Documento de Resultado de Pruebas	• Equipo de Trabajo	• Gerente de Proyecto
	• Lista de verificación de dispositivos	• Equipo de Trabajo	• Gerente de Proyecto
	• Documento resumen del ciclo de pruebas	• Equipo de Trabajo	• Gerente de Proyecto

Tabla Nº 7: Cuadro de Responsabilidades en la elaboración de los documentos del Procesos de Pruebas

Proceso	Actividades	Puntos de Control	Entregables
Proceso de Pruebas	• Elaboración de documentos de prueba • Verificación de dispositivos • Ejecución de pruebas • Resumen del ciclo de pruebas	**Revisión:** • Documento de Especificación de Pruebas • Lista de verificación de dispositivos **Validación:** • Resultado de las pruebas	• Documento de Especificación de Pruebas • Documento de Resultado de Pruebas • Lista de verificación de dispositivos • Informe de resumen del ciclo de pruebas

Tabla Nº 8: Cuadro de Actividades vs. Entregables del Procesos de Pruebas

9.2.2. Elaboración de las Plantillas de Especificación y Resultado de Pruebas

Como parte del modelo se presentan las plantillas que ayudan al equipo de trabajo encargado de las actividades del proceso de pruebas a realizar las tareas encomendadas, es necesario definir previamente un documento que detalle las características y objetivos que se pretenden con la elaboración de las especificaciones de las pruebas, en los anexos del presente documento se detallan las siguientes plantillas:

- **Anexo A:** ***Plantilla de Especificación de Pruebas***
- **Anexo B:** ***Plantilla de Resultados de Pruebas***
- **Anexo C:** ***Lista de Verificación de Dispositivos***
- **Anexo D:** ***Informe Resumen de Ciclo de Pruebas***

CONCLUSIONES

1. A lo largo del desarrollo de esta tesis se demostró la utilidad de los estándares y modelos de madurez para la evaluación de software a través de los procesos de pruebas, particularmente TMM. Para ello fue necesario hacer previamente una introducción detallada de los temas relacionados de manera tal que sean incorporados los conocimientos necesarios, alcance y limitaciones.

2. En particular sobre todo lo estudiado y presentado en el presente trabajo, se puede considerar que TMM es uno de los modelos de madurez más completo y objetivo, el cual brinda no solo las características de un entorno de pruebas, sino que gracias a su diseño basado en las mejores prácticas de CMM y CMM-SW, se pueden obtener mejoras importantes que ayudan a incrementar el nivel de la organización y hacer del proceso de pruebas un proceso estandarizado de mejora continua que ayude en la determinación de los niveles de calidad del software, por lo que se hace más que importante la aplicación de este modelo en la industria del software como un soporte adicional a la aplicación de ISO, CMM o CMM-SW.

3. La decisión de la aplicación de un estándar para el proceso de pruebas en el desarrollo de software, no asegura que el mismo brinde una solución infalible ya que como todo proceso requiere no solo de aprenderlo, y conocerlo sino que se hace necesario que el mismo sea adoptado como un estilo de trabajo en toda la organización o área involucrada en el desarrollo de software.

4. En cuanto al modelo de pruebas propuesto, constituye efectivamente una herramienta de evaluación y propuesta de mejoras de acuerdo a los objetivos del Nivel 2 de TMM, esta afirmación se puede justificar con las siguientes razones:

   - El modelo esta diseñado de manera que cumple los objetivos y sub-objetivos propuestos por el Nivel 2 de TMM que se pretende alcanzar.
   - Incorpora los tipos de pruebas que se estudiaron durante el desarrollo de la tesina.
   - Contiene los procedimientos de Especificación de Pruebas y las plantillas para la Verificación de Dispositivos, Especificaciones y Resultados de Pruebas.

5. La aplicabilidad del modelo de pruebas presentado esta restringida por el uso de una metodología de calidad en la organización, el modelo de pruebas y en general el proceso de madurez de TMM esta condicionado por la evolución de la organización en un modelo superior que puede ser incluso el ISO para sus primeras etapas pero que de preferencia debería ser CMM o CMM-SW.

REFERENCIAS BIBLIOGRÁFICAS

Berlinches Cerezo, Andrés
2002 Calidad. España: Thomson Paraninfo, 6ta Edición, 2002.

Capability Maturity Model
2005 Capability Maturity Model for Software (SW-CMM) http://www.sei.cmu.edu/cmm/cmms/cmms.html: 2005-04-15.

Capability Maturity Model
2005 Introduction to CMMI (Staged and Continuous) http://www.sei.cmu.edu/cmmi/cmmi.html: 2005-02-10.

Glenford J. Myers.
1999 El Arte de Probar el Software. Principios de las Pruebas de Software. Buenos Aires: Ateneo,1999.

Hutcheson, Marnie.
2003 Software Testing Fundamentals, Methods and Metrics. EE.UU: Wiley Publishing, 1ra Edición, 2003.

IEEE
1990 Standard 610, Computer Dictionary. Nueva York.

Instituto Nacional de Estadística e Informática (INEI)
1999 ¿Qué es Calidad Total?. Calidad Total en los Sistemas de Información. Editorial INEI: Lima-Perú, 1999.

ISO8402
1994 Gestión de la Calidad y Aseguramiento de la Calidad.

ISO9000
2000 Estándar de Gestión de Calidad.

ISO/IEC-9126
1991 International Standard, “Information technology – Software product evaluation – Quality characteristics and guidelines for their use”

Jara, Eduardo G.
2004 Sistema de Aseguramiento de Calidad. Bases del Sistema de Aseguramiento de Calidad. Tesis de Bachiller. Facultad de Sistemas de Información, Universidad del Bío-Bío, Concepción, Chile.

JIS
1981 Japanese Industrial Standard JIS z8110-1981

Minguet Melián, Jesús María
2005 Calidad de Software. http://www.issi.uned.es/CalidadSoftware/contenidos.html: 2005-01-24.

Piattini, Mario
2003 Calidad en el Desarrollo y Mantenimiento de Software. Gestión de la Calidad de Software. España: Ra-Ma, 2003.

Poma Dioses, James
2003 Gestión de la Calidad Total. Conceptos de la Calidad Total. España: Prentice Hall, 1997.

Pressman, Roger S.
2002 Ingeniería de Software. Introducción a la Ingeniería de Software. España: Mc. Graw-Hill, 2002.

Software Quality Professional
1999 A Testing Maturity Model for Software Test Process Assessment and Improvement http://www.asq.org/pub/sqp/past/vol1_issue4/burnstein.html: 1999-09-01.

Tim Koomen, Martin Pool
1999 Test Process Improvement – A practical step-by-step guide to structured testing. New York: ACM Press Books, 1999

ANEXO A: PLANTILLA DE ESPECIFICACIÓN DE PRUEBAS

<NOMBRE DEL PROYECTO>
<NOMBRE DE LA EMPRESA>
DOCUMENTO DE ESPECIFICACION DE PRUEBAS
VERSIÓN <Nº de Versión>
FECHA <dd/mmm/aaaa>

1. OBJETIVO

Establecer los lineamientos y procedimientos a seguir para realizar la actividad propuesta.

2. ALCANCE

De acuerdo al documento de análisis de requerimiento se puede plantear el alcance de la actividad.

3. ESQUEMA DE PRUEBAS

Establecer en que consiste las pruebas, cómo se van a llevar a cabo, etc. Las pruebas que se realizaran consisten en los siguientes frentes:

- Pruebas de Interfaz
- Pruebas de Funcionalidad
- Pruebas de Revisión de Código
- Pruebas de Integración
- Condiciones de Excepción
- Pruebas de Mensajes de Error
- Pruebas de Integridad y Seguridad
- Pruebas de Perfomance

4. REQUERIMIENTO PARA LAS PRUEBAS

Para el desarrollo de las pruebas se requiere contar con los siguientes requerimientos:

4.1. Requerimientos de Datos

4.2. Requerimientos de Simuladores y Generadores de Datos

4.3. Requerimientos de Hardware y Software

4.4. Requerimiento de Recursos Humanos

4.5. Otros Requerimientos

5. CRONOGRAMA DE PRUEBAS

Establecer el cronograma de las pruebas indicando los recursos que se encargaran de cada una de las actividades.

6. ESCENARIOS Y CASOS DE PRUEBAS

6.1. Pruebas de Interfase

Estas pruebas deben realizarse teniendo en cuenta los siguientes criterios:

- Tamaño de tramas
- Tipos de datos soportados
- Mínima y Máxima Capacidad de Campos
- Valores Mínimos y Máximos
- Caracteres Especiales

- Usabilidad

Nro	Descripción de la prueba	Respuesta esperada	Comentarios

Revisión de estándares de presentación				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿Están claramente definidos los bloques de información (Frames)?				
¿Tiene los encabezados de título y nombre de aplicación correctos?				
¿Las etiquetas de los campos son claras y representativas?				
¿Los campos de despliegue están completamente inhabilitados y del color respectivo?				
¿Los campos de solamente despliegue están claramente identificados?				
¿Tiene los colores estándar?				
¿Los campos fecha tienen el formato DD-MMM-AAAA y se puede ingresar los datos como Ej: 12-AGO-2004?				
Cuando se tiene un formulario con múltiples tabs, ¿se conoce cuál es el registro padre de los tabs?				
¿La forma tiene la dimensión correcta?				
¿Los Radio Groups tienen un frame que los abarca?				
¿Los campos están alineados en forma correcta?				
¿Los campos requieren y tienen Tooltip?				
…				

6.2. Pruebas de Funcionalidad

Estas pruebas deben realizarse para la nueva funcionalidad desarrollada así como para cualquier otra funcionalidad que pueda verse afectada por los módulos nuevos y/o ajustados.

Nro	Descripción de la prueba	Respuesta esperada	Comentarios

Revisión de funcionalidad				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿El formulario realiza la función que se necesita?				

¿Los datos del formulario cambian en forma sincronizada?				
¿Es rápido y fácil el manejo de la forma?				
Cuando se cambia el valor de un campo de entrada, ¿se modifica también el campo de despliegue?				
¿Los bloques hijos están coordinados con el bloque padre en consulta, borrado y cuando se limpia el formulario?				
Los campos que hacen referencia a datos de otras tablas ¿tienen cada uno su lista de valores?				
¿Las listas de valores son lentas para recuperar la información?				
¿El tiempo de respuesta es adecuado?				
¿El orden de navegación de los campos es el correcto?				
¿Los mensajes graves son manejados adecuadamente?				
¿Si el reporte requiere mucho tiempo, esto le es notificado al usuario?				
¿Está el formulario documentada?				
…				

6.3. Pruebas de Revisión de Código

Estas pruebas deben realizarse para asegurar que el código de programación del aplicativo cumple con los estándares de la organización o del cliente.

Revisión del Código				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿Se ha hecho revisión por pares?				
¿Se ha realizado el proceso de afinamiento sql?				
¿El código cumple con los estándares?				
...				

Estándares de las tablas				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿El nombre de la tabla es correcto según los estándares?				
¿Tiene las descripciones de la columna en la base de datos?				
¿Tiene las llaves e índices adecuados?				
…				

6.4. Pruebas de Integración

Estas pruebas se realizan para asegurar la integración de los nuevos módulos a todo el sistema o con otros sistemas existentes.

Nro	Descripción de la prueba	Respuesta esperada	Comentarios

6.5. Pruebas de Condición de Excepción

Para la construcción de estas pruebas se considerarán situaciones de errores o acciones que interrumpan el funcionamiento del software, como podrían ser:

- Archivo no existente, sin información, nombre errado, tamaño de registro errado, data inválida, etc.
- Desconexiones de base de Datos, bloqueo de tablas
- Desconexión de la red, etc.

Nro	Descripción de la prueba	Respuesta esperada	Comentarios

6.6. Pruebas de Mensajes de Error

Se define cada uno de los casos de error y se verifica que los mensajes y/o códigos de error sean consistentes con el error presentado.

Nro	Descripción de la prueba	Respuesta esperada	Comentarios

6.7. Pruebas de Integridad y Seguridad

Se define cada uno de los casos de prueba de integridad y seguridad

Nro	Descripción de la prueba	Respuesta esperada	Comentarios

6.8. Pruebas de Performance

Se definen las pruebas de performance las cuales deben incluir:

6.8.1. Consumo de Recursos

Nº Procesos Concurrentes	Cantidad de TXN	Memoria (Kb)	CPU (%)	Espacio en Disco (Mb)	Duración (min)
1	**10**				
1	100				
1	1000				
1	::::				
5	**10**				
5	100				
5	1000				
5	::::				
10	**10**				
10	100				
10	1000				
10	::::				
::::	**::::**				

6.8.1.1. Tiempo de Respuesta

Pruebas individuales :

Cantidad de TXN	Hora de Inicio	Hora de Fin	Tiempo Total	TXN por Segundo
1				
10				
1000				
10000				
::::				

Pruebas de Concurrencia: Diez Instancias

Cantidad de TXN	Hora de Inicio	Hora de Fin	Tiempo Total	TXN por Segundo
10				
100				
1000				
::::				

Pruebas de Concurrencia: "n" Instancias

Cantidad de TXN	Hora de Inicio	Hora de Fin	Tiempo Total	TXN por Segundo
10				
100				
1000				
::::				

ANEXO B: PLANTILLA DE RESULTADOS DE PRUEBAS

<NOMBRE DEL PROYECTO>
<NOMBRE DE LA EMPRESA>
DOCUMENTO DE RESULTADOS DE PRUEBAS
VERSIÓN <Nº de Versión>
FECHA <dd/mmm/aaaa>

1. OBJETIVO

Establecer los lineamientos y procedimientos a seguir para realizar la actividad propuesta.

2. ESCENARIOS Y CASOS DE PRUEBAS

En los comentarios de ser posible debe colocarse la fecha en que se realizaron las pruebas y/o ajustes.

2.1. Pruebas de Interfase

Nro	Descripción de la prueba	Respuesta	Comentarios

Revisión de estándares de presentación				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿Están claramente definidos los bloques de información (Frames)?				
¿Tiene los encabezados de título y nombre de aplicación correctos?				
¿Las etiquetas de los campos son claras y representativas?				
¿Los campos de despliegue están completamente inhabilitados y del color respectivo?				
¿Los campos de solamente despliegue están claramente identificados?				
¿Tiene los colores estándar?				
¿Los campos fecha tienen el formato DD-MMM-AAAA y se puede ingresar los datos como Ej: 12-AGO-2004?				
Cuando se tiene un formulario con múltiples tabs, ¿se conoce cuál es el registro padre de los tabs?				
¿La forma tiene la dimensión correcta?				
¿Los Radio Groups tienen un frame que los abarca?				
¿Los campos están alineados en forma correcta?				
¿Los campos requieren y tienen Tooltip?				
…				

2.2. Pruebas de Funcionalidad

Nro	Descripción de la prueba	Respuesta	Comentarios

Revisión de funcionalidad				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿El formulario realiza la función que se necesita?				
¿Los datos del formulario cambian en forma sincronizada?				
¿Es rápido y fácil el manejo de la forma?				
Cuando se cambia el valor de un campo de entrada, ¿se modifica también el campo de despliegue?				
¿Los bloques hijos están coordinados con el bloque padre en consulta, borrado y cuando se limpia el formulario?				
Los campos que hacen referencia a datos de otras tablas ¿tienen cada uno su lista de valores?				
¿Las listas de valores son lentas para recuperar la información?				
¿El tiempo de respuesta es adecuado?				
¿El orden de navegación de los campos es el correcto?				
¿Los mensajes graves son manejados adecuadamente?				
¿Si el reporte requiere mucho tiempo, esto le es notificado al usuario?				
¿Está el formulario documentada?				
…				

2.3. Pruebas de Revisión de Código

Estas pruebas deben realizarse para asegurar que el código de programación del aplicativo cumple con los estándares de la organización o del cliente.

Revisión del Código				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿Se ha hecho revisión por pares?				
¿Se ha realizado el proceso de afinamiento sql?				
¿El código cumple con los estándares?				
...				

Estándares de las tablas				
Actividad	**Si**	**No**	**No aplica**	**Información Adicional**
¿El nombre de la tabla es correcto según los estándares?				

¿Tiene las descripciones de la columna en la base de datos?				
¿Tiene las llaves e índices adecuados?				
…				

2.4. Pruebas de Integración

Nro	Descripción de la prueba	Respuesta	Comentarios

2.5. Pruebas de Condición de Excepción

Nro	Descripción de la prueba	Respuesta	Comentarios

2.6. Pruebas de Mensajes de Error

Nro	Descripción de la prueba	Respuesta	Comentarios

2.7. Pruebas de Integridad y Seguridad

Nro	Descripción de la prueba	Respuesta	Comentarios

2.8. Pruebas de Performance

2.8.1. Consumo de Recursos

Nº Procesos Concurrentes	Cantidad de TXN	Memoria (Kb)	CPU (%)	Espacio en Disco (Mb)	Duración (min)
1	**10**				
1	100				
1	1000				
1	::::				
5	**10**				
5	100				
5	1000				
5	::::				
10	**10**				
10	100				
10	1000				
10	::::				
::::	**::::**				

2.8.1.1. Tiempo de Respuesta

Pruebas individuales :

Cantidad de TXN	Hora de Inicio	Hora de Fin	Tiempo Total	TXN por Segundo
1				
10				
1000				
10000				
::::				

Pruebas de Concurrencia: Diez Instancias

Cantidad de TXN	Hora de Inicio	Hora de Fin	Tiempo Total	TXN por Segundo
10				
100				
1000				
::::				

Pruebas de Concurrencia: "n" Instancias

Cantidad de TXN	Hora de Inicio	Hora de Fin	Tiempo Total	TXN por Segundo
10				
100				
1000				
::::				

ANEXO C: FORMATO DE LISTA DE VERIFICACIÓN DE DISPOSITIVOS

LISTA DE VERIFICACIÓN DE DISPOSITIVOS

I. INFORMACIÓN DEL DISPOSITIVO:

Dispositivo: _____

Descripción: _____

Procesador: _____ **Memoria:** _____

Espacio en Disco: _____ **S.O:** _____

Software Instalado: _____

II. RESULTADOS DE LA VERIFICACIÓN:

	SI	NO
El dispositivo cumple con los requisitos:.....	☐	☐
El Hardware esta conforme:.....	☐	☐
El Sistema Operativo esta conforme:.....	☐	☐
El espacio en disco disponible es suficiente:.....	☐	☐
Tarjeta de red conforme:.....	☐	☐
Instalación de software base conforme:.....	☐	☐

III. OBSERVACIONES:

_____

_____

_____

_____

_____

Verificador:_____ **Firma:**_____

Fecha:_____

ANEXO D: FORMATO INFORME DEL CICLO DE PRUEBAS

INFORME RESUMEN DE CICLO DE PRUEBAS

Proyecto/Producto : _____

Módulo : _____

Nº Ciclo : _____

Fecha : _____

TIPOS DE PRUEBA	N° ERRORES	N° SUGERENCIA DE MEJORA	N° CONSULTA AL DISEÑO	TOTALES
Pruebas de Interfase				
Pruebas de Funcionalidad				
Pruebas de Integración				
Pruebas de Condición de Excepción				
Pruebas de Mensajes de Error				
Pruebas de Seguridad				
Pruebas de Performance				
...				
....				
TOTALES:				

Observaciones:

_____

_____

_____

_____

Elaborado por:_____ **Firma:**_____

Revisado por:_____ **Firma:**_____

Fecha:_____

GLOSARIO

Aseguramiento de Calidad: todas las actividades planificadas y sistemáticas necesarias para aportar la confianza suficiente en que un producto o servicio cumplirá con unos requisitos dados de calidad.

Comprensión: subcaracterística de facilidad de uso, que indica las características del software que influyen en el esfuerzo del usuario para reconocer el concepto lógico y su aplicación

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

Estándar: nivel de desempeño esperado y alcanzable, comparable con el nivel de desempeño actual.

Fiabilidad: grado en que el sistema responde bajo las condiciones definidas durante un intervalo de tiempo dado. Se divide en las subcaracteríticas madurez, tolerancia a fallos, capacidad de recuperación

Funcionalidad: grado en que las necesidades asumidas o descritas se satisfacen. Se divide en las subcaracteríticas idoneidad, precisión, interoperabilidad, seguridad

Interoperabilidad: subcaracterística de funcionalidad, que indica el grado en que el sistema puede interactuar con otros sistemas

Madurez: subcaracterística de fiabilidad, que indica la frecuencia con que ocurren los fallos

Mantenimiento: esfuerzo requerido para implementar cambios. Se divide en las subcaracteríticas capacidad de ser analizado, cambiabilidad, estabilidad, facilidad de prueba

Operabilidad: subcaracterística de facilidad de uso, que indica las características del software que influyen en el esfuerzo del usuario para operar y control operacional

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

Procedimiento: método o sistema estructurado para ejecutar algunas cosas. Acto o serie de actos u operaciones con que se hace una cosa.

Revisión: reuniones de un grupo definido de personas cuyo objetivo es encontrar errores en un artefacto de software. Con revisiones para testear requisitos, diseño, planes, manuales y software Participantes de los revisiones son: los autores que han escrito el artefacto; los revisores que tienen que detectar errores; el secretario que documenta los errores encontrados; el presentador que expone/explica el artefacto bajo testeo; el líder que dirige la reunión, elige la fecha para la reunión y invita a los participantes. Generalmente se distingue 2 tipos de revisiones: inspecciones (formal) walkthroughs (más informal)

Seguridad: subcaracterística de funcionalidad, que indica el grado en que un acceso no autorizado (accidental o deliberado) se prevenga y se permita un acceso autorizado

Testware: cualquier resultado de los procesos de testeo (casos de prueba, planes, etc.)

Usabilidad: 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.