UNA ESTRATEGIA PARA EL MEJORAMIENTO DEL PROCESO DE VERIFICACIÓN Y VALIDACIÓN

CARLOS MÉNDEZ CÁLIZ

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
BOGOTA, D.C.
2004

UNA ESTRATEGIA PARA EL MEJORAMIENTO DEL PROCESO DE VERIFICACIÓN Y VALIDACIÓN

CARLOS MÉNDEZ CÁLIZ

Proyecto de grado para obtener el título
de Magíster en Ingeniería de Sistemas

Director de Tesis: Rubby Casallas

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
BOGOTA, D.C.
2004

TABLA DE CONTENIDO

FIGURAS

1.CONTEXTO

La mala calidad del software es un problema a nivel mundial. A medida que el software se convierte en parte integral de nuestras vidas, las consecuencias de la mala calidad pueden ser incluso catastróficas. En junio de 2001, un defecto en el software de servidor web Microsoft IIS[1] permitió que miles de servidores de Internet fueran atacados por el gusano[2] "Código Rojo"(En ingles, "Code Red"), lo que causó que muchos sitios web e Intranet corporativas suspendieran el servicio por varias horas y en algunos casos por varios días [31]. Se estima que las suspensiones del servicio causaron miles de millones de dólares en perdidas a la economía mundial. Algunos años antes, en 1996, el cohete francés Arianne 5 explotó en el aire 40 segundos después de haber despegado. El comité creado para determinar las causas del percance, sostuvo que el accidente se debió a un "error sistemático en el diseño del software". El cohete Arianne llevaba un satélite de 500 Millones de dólares del cual quedó nada [31].

Algunos problemas menos críticos relacionados con la calidad del software son vividos día a día por miles de usuarios empresariales. En una encuesta realizada en 1999 a usuarios empresariales de Gran Bretaña, el 23 % reportó que su trabajo era interrumpido al menos una vez al día y un porcentaje similar(el 20%) afirmó que gastaban al menos 3 horas diarias solucionando problemas de informática [23].

Un estudio realizado por Standish Group encontró que los costos por perdida de productividad y reparaciones de fallos en el software alcanzaban la cifra de 100 mil millones de dólares solamente en el año 2000, y que el 45% del tiempo en que los sistemas estaban fuera de servicio se debía a defectos en el código fuente[49].

La mayoría de estudios estiman los costos asociados a fallas causadas por la baja o nula confiabilidad del software. Sin embargo, también existen costos debido a sistemas difíciles de usar[17] . Un sistema que los usuarios encuentran difícil de usar incrementa los costos de entrenamiento y soporte, además que disminuye (en vez de aumentar) la productividad de los procesos y los usuarios.

La mala calidad del software es sin duda un problema con grandes consecuencias económicas para la sociedad.

Para mejorar la calidad del software, autores de la academia, institutos de investigación y de la industria plantean diversas propuestas. Estas propuestas se pueden categorizar en tres áreas[3]: Procesos, Tecnologías y Personal. En el área de Procesos se incluyen nuevas prácticas y métodos, modelos de valoración y mejoramiento del proceso, estándares de ciclo de vida, entre otros. En el área de Tecnologías podemos mencionar nuevos lenguajes de programación, herramientas CASE, ambientes integrados de desarrollo (en ingles IDE's), tecnologías de reuso (Ej. librerías, componentes, arquitecturas para reuso o "frameworks"), herramientas para automatizar o apoyar actividades del proceso (Ej:gestión de configuración, pruebas,

[1] El defecto era el conocido "buffer overflow"
[2] Un gusano es un virus de computador especial
[3] Estas áreas corresponden a los "puntos de apalancamiento de calidad". Ver Tutorial de CMMI en: http://www.sei.cmu.edu/cmmi/presentations/euro-sepg-tutorial/

análisis de código, gestión de proyectos, gestión de requerimientos, etc). En el área de Personal se pueden mencionar nuevos modelos[4], principios y técnicas en temas como conformación de equipos de desarrollo, selección y asignación de personal, desarrollo de carrera de ingenieros, incentivos y motivación, entre otros.

Siguiendo las ideas de gestión de calidad aplicadas con éxito por los japoneses después de la segunda guerra mundial y luego por Occidente en la década de 1980, la industria de software y los académicos han orientado un gran número de propuestas e ideas a mejorar el proceso antes que las tecnologías de desarrollo de software. En esta área, las propuestas y nuevas ideas generalmente pertenecen a dos escuelas de pensamiento diferentes[13][47]. Una primera escuela de pensamiento, que podríamos llamar "formal", se caracteriza por promover la aplicación de los principios de gestión total de la calidad y gestión de procesos. En esta escuela de pensamiento, las propuestas representativas son el modelo CMMI y el modelo SPICE. Una segunda escuela de pensamiento, autodenominada "procesos ágiles", se caracteriza por promover la adaptación al cambio y un balance entre la orientación al proceso y a las personas. En esta escuela de pensamiento, la propuesta representativa es la metodología XP[8]. Los principios que promueve la escuela de "procesos ágiles" se encuentran resumidos en el sitio web del "manifiesto ágil"(http://www.agilemanifiesto.org) .

Las principales diferencias entre las dos escuelas de pensamiento se encuentran resumidas en la siguiente tabla:

Escuela de procesos "Formal"	**Escuela de procesos "Ágiles"**
-Capacidad de predicción -Estabilidad de Especificaciones y Arquitectura -Asume que el costo del cambio aumenta exponencialmente al transcurrir el ciclo de vida -Gran número de productos de trabajo y documentación -Gestión de procesos: definir procesos, controlar y medir su desempeño y resultados -Aplicación principios de Calidad Total -Aseguramiento y control de calidad independiente -Paradigma mecanicista -Se origina en la década de 1980	-Capacidad de adaptación -Cambio -Requerimientos cambiantes -Asume que el costo del cambio aumenta de forma lineal o se mantiene constante al transcurrir el ciclo de vida -Mínimo número de productos de trabajo y documentación -Orientación al personal -Aseguramiento y control de calidad realizado por el mismo equipo -Paradigma biológico -Se origina en la década de 1990

Ambas escuelas de pensamiento tienen historias de éxitos y fracasos que mostrar, sin embargo, la escuela de procesos "ágiles" por ser relativamente nueva , le corresponde

[4] Un modelo para desarrollo de personal muy conocido es People-CMM. Puede descargar el modelo en el sitio web del SEI: http://sei.cmu.edu.co

probar que sus propuestas pueden ser una alternativa viable a las de la escuela de procesos “formal”.

Las organizaciones de desarrollo de software a nivel mundial, pueden adoptar las propuestas de cualquiera de estas escuelas bien sea de manera exclusiva o intentando tomar lo mejor que ofrece cada una según sirva a sus necesidades particulares.

En este punto vale la pena preguntarse que están haciendo las empresas de desarrollo de software de nuestro país frente al tema del mejoramiento del proceso. Siguen de forma exclusiva alguna de las escuelas de pensamiento mencionadas(formal o ágil)? Si es así, cuál de ellas? O intentan combinar lo mejor que ofrece cada una?

De acuerdo a un estudio realizado por Consuelo Ardila del sector de software en Colombia[2], la tendencia, al menos al revisar las iniciativas del estado, de los gremios y de las empresas líderes, es hacia la escuela de pensamiento de procesos “formal”. Esta tendencia se evidencia por el interés existente en las certificaciones ISO 9001:2000 y CMMI[5] . Ardila menciona: “El tema de calidad es primordial para el Ministerio de Comercio Exterior y está incluido dentro del convenio de competitividad exportadora para la cadena productiva de software y asociados. Al programa de Aseguramiento y Certificación de la calidad se encuentran vinculadas entidades como el SENA y Proexport quienes financian parcialmente proyectos para la certificación de la calidad y Fedesoft y CATI trabajan en cuatro programas: ISO 9001:2000, PSP/TSP para empresas pequeñas, mejores practicas de desarrollo, CMM”.

Al momento de la elaboración de este documento, varias empresas de software colombianas han obtenido certificaciones ISO 9001 y una empresa, ha sido valorada en CMMI nivel 5[6].

Bajo este contexto, las propuestas sobre mejoramiento de procesos deberían principalmente apoyarse en las teorías de gestión total de la calidad y gestión de procesos, que son la base de la escuela formal de procesos de software. Sin embargo, no deberían descartarse de plano las propuestas de procesos ágiles. Estas pueden ser una alternativa en proyectos y/o organizaciones pequeñas y cuando se tengan requerimientos poco estables o desconocidos al inicio del desarrollo. También pueden ser un paso inicial antes de intentar adoptar propuestas de la escuela “formal”.

---

[5] En realidad, no existe un proceso de certificación en CMMI, como lo existe para ISO 9001. Lo que se realiza son valoraciones (el termino en ingles es “Appraisal”) para determinar el nivel de madurez de una organización. Estas valoraciones no requieren autorización del SEI y este no guarda un registro formal de que empresas fueron valoradas y en que nivel. Tampoco existe un requisito de realizar valoraciones de forma periódica.

[6] La empresa valorada es PSL, de la ciudad de Medellín. La dirección del sitio web de la empresa es www.psl.com.co

2.JUSTIFICACIÓN

La mala calidad del software origina costos significativos para las organizaciones y la sociedad en general. Para comprender el problema e identificar posibles soluciones debemos realizar un análisis económico de la calidad del software.

El costo total de la calidad del software se puede dividir en dos factores, el costo de conformidad y el costo de no-conformidad[11][28]:

$C_{calidad} = C_{conformidad} + C_{no\text{-}conformidad}$

El costo de conformidad incluye los costos de prevención y los costos de detección de defectos. El costo de no-conformidad incluye los costos de fallas internas y los costos de fallas externas. Estos costos se originan al realizar las actividades del ciclo de vida del software y elaborar productos de trabajo como especificaciones, modelos de arquitectura, código fuente, código ejecutable, casos de pruebas, manuales técnicos manuales de usuario, entre otros.

La ecuación inicial de costos queda entonces así:
$C_{calidad} = C_{prevención} + C_{detección} + C_{fallas\ internas} + C_{fallas\ externas}$

Los costos de prevención son los relacionados con las actividades para prevenir que los productos de trabajo sean de mala calidad. Estas actividades se realizan al utilizar buenos métodos, técnicas y estándares para elaborar los diferentes productos de trabajo.

Los costos de detección son los relacionados con las actividades para evaluar la presencia o ausencia de las características de calidad deseadas y en el grado requerido en cada uno de los productos de trabajo del proceso. Estas actividades se conocen como Verificación y Validación. Las actividades de Verificación y Validación incluyen: pruebas de software, inspección de especificaciones, inspección de diseño, inspección de código fuente, entre otras.

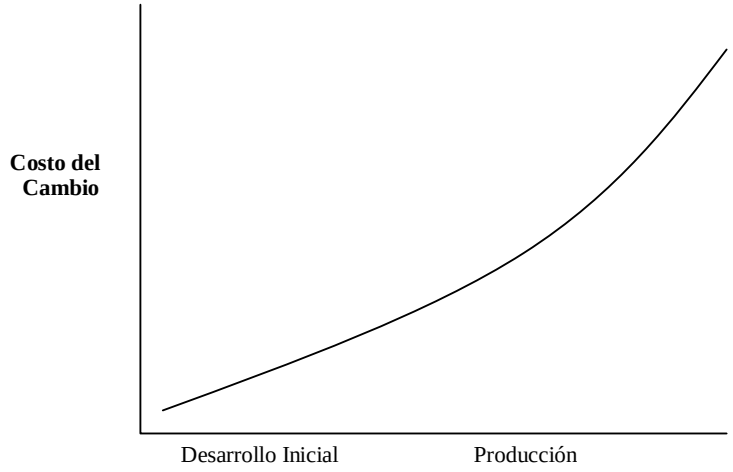
Los costos de fallas internas son los relacionados con la solución de defectos *antes* de liberar el software. Ej.: Modificar el código cuando se encuentran defectos en pruebas, modificar el diseño de la arquitectura porque no es eficiente.

Los costos de fallas externas son los relacionados con las actividades necesarias para la solución de defectos que encuentran los usuarios cuando el software ya fue liberado y se encuentra en producción. Estos costos tienden a ser los mas altos debido a que el costo de cambio en ingeniería de software aumenta a medida que transcurre el ciclo de vida (Ver figura 1). Según datos encontrados por Boehm[12], encontrar y corregir un error luego de la entrega puede costar desde 5 hasta 100 veces mas que si se hubiera detectado al realizar inspección de las especificaciones o de un modelo de la arquitectura. Los costos de fallas externas pueden ser considerados aún mas altos si se incluyen aspectos como la mala imagen para la empresa y las posibles ventas que se pierden en el futuro debido a esta mala imagen, sea de potenciales clientes nuevos o

clientes actuales que deciden cambiar de proveedor. En algunos casos, las fallas de un sistema en producción incluso originan problemas legales.

Al identificar estos factores de costo, es evidente que las actividades de Verificación y Validación (V & V) son esenciales para disminuir el costo total de la calidad. Sin un proceso adecuado de V & V, la probabilidad de que aumenten los costos de fallas externas es mayor, puesto que no se cuenta con criterios precisos y objetivos sobre la presencia de las características de calidad requeridas por los usuarios.

Figura 1. Costo del Cambio en Ing. De Software



Como proceso de verificación y validación, sin embargo, la gran mayoría de las empresas de desarrollo de software solo realiza actividades de pruebas (ejecutando el software) y un gran porcentaje lo hace de manera informal, sin la capacitación y técnicas necesarias, sin planear las actividades y sin utilizar herramientas automatizadas. Generalmente se presta poca atención a las pruebas, que son vistas como un mal necesario. Por otro lado, otras actividades tales como revisión de artefactos (Ej.: código fuente, modelos de diseño, especificaciones de requerimientos) son desconocidas para gran parte de las empresas o son realizadas de forma eventual de acuerdo al estilo de trabajo de las personas en cada equipo de desarrollo y sin que existan políticas de la organización al respecto.

El estado de informalidad actual del proceso de V & V en la mayoría de las empresas de desarrollo de software permite mantener bajos los tres primeros elementos del costo de la calidad. Es decir, se mantienen bajos los costos de prevención, detección y fallas internas.

Desafortunadamente esto no garantiza un menor costo **total** de la calidad, porque los costos de fallas externas se mantienen altos. Estos costos se evidencian cuando las

empresas deben gastar cuantiosas sumas corrigiendo defectos y prestando soporte gratuito[7].

Las iniciativas de mejoramiento del proceso de Verificación y Validación deliberadamente aumentan los costos de detección y los costos de fallas internas con el objetivo de disminuir los costos de fallas externas que tienden a ser los mas altos. De esta forma se pueden disminuir los costos totales de calidad al detectar y corregir los errores antes de liberar el software a producción.

Acceso a Mercados Internacionales

Para facilitar el acceso a mercados internacionales, el gremio de empresas de software colombiano ha identificado mejorar la madurez del proceso de desarrollo de software como un elemento esencial de la estrategia, replicando el éxito de la India. Consuelo Ardila sostiene en las conclusiones de su estudio que "la certificación de calidad es fundamental para la exportación de software. Las acreditaciones de calidad son lo que primero exigen las compañías en Estados Unidos" [2].

El Modelo CMMI[19] (la evolución del conocido Modelo SW-CMM), utilizado para valorar la madurez de las organizaciones de desarrollo de software, incluye las áreas de proceso de Verificación y Validación.

Dedicar esfuerzos al mejoramiento del proceso de V & V contribuye a la madurez de los procesos de las empresas colombianas de software y por lo tanto facilita su acceso a los mercados internacionales.

Sin embargo, los modelos de mejoramiento creados para o que incluyen lineamientos para el proceso de V & V están basados en un marco de referencia que supone organizaciones de software grandes y con gran presupuesto, típicas de los países más avanzados tecnológicamente como Estados Unidos, Japón o la Unión Europea. Ese marco de referencia dificulta el uso de estos modelos en las empresas colombianas de software, cuyos ingresos y numero de empleados promedio son mucho menores[2].

Se requiere entonces, de una interpretación apropiada de estos modelos y de las prácticas que sugieren, con respecto a las características de las empresas colombianas. Solo con esa interpretación, estos modelos pueden ser herramientas útiles para el mejoramiento de los procesos de software de las organizaciones colombianas.

[7] Las empresas contabilizan estos costos como "horas de ingeniería/consultoría no facturables"

3.OBJETIVOS

3.1 OBJETIVO GENERAL

El objetivo general de este trabajo de investigación es:

- Elaborar una estrategia para el mejoramiento del proceso de Verificación y Validación.

3.2 OBJETIVOS ESPECIFICOS

Los objetivos específicos de este trabajo de investigación:

- Realizar una síntesis del estado del arte en 4 grandes temas: Calidad de Software, el Proceso de Verificación y Validación, el Mejoramiento del Proceso de Desarrollo de Software, y Modelos de Mejoramiento del Proceso de Verificación y Validación.
- Identificar los requerimientos para una estrategia de mejoramiento del proceso de V & V a partir de la revisión del marco teórico y el contexto colombiano.
- Elaborar las prácticas de la estrategia y el programa para implementarlas
- Elaborar un paquete de transición[8] que apoye la aplicación de la estrategia
- Aplicar la estrategia en una empresa colombiana de software

[8] Un paquete de transición es un conjunto integrado de elementos que apoyan la adopción de una tecnología especifica. Un paquete de transición de un proceso incluiría formatos, plantillas, listas de verificación, entre otros. Ver sección Estrategia de Mejoramiento

4. MARCO TEORICO

4.1 INTRODUCCIÓN

Como marco teórico se presenta una síntesis de los siguientes temas:

1. Definiciones de Calidad de Software
2. El Proceso de Verificación y Validación
3. Mejoramiento del Proceso de Software
4. Modelos de Mejoramiento del Proceso de Verificación y Validación

En cada uno de los temas se realiza una revisión del estado del arte a la vez que se presentan ideas propias del autor.

4.2 DEFINICIONES DE CALIDAD DE SOFTWARE

4.2.1 ¿QUÉ ES CALIDAD?

Calidad es un termino muy común hoy en día en la literatura de administración de negocios y también de las diferentes ramas de ingeniería. Los medios de comunicación que siguen las tendencias empresariales comúnmente publican artículos sobre el tema. En número creciente, avisos publicitarios de las empresas colombianas informan al publico sobre la obtención de certificaciones en el estándar de calidad ISO 9000.

Pero qué entendemos cuando hablamos de "calidad"? La pregunta es relevante puesto que en la literatura se manejan definiciones diferentes del término [6].

Algunas de las definiciones que se encuentran son:

- **Características superiores:** Un producto o servicio es de mejor calidad si tiene mas funcionalidad, facilidad de uso, seguridad, rapidez, etc. Es decir, sus características están en un nivel superior que los productos o servicios iguales o similares[34]. Por ejemplo, una casa grande con los últimos adelantos tecnológicos y excelentes acabados será mejor que un apartamento de menor tamaño construido hace varios años y con acabados apenas aceptables. Según Plfeeger esta definición corresponde a una *visión de producto*[42].
- **Conformidad con especificaciones:** Un producto o servicio es de mejor calidad en la medida en que cumpla mejor con sus especificaciones. De manera análoga, un producto o servicio es de peor calidad si tiene defectos, los cuales se definen como variaciones o no conformidades con las especificaciones del producto o servicio. Según PLFEEGER esta definición corresponde con una *visión de manufactura*[42]. Bajo esta visión la casa grande, moderna y elegante tiene igual calidad que el apartamento de tamaño mediano, viejo y poco elegante si ambos cumplen con sus especificaciones. El estándar ISO también acepta esta definición. Según ISO 9000:2000, la calidad es: "El grado en que un conjunto de características inherentes cumple con los requisitos"[25]. Con esta definición se requiere tener excelentes

especificaciones del producto o servicio que reflejen las necesidades de los usuarios.

- **Adecuación al uso:** Un producto servicio es de mejor calidad en la medida en que se ajusta mejor a las necesidades de los usuarios[6]. Según PLFEEGER esta definición corresponde a una *visión de usuario*[42].

Sugiero la siguiente definición de calidad: "el grado en que las características inherentes de un producto o servicio cumplen o sobrepasan las expectativas del usuario en el momento en que hace uso del producto o servicio". Esta definición se basa en los siguientes supuestos:

1. Los usuarios perciben un conjunto de características inherentes al producto o servicio, las cuales evalúan. Con base en esta evaluación se forman una opinión del producto o servicio(Ej.: "Es de buena calidad", "Es de mala calidad"). Estas características varían en cada tipo de producto o servicio. Algunas características usuales de un producto cualquiera son funcionalidad, eficiencia, seguridad, facilidad de uso, presentación y conveniencia.
2. Los usuarios tienen expectativas sobre el producto o servicio que piensan utilizar o recibir. Las expectativas son lo que espera el usuario recibir. Estas expectativas dependen de varios factores pero en especial del precio que pagan por el producto o servicio y de la "calidad promedio" del producto o servicio en el mercado. Ambos aumentan las expectativas del usuario. Por un precio mayor todos esperamos mejor calidad. A su vez, cuando el estándar de calidad de un producto o servicio es mas alto, nos volvemos mas exigentes al evaluar alternativas porque sabemos cual es la "calidad promedio" que ofrece el mercado(Ej.: cuales son los beneficios mínimos que todas las aseguradoras de automóviles ofrecen).
3. Las expectativas de los usuarios aumentan al pasar el tiempo. El producto o servicio considerado de mejor calidad hace algún tiempo puede hoy ser considerado "normal" si los demás proveedores han igualado la oferta. En una economía de mercado, donde existe libertad para crear y vender productos y servicios, se esperaría que la "calidad promedio" aumente continuamente. Por lo tanto, también se esperaría que aumentarán las expectativas de los usuarios. Según Tom Peters: "No existe algo así como un producto o servicio de alta calidad. Toda calidad es relativa. Cada día, cada producto o servicio esta mejorando o empeorando, pero nunca se queda como esta"[41].

Esta definición no es completamente objetiva puesto que afirma que la calidad depende de las expectativas de los usuarios. Es por esto, que en la práctica, una de las tareas principales en el desarrollo de un producto o la prestación de un servicio, es identificar las expectativas de los usuarios y establecer definiciones objetivas, es decir, especificaciones de requerimientos. Sin embargo, es importante reconocer que la calidad es algo que evalúan las personas (sean usuarios o el mismo equipo que desarrolla o presta el servicio) por lo que no es posible desconocer que es un proceso en parte subjetivo.[9]

[9] Por esta razón la clave parece estar en aprender a escuchar muy bien a los clientes. Un recomendación frecuente de los expertos en áreas como mercadeo, ventas y servicio al cliente.

Al aplicar esta definición de calidad al software, surge la siguiente pregunta: cuales son las características inherentes de un *producto de software* cuyo grado debe igualar o superar las expectativas de los usuarios?. Para responder esta pregunta varios autores han propuesto varias taxonomías de características(o atributos de calidad) [27].La mas aceptada de estas taxonomías es el estándar ISO 9126. Este estándar afirma que las características que se deben tener en cuenta al evaluar software son: funcionalidad, confiabilidad, calidad en uso (en ingles se utiliza el termino "usability"), eficiencia, facilidad de mantenimiento y portabilidad [10]. A su vez, cada característica, agrupa varias subcaracterísticas. Por ejemplo, la característica "eficiencia" incluye las siguientes subcaracterísticas:

- Comportamiento de tiempo: velocidad de respuesta y tiempos de procesamiento promedio ("throughput") del producto de software para cumplir sus funciones
- Utilización de recursos: Cantidad de recursos utilizados y el tiempo de duración de uso, que requiere el producto de software para cumplir sus funciones
- Conformidad: Adherencia del producto de software a estándares, convenciones, leyes, protocolos, etc

Para obtener software de calidad, se debería identificar, para cada una de estas características y subcaracterísticas, cuál es el grado esperado por el usuario. Por ejemplo, aquellas aplicaciones llamadas de "misión crítica" deben tener un grado más alto de la característica "confiabilidad" en comparación con aplicaciones de oficina típicas como por ejemplo, procesadores de palabra y hojas de cálculo, que se espera que tengan un grado más alto de la característica "calidad en uso".

4.2.2 COSTO DE LA CALIDAD

El costo total de la calidad del software se puede dividir en dos factores, el costo de conformidad y el costo de no-conformidad[11][28]:

$C_{calidad} = C_{conformidad} + C_{no\text{-}conformidad}$

El costo de conformidad incluye los costos de prevención y los costos de detección de defectos. El costo de no-conformidad incluye los costos de fallas internas y los costos de fallas externas. Estos costos se originan al realizar las diferentes actividades del ciclo de vida del software y elaborar diferentes productos de trabajo como especificaciones, modelos de la arquitectura, código fuente, código ejecutable, casos de pruebas, manuales técnicos, manuales de usuario, entre otros.

La ecuación inicial de costos queda entonces así:
$C_{calidad} = C_{prevención} + C_{detección} + C_{fallas\ internas} + C_{fallas\ externas}$

Los costos de prevención son los relacionados con las actividades que buscan prevenir que los productos de trabajo sean de mala calidad.Estas actividades se realizan al

utilizar buenos métodos, técnicas y estándares para elaborar los diferentes productos de trabajo.

Los costos de detección son los relacionados con las actividades que ayudan a evaluar la presencia o ausencia de las características de calidad deseadas y en el grado requerido en cada uno de los productos de trabajo del proceso. Estas actividades se conocen como Verificación y Validación. Las actividades de Verificación y Validación incluyen: prueba de software, inspección de especificaciones, inspección de diseño, inspección de código fuente, entre otros.

Los costos de fallas internas son los relacionados con la solución de defectos *antes* de liberar el software. Ej.: Modificar el código cuando se encuentran defectos en pruebas, modificar el diseño de la arquitectura porque no es eficiente.

Los costos de fallas externas son los relacionados con las actividades necesarias para la solución de los defectos que encuentran los usuarios cuando el software ya fue liberado y se encuentra en producción. Estos costos tienden a ser los mas altos debido a que el costo del cambio en ingeniería de software aumenta de forma exponencial a medida que transcurre el ciclo de vida[10]. Según datos encontrados por Boehm[12], detectar y corregir un error luego de la entrega puede costar desde 5 hasta 100 veces mas que si se hubiera detectado al realizar inspección de las especificaciones o el diseño de la arquitectura. Los costos de fallas externas pueden ser considerados aún mas altos si se incluyen aspectos como la mala imagen para la empresa y las posibles ventas que se pierden en el futuro debido a esta mala imagen, sea de potenciales clientes nuevos o clientes actuales que deciden cambiar de proveedor. En algunos casos, las fallas en producción incluso originan problemas legales entre cliente y proveedor.

4.3 EL PROCESO DE VERIFICACIÓN Y VALIDACIÓN

Con base en los factores del costo de calidad, las actividades que tienen como objetivo asegurar la calidad del software se pueden clasificar en 3 categorías [29]:

- **Actividades Preventivas:** Son aquellas actividades que ayudan a prevenir que los productos de trabajo del proceso carezcan de las características de calidad deseadas y en el grado requerido. El factor de costo es “prevención”.
- **Actividades de Evaluación:** Son aquellas actividades que ayudan a evaluar la presencia o ausencia de las características de calidad deseadas y en el grado requerido en cada uno de los productos de trabajo del proceso. El factor de costo es “detección”.
- **Actividades de Corrección:** Son aquellas actividades que solucionan la ausencia de características de calidad deseadas y/o el hecho de que el grado

[10] Este es uno de los supuestos fundamentales de la ingeniería de software. En los últimos años, algunos autores de procesos ágiles sostienen que si se cumplen ciertas condiciones el supuesto deja de ser cierto. En este trabajo adoptamos un punto de vista de ingeniería de software tradicional y por lo tanto aceptamos el supuesto de que el costo del cambio aumenta exponencial.

requerido no sea el deseado (Ej.: "el producto no es tan fácil de usar como se quiere").El factor de costo es "fallas internas"[11].

De igual forma, las actividades de evaluación se pueden clasificar en dos categorías: actividades de Verificación y actividades de Validación. Las actividades de Verificación, según el SEI (Software Engineering Institute), tienen como objetivo "asegurar que los productos de trabajo seleccionados cumplan los requerimientos **especificados**"[19]. Los productos de trabajo son todos aquellos artefactos creados como parte del proceso de desarrollo de software y que pueden o no ser entregados al usuario final. La Verificación esta basada en la premisa de que la calidad del producto de software final es el resultado o depende en gran medida de la calidad del conjunto de artefactos elaborados durante el ciclo de vida del producto, y por lo tanto se debe verificar que ese conjunto de artefactos cumple con ciertos criterios de calidad.

Las actividades de Validación, tienen como objetivo "demostrar que un producto o un componente satisface su uso requerido en el ambiente y las condiciones en que será utilizado"[19]. La Validación busca garantizar que el producto o "la solución" ofrecida realmente es la que **necesitan** los usuarios y/o grupos afectados y que es adecuada para el "ambiente", es decir, para las características del medio (o la organización) donde se utilizará el producto. Entre las características del medio podemos mencionar: estructura, cultura de trabajo de la organización, interfaces con sistemas y procesos existentes, elementos de hardware, entre otros.

Las actividades de verificación y validación presentan similitudes [35][19]:

- Ambas pueden utilizar métodos y técnicas similares como pruebas, revisiones(formales o informales), análisis, demostraciones, simulaciones
- Ambas pueden ser realizadas a lo largo de todo el ciclo de vida del software
- Ambas pueden ser realizadas sobre el producto o un componente del producto (Ej.: un "módulo", una clase)
- Ambas pueden ser realizadas concurrentemente

A pesar de estas similitudes, las actividades de verificación y validación, tienen marcadas diferencias. Algunas son[35][19]:

VERIFICACIÓN	**VALIDACIÓN**
Enfocada a cumplimiento de especificaciones de requerimientos	Enfocada a evaluar la comprensión de necesidades, expectativas y restricciones de los usuarios finales
Enfocada a cumplimiento de estándares e interfaces especificadas	Enfocada a evaluar comprensión del contexto en que se utilizará el producto
Responde a la pregunta: Se implemento el sistema correctamente?	Responde a la pregunta: Se implemento el sistema correcto?
Los productos de trabajo a verificar son	Los productos de trabajo a validar son

[11] El lector puede observar que el factor de costo "fallas externas" no esta asociado a ninguna actividad de aseguramiento de calidad. Esto es porque los costos de "fallas externas" son posteriores a la liberación del software, cuando el software esta siendo utilizado. Las "fallas externas" implican actividades de mantenimiento correctivo, no de aseguramiento de calidad.

seleccionados basados en su contribución a cumplir objetivos del proyecto y requerimientos	seleccionados basados en que tan bien predicen que el producto logrará satisfacer las necesidades de los usuarios. “En lo posible la validación debe ser realizada utilizando el producto o componente operando en el ambiente donde se pretende implantar el producto”[19]
Ambiente y métodos de verificación de acuerdo a los productos de trabajo y a los requerimientos	Ambiente y métodos de validación deben representar el ambiente donde se pretende implantar el producto
No se requiere que los usuarios participen en las actividades	Usualmente, los usuarios participan en las actividades

Tanto las actividades de Verificación como las actividades de Validación pueden ser realizadas en las diferentes fases del ciclo de vida de un producto de software. Algunos ejemplos de actividades de verificación y validación son [37][1]:

Fase	**Verificación**	**Validación**
Especificación de Requerimientos	Inspección de requerimientos	Revisión con el usuario, Prototipos para demostración del usuario
Diseño Alto-Nivel	Inspección de diseño, Prototipos	Prototipos para demostración del usuario
Codificación	Inspección, Pruebas de Unidad	Simulación
Pruebas	Pruebas de Integración, Pruebas Funcionales, Pruebas de Desempeño	Pruebas de Aceptación, Pruebas Beta, Pruebas de Instalación (en ambiente de cliente)
Mantenimiento	Pruebas Funcionales, Pruebas de Regresión	Pruebas de Aceptación, Pruebas de Instalación (en ambiente de cliente)

El proceso de Verificación y Validación es el conjunto integrado de actividades (de ambas categorías), que utiliza una organización en sus proyectos de desarrollo de software, para evaluar la calidad del producto a lo largo de **todo el ciclo de vida**. Según el estándar IEEE 1074 para procesos del ciclo de vida, Verificación y Validación es un proceso perteneciente al grupo de “Procesos Integrales” , que son aquellos procesos que se realizan durante todo el ciclo de vida [16].

Según IEEE 1074, el proceso de Verificación y Validación incluye estas actividades(no secuenciales):

- Planear Verificación y Validación
- Ejecutar tareas de V & V
- Coleccionar y analizar métricas

- Planear Pruebas
- Elaborar Requerimientos de Pruebas
- Ejecutar Pruebas

El estándar ISO 12207, que define un estándar de procesos para el ciclo de vida del software, incluye Verificación y Validación como parte de los "procesos de soporte" que son invocados por otros procesos cuando se requiere[33].

El modelo CMMI(Capability Maturity Model Integrated) del SEI(Software Engineering Institute) incluye las áreas de proceso Verificación y Validación, en el nivel de madurez 3, o nivel "definido". Revisando los componentes de cada área de proceso(objetivos y prácticas), es posible determinar un proceso general de Verificación y Validación con las siguientes actividades:

1. Seleccionar productos de trabajo y componentes de producto
2. Determinar para cada producto de trabajo si debe ser verificado, validado o ambos
3. Especificar y crear el ambiente de verificación y el ambiente de validación o un ambiente que sirva para las dos categorías de actividades
4. Especificar procedimientos y criterios de verificación y validación
5. Realizar verificación y validación
6. Analizar resultados de la verificación y validación e identificar acciones correctivas

4.4 MEJORAMIENTO DEL PROCESO DE SOFTWARE

4.4.1 GESTIÓN DE PROCESOS DE SOFTWARE

Un proceso es una secuencia de pasos en el cual se utilizan recursos humanos, materiales y herramientas para transformar entradas en salidas o en resultados previamente definidos que agregan valor para un tercero, usualmente llamado cliente[32]. El objetivo puede ser por ejemplo crear un producto o realizar la prestación de un servicio. El proceso de desarrollo y mantenimiento de software(que en la literatura es abreviado como "proceso de software") es la secuencia de pasos que utiliza una persona, un equipo o toda una organización para crear o modificar una aplicación (o un componente) de tal forma que se logre satisfacer las necesidades de los usuarios futuros o actuales además de otros grupos afectados. La satisfacción incluye no solamente un producto de calidad, sino cumplir con los tiempos y costos establecidos y acordados con el cliente.

Los procesos integran factores como personas, procedimientos/métodos, gestión, mediciones, herramientas/maquinas, medio ambiente y recursos económicos [32][38]. Un proceso puede ser definido respondiendo las siguientes preguntas:

- **¿Qué se transforma?:** ¿Cuáles son las entradas del proceso (información, documentos, software)?
- **¿Qué resultados, salidas, productos se obtienen?:** Ej.: planes, reportes, documentos, diseños, producto manufacturado, entregables de un servicio, software, descripción de otro proceso

- **¿Cómo se transforma? :** ¿Qué pasos se debe seguir? ¿Qué métodos y herramientas se deben utilizar? :
- **¿Quién lo realiza? :** ¿Qué cargos, roles, personas intervienen?
- **¿Cuándo se inicia? :** ¿Cuándo se realiza el proceso? ¿Qué prerrequisitos deben existir?
- **¿Cuándo termina? :** ¿Qué requisitos deben cumplirse para terminar el proceso?
- **¿Con qué se realiza el proceso? :** ¿Qué recursos, herramientas, maquinas? (documentos, software, etc.)

Un proceso puede ser clasificado en alguna de estas categorías [38] [19] [46]:

Caótico,Ad-hoc: El proceso no está claramente definido. Aunque es posible que se repitan algunas buenas prácticas basadas en la experiencia, la mayoría de las veces los pasos del proceso se definen justo antes de realizarlos o de manera reactiva. No existe control sobre el trabajo, y si el proceso es realizado por un grupo de trabajo, se evidencia descoordinación, falta de comunicación y toma de decisiones injustificadas. El proceso puede lograr el objetivo planteado gracias a las capacidades individuales de las personas que lo realizan, quienes definen sus propios "procesos privados" [BACH] que algunas veces pueden ser muy efectivos y eficientes. Los integrantes del grupo de trabajo perciben autonomía y flexibilidad para realizar su trabajo(lo cual los motiva), pero también incertidumbre, desorden y necesidad de realizar esfuerzos adicionales a través de horas extras (lo cual los desmotiva) debido a la escasa o nula planeación que se realiza. El líder del grupo(Ej.: Director del Proyecto) es percibido como un héroe y un gran motivador, pero no como una persona que establece lineamientos para organizar el trabajo.

Informal: El proceso es realizado informalmente aunque están definidas ciertas actividades y productos de trabajo. Si es un grupo de trabajo, las personas saben que actividades deben seguir, aunque no han sido formalmente capacitadas en el proceso. El proceso no es planeado y tampoco se realiza seguimiento a su ejecución. La conformidad del proceso con su descripción no es evaluada. Los integrantes del grupo de trabajo perciben cierta consistencia en las prácticas que se realizan pero muchas veces olvidan realizarlas o afirman que "no hay tiempo para eso".

Controlado, bajo gestión: El proceso es planeado y se le realiza seguimiento durante su ejecución de manera cuantitativa y cualitativa. Se establecen controles que permiten alcanzar el objetivo planteado cumpliendo restricciones de tiempo y presupuesto y la conformidad con la descripción del proceso. Si el proceso es realizado por un grupo de trabajo, los integrantes perciben que el trabajo es planeado con estimaciones razonables, que existe coordinación y que se tiene claro que se debe realizar, cuando y por quiénes. Existe disciplina para cumplir compromisos y existen mecanismos para garantizar que se siga el proceso planeado. Disciplina y orden son valores muy apreciados por la organización. En el caso del proceso de desarrollo de software, según el modelo CMMI[19], este es controlado cuando se cumplen ciertos criterios(llamados objetivos o "goals") en aspectos (llamados "áreas de proceso") como: gestión de requerimientos, planeación de proyectos, control y monitoreo de proyectos, aseguramiento de calidad del producto y del proceso, gestión de la configuración, gestión de acuerdos con

proveedores (del proyecto), y medición y análisis. Adicionalmente según CMMI, el proceso debe tener atributos como:

- Conformidad con políticas organizacionales (las cuales deben estar explícitamente definidas)
- Seguir un plan establecido y una descripción
- Contar con los recursos necesarios de presupuesto, personas y herramientas
- Responsabilidades claramente definidas y autoridad de las personas para realizar el proceso
- Personas capacitadas para realizar el proceso
- Productos de trabajo bajo el nivel apropiado de gestión de la configuración
- Participación de los grupos afectados ("stakeholders")
- Monitoreo y control del proceso e identificación de acciones correctivas. Implica medición de atributos del proceso y de los productos de trabajo
- Evaluación objetiva de la conformidad del proceso con el plan establecido y de los productos de trabajo con los estándares definidos
- Revisión de actividades, estado y resultados con la alta gerencia e identificación de acciones correctivas

Definido: El proceso, en una organización, esta estandarizado de manera general, de tal forma que puede ser utilizado por diferentes tipos de proyectos, líneas de negocio o unidades organizacionales. Cada proyecto realiza una adaptación "local" de acuerdo a las características propias del proyecto (Ej.: tamaño, tecnologías a utilizar). Como el proceso es estándar, el conocimiento de las prácticas que la organización ha identificado como efectivas y eficientes esta difundido a través de toda la organización. Esto permite que las personas puedan trabajar mas fácilmente en diferentes grupos de trabajo y "moverse" a través de los proyectos y líneas de negocio. Las personas perciben la existencia de guías sobre como realizar su trabajo, aunque también la perdida de autonomía y flexibilidad. Las personas claramente identifican una o mas metodologías de la organización y existe una infraestructura dedicada a apoyar el uso de estas metodologías. Consistencia y estandarización son valores muy apreciados por la organización. El proceso de desarrollo de software, según el modelo CMMI, esta definido cuando se cumple ciertos "objetivos" en las áreas de proceso de:

- Desarrollo de Requerimientos
- Solución Técnica
- Integración del Producto
- Verificación
- Validación
- Enfoque de Procesos Organizacionales
- Definición de Procesos Organizacionales
- Capacitación Organizacional
- Gestión Integrada de Proyectos
- Gestión de Riesgos
- Análisis y Resolución de Decisiones

Adicionalmente, según CMMI, el proceso debe tener atributos como:

- Una descripción del proceso para el proyecto la cual es una adaptación del proceso estándar de la organización
- Identificación y colección de información para el mejoramiento del proceso (Ej.: mediciones, lecciones aprendidas)
- Todos los atributos de un proceso controlado

Administrado Cuantitativamente: El proceso tiene definido objetivos cuantitativos para atributos de sí mismo(Ej.: esfuerzo, costo) y para el resultado (Ej.: atributos del producto o servicio). Estos atributos son entonces medidos de forma sistemática durante todo el proceso y son analizados utilizando técnicas estadísticas. Con base en este análisis estadístico, el proceso es controlado reduciendo la variabilidad y asegurando que los atributos medidos sean predecibles dentro de un rango definido. Es posible identificar claramente las causas de variaciones debido a circunstancias excepcionales. Se identifican acciones correctivas cuando suceden variaciones excepcionales y se previene su ocurrencia en el futuro. Estabilidad y capacidad de predicción es un valor muy apreciado en la organización. El proceso de desarrollo de software, según CMMI, esta administrado cuantitativamente cuando se cumplen ciertos objetivos en las áreas de proceso de:

- Desempeño del proceso a nivel organizacional
- Gestión cuantitativa de proyectos

En optimización: El proceso se encuentra bajo mejoramiento continuo a través de la identificación de los problemas usuales que causan que el proceso tenga resultados variables. Las mejoras(Ej.: nuevas prácticas, nuevas tecnologías) son seleccionadas con base en una comprensión cuantitativa y un análisis costo-beneficio. Las innovaciones son adoptadas de forma efectiva por toda la organización. Se busca identificar las causas principales de los problemas para eliminarlos "de raíz". Mejoramiento continuo e innovación planeada son valores muy apreciados por la organización. Según CMMI, el proceso de desarrollo de software, esta en optimización cuando se cumplen ciertos objetivos en las áreas de proceso de:

- Innovación e implantación a nivel organizacional
- Análisis causal y resolución

El mejoramiento del proceso de software(MPS) consiste en el cambio **planeado** del proceso de desarrollo y mantenimiento de aplicaciones o componentes de software para apoyar el logro de los objetivos de la organización. Según Rico, MPS usualmente se realiza para lograr mejorar atributos del proceso tales como[43]:

- **Esfuerzo:** medida de la cantidad de horas que requiere el proceso.
- **Costo:** medida de la cantidad de recursos monetarios que requiere el proceso.
- **Tiempo de Ciclo:** medida de la cantidad de tiempo que requiere el proceso.
- **Calidad:** medida del número de defectos resultado del proceso[12].

[12] Esta definición de calidad corresponde con una visión de producto. Ver sección "Definiciones de Calidad"

- **Productividad:** medida de cuantas unidades (Ej.: líneas de código, puntos de función) produce el proceso
- **Capacidad de Predicción:** medida de la precisión estadística de los resultados del proceso
- **Eficiencia:** medida del número de recursos requeridos por el proceso relativos a los resultados o salidas del proceso
- **Satisfacción del cliente:** medida de que tan bien logra el proceso satisfacer las necesidades y expectativas del cliente

Rico sostiene que al mejorar estos atributos del proceso las organizaciones obtienen beneficios como:

- Aumento de ingresos
- Aumento de utilidades
- Reducción de costos
- Estabilidad
- Capacidad de predicción de resultados del proceso

Rico menciona que el campo de MPS ha evolucionado para incluir ahorros de costos, ya que "los primeros esfuerzos de MPS fueron diseñados para mejorar la calidad y confiabilidad a cualquier costo" [43]. En otras palabras, no se consideraba el costo total de calidad, sino que se buscaba disminuir solamente los factores de costo de no conformidad.

Cuando las empresas no dedican esfuerzos al MPS, dice Rico, los procesos pueden ser poco efectivos lo que usualmente tiene malos resultados como:

- Altos costos de operaciones
- Uso ineficiente de recursos
- Perdida de oportunidades en el mercado
- Falta de calidad
- Falta de satisfacción del cliente
- Baja moral en los desarrolladores

Al estudiar procesos, se debe hablar del concepto de madurez de proceso. Según el modelo CMMI, cuando un proceso utilizado por una organización puede ser clasificado como "en optimización", es porque es un proceso con un alto nivel de "madurez". En otras palabras, un proceso "maduro" se caracteriza por tener los atributos de un proceso controlado, definido, administrado cuantitativamente, en optimización además de ser efectivo. Es decir, para que el proceso se considere maduro debe ser eficaz y eficiente, además de estar definido, administrado, medido y mejorado continuamente. Según [9] la definición de madurez combina la evolución con la adopción de mejores prácticas.

El concepto de niveles de madurez se origina en los trabajos de Crosby [9] quien definió 5 niveles para la madurez en la gestión de calidad de las organizaciones. Esos 5 niveles son: "Incertidumbre", "Despertar", "Iluminación", "Sabiduría" y "Certidumbre".

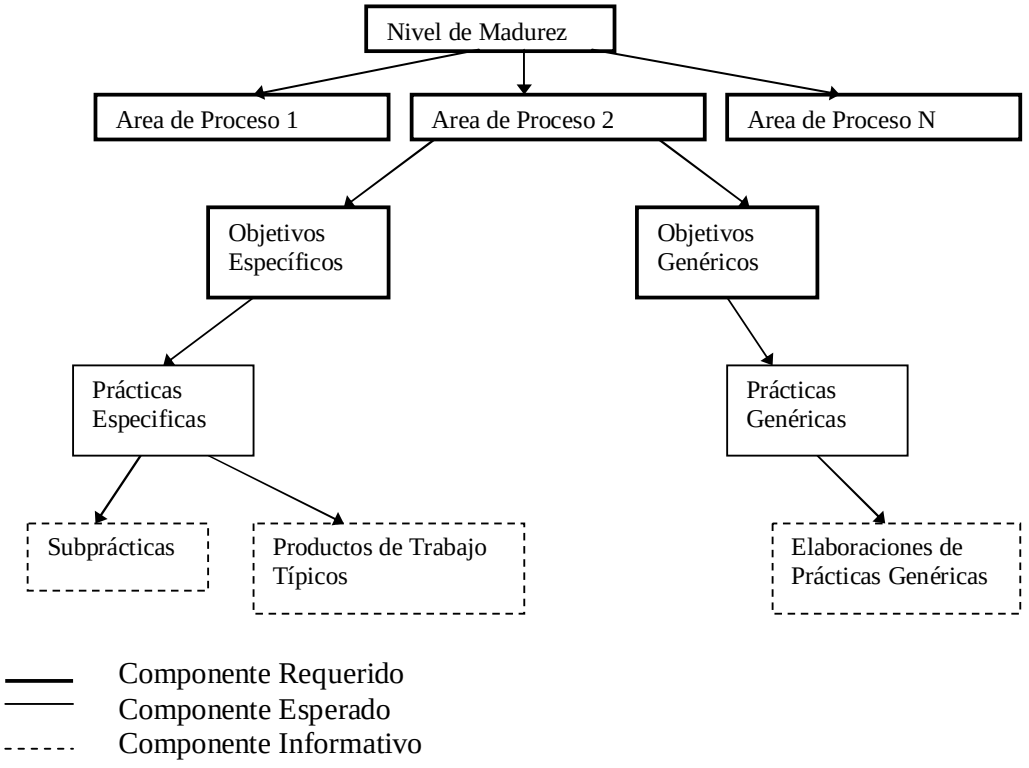
4.4.2 EL MODELO CMMI

El modelo CMMI, tal como su antecesor el modelo SW-CMM, propone mejorar el estado de la práctica de la ingeniería de software con base en los siguientes supuestos:

1. La calidad de un producto depende en gran medida de la calidad del proceso utilizado y en menor medida de las personas y la tecnología
2. Para mejorar la calidad del proceso se debe tener un proceso "maduro", es decir un proceso controlado, definido, administrado cuantitativamente y en optimización. Solo de esta forma, afirma el modelo CMMI, las mejoras en métodos, técnicas y tecnologías pueden ser adoptadas de forma efectiva ya que el problema fundamental de las organizaciones de desarrollo de software es la falta de capacidad para la gestión del proceso [39]
3. De 1 y 2 se deduce que al lograr alcanzar un proceso maduro las organizaciones pueden lograr mejorar de forma consistente la calidad de sus productos.

A partir de estos supuestos, el modelo CMMI(en la representación escalonada), define 5 niveles de madurez para el proceso de desarrollo de software. Estos niveles corresponden a las categorías previamente descritas, excepto la categoría informal. Los niveles son los componentes principales del modelo, que a su vez incluye áreas de proceso, objetivos específicos, objetivos genéricos, prácticas específicas, prácticas genéricas, entre otros componentes informativos. La estructura del modelo, que muestra las relaciones entre estos componentes, aparece en la figura 2.

Figura 2. Estructura Modelo CMMI



Tal como se muestra en la figura 2, un nivel de madurez contiene varias áreas de proceso. Dentro de cada área de proceso, existe un conjunto de criterios llamados "objetivos específicos" y "objetivos genéricos", que definen como son las características deseadas para un proceso. En cierta forma, los objetivos específicos y genéricos son los requerimientos que deben cumplir los procesos para que puedan ser valorados en un nivel de madurez dado. Los objetivos específicos son, tal como su nombre lo indica, específicos de cada área de proceso. Definen las características propias de esa área de proceso. Los objetivos genéricos, por su parte, son iguales para todas las áreas de proceso de un mismo nivel de madurez, pero solo existen para los niveles 2 y 3. Los objetivos genéricos son:

- **Objetivo genérico Nivel 2:** El proceso es institucionalizado como un proceso administrado
- **Objetivo genérico Nivel 3:** El proceso es institucionalizado como un proceso definido

Los objetivos específicos y genéricos son componentes "requeridos" en las valoraciones y se utilizan para determinar satisfacción de áreas de proceso y el nivel de madurez de una organización.

Asociados a los objetivos específicos y genéricos existen prácticas especificas y genéricas respectivamente. Cada práctica especifica y genérica esta asociada a uno y solo un objetivo especifico o genérico y son actividades que permiten satisfacer esos objetivos. Las prácticas genéricas para el objetivo genérico de nivel 2 corresponden a los atributos de un proceso de categoría "controlado". Las prácticas genéricas para el objetivo genérico de nivel 3 corresponden a los atributos de un proceso de categoría "definido".

Las prácticas especificas y genéricas son componentes "esperados", en el sentido de que describen lo que normalmente incluirían los procesos de una organización para lograr los objetivos específicos y genéricos. Sin embargo, las organizaciones pueden identificar prácticas alternativas para satisfacer los objetivos. En este sentido, las prácticas especificas y genéricas son sugerencias importantes a tener en cuenta al institucionalizar procesos para mejorar el nivel de madurez de la organización.

Las prácticas especificas y genéricas a la vez contienen componentes "informativos" que sirven de ejemplo de lo que significa una práctica. Las prácticas especificas incluyen subprácticas y productos de trabajo típicos. Las prácticas genéricas incluyen "elaboraciones de prácticas genéricas" Estos componentes informativos existen únicamente como ejemplos de implementación de las prácticas.

Es importante aclarar que las áreas de proceso no son procesos. Según CMMI[19]: "los procesos reales utilizados en una organización dependen de varios factores incluyendo el dominio de aplicación, la estructura de la organización y el tamaño. En particular, las áreas de proceso de un modelo CMMI típicamente no corresponden una a uno con los procesos utilizados en su organización". En opinión del autor, para evitar confusiones sería mejor llamarlas "áreas de mejoramiento del proceso".

El modelo, tal como a su antecesor SW-CMM [5][44], implica tener en cuenta los siguientes riesgos:

- Fue elaborado inicialmente para evaluar contratistas de desarrollo de software del departamento de defensa de Estados Unidos y por lo tanto puede ser difícil de aplicar en otro tipo de empresas. En especial, puede ser difícil de aplicar para empresas que desarrollan y venden paquetes de software estándar de forma masiva(Ej.: Microsoft), donde el tiempo de ciclo depende de las condiciones del mercado, las cuales son impredecibles.
- Tiene demasiados requerimientos y prácticas para adoptar, especialmente en los niveles iniciales (2 y 3), lo que dificultaría alcanzar logros tempranos
- Carece de un fundamente teórico formal. Es un modelo basado en la "experiencia" por lo que podría ser difícil justificar la aplicación de algunas prácticas en ciertos contextos.
- Es ambiguo, no ofrece una sola interpretación. Esto podría dificultar su uso si no se cuenta con la formación y experiencia apropiada en mejoramiento del proceso de software.
- No indica como realizar su adopción. No incluye un modelo de implementación para las áreas de proceso. Algunas organizaciones preferirían contar con una guía mas detallada.
- Puede no ser adecuado para empresas pequeñas y/o que compiten por costo o tiempos de desarrollo. Prácticas están orientadas a organizaciones y proyectos grandes.
- Podría desviar la atención de las organizaciones hacia el objetivo de alcanzar un "nivel de madurez", en vez de concentrarse en el objetivo real de mejorar el proceso de software.

4.4.3 EL MODELO SPICE

Otro modelo de mejoramiento del proceso de software, es SPICE ("Software Process Improvement and Capability Determination"). Este modelo es un estándar internacional(ISO 15504) para realizar valoraciones del proceso de software ya sea con el objetivo de realizar mejoras al proceso o evaluar a un proveedor.

SPICE esta compuesto de 9 partes[46]:

1. Conceptos y guía introductoria
2. Modelo para gestión de procesos
3. Procesos de calificación(En ingles, "rating")
4. Guía para realizar valoraciones
5. Construcción, selección y uso de instrumentos y herramientas de valoración
6. Calificación y entrenamiento de evaluadores
7. Guía para uso en mejoramiento del proceso
8. Guía para uso en determinar capacidad de proceso de proveedores
9. Vocabulario

De esas 9 partes, solo la 2,3 y 5 son “normativas”. Las demás partes son consideradas “informativas”.

En la parte 2, “Modelo para gestión de procesos”, SPICE define un conjunto de procesos de ingeniería de software. Los procesos están agrupados en las siguientes categorías:

Categoría de Proceso	**Descripción**
Cliente-Proveedor	Procesos que afectan directamente el cliente
Ingeniería	Procesos de ingeniería de sistemas e ingeniería de software
Proyecto	Procesos de gestión de proyecto
Soporte	Procesos que apoyan otros procesos
Organización	Procesos que definen objetivos de negocio y elaboran elementos de producto, procesos y recursos que ayudan a la organización a lograr sus objetivos

En la parte llamada “Modelo para gestión de procesos” existen 2 tipos de prácticas:

- **Prácticas base:** Son las prácticas que componen cada proceso
- **Prácticas genéricas:** Son prácticas que aplican para todos los procesos y que apoyan la gestión, institucionalización y mejora del proceso

Para cada proceso, SPICE mide el nivel de capacidad. Existen 6 niveles de capacidad de proceso que se numeran a partir de 0. Los niveles son:

0. **No realizado:** No se realizan las prácticas base del proceso.
1. **Realizado informalmente:** Se realizan las practicas base del proceso aunque no se realiza planeación y monitoreo. Desempeño del proceso depende de las personas. Es posible identificar productos de trabajo del proceso
2. **Planeado y Monitoreado:** El proceso es planeado y monitoreado.
3. **Bien definido:** Las practicas base se realizan de acuerdo a un proceso bien definido, adaptado de un estándar de la organización
4. **Controlado Cuantitativamente:** El desempeño del proceso es medido detalladamente. Se logra una comprensión cuantitativa del proceso
5. **Mejorado continuamente:** Se establecen objetivos cuantitativos de eficacia y eficiencia para el proceso. Se realiza mejoramiento continuo del proceso basado en mediciones del proceso

4.4.4 EL MODELO IDEAL

Para realizar el mejoramiento de procesos de forma planeada, se requiere una metodología. Un ejemplo de metodología es el modelo IDEAL elaborado por el SEI [26]. El modelo IDEAL define el mejoramiento como un proceso cíclico con las siguientes fases:

Inicio: En esta fase se estudia la relación de la iniciativa de cambio con los objetivos de la organización. Se deben establecer los objetivos de la organización que se lograrán o serán apoyados. En esta fase también se definen los grupos de trabajo y los recursos para la iniciativa de mejoramiento.

Diagnóstico: En esta fase se realiza un diagnóstico del estado actual de la organización y se modelan los procesos actuales. Con base en el diagnóstico se elaboran recomendaciones.

Establecimiento: En esta fase se planean las acciones de mejoramiento con base en las recomendaciones del diagnóstico.

Acción: En esta fase se definen soluciones a los problemas identificados en el diagnóstico y priorizados en la fase de establecimiento Las soluciones son probadas en pilotos y se refinan. Finalmente, se implementan las soluciones en toda la organización.

Aprendizaje: En esta fase se revisan las lecciones aprendidas en todo el ciclo de mejoramiento. Con base en esto se identifican mejoras para el siguiente ciclo de mejoramiento que comienza nuevamente en la fase de diagnóstico.

4.4.5 GESTIÓN DEL CAMBIO ORGANIZACIONAL

El mejoramiento del proceso de software implica un cambio organizacional[45]. Cambiar la forma como se realiza un proceso implica cambiar el comportamiento habitual de las personas que realizan el proceso. Este comportamiento habitual es causa y efecto a la vez, de la cultura organizacional, la cual determina los supuestos y creencias básicos compartidos sobre como se debe realizar el trabajo.

Para realizar mejoramiento del proceso de software de forma planeada, se requiere realizar gestión del cambio organizacional. Esta se define como “realizar los cambios de una forma sistemática o planeada y administrada”[36]. Esta definición distingue el cambio planeado del cambio “espontáneo” que ocurre de forma natural pero que no es controlado.

El cambio organizacional frecuentemente causa resistencia por parte de las personas que deben cambiar su comportamiento. Las causas de resistencia incluyen [48][14]:

- Las personas creen que los cambios no son buenos para la organización
- Las personas perciben que los cambios les traerán mas trabajo (Ej.: Escribir mas documentos)
- Las personas perciben que perderán poder o prestigio con los cambios
- Las personas perciben que los cambios no les convienen a sus intereses personales
- Las personas favorecen el status quo, son “conservadoras” y poco innovadoras.
- Las personas quieren evitar el esfuerzo del aprendizaje de nuevas habilidades y destrezas
- Las personas temen no ser capaces de adoptar los cambios o los ven como complicados

En mejoramiento de procesos de software, la causa principal de resistencia suele ser que adoptar nuevas prácticas implica mas tiempo y/o costo aunque efectivamente se logre obtener un producto de mejor calidad.

Para lograr superar la resistencia al cambio, Kotter y Schlesinger [30] proponen combinar 6 estrategias genéricas:

1. **Educación y comunicación:** Educar y comunicar a las personas sobre la necesidad del cambio
2. **Participación e involucramiento:** Involucrar a los afectados en la especificación e implementación de los cambios
3. **Facilitación y soporte:** Ayudar a las personas a adoptar los cambios
4. **Negociación y acuerdo:** Ofrecer incentivos a las personas que se quiere que adopten los cambios
5. **Manipulación y co-optación:** Involucrar a potenciales oponentes al cambio como participantes de la iniciativa
6. **Coerción implícita y explicita:** Utilizar amenazas de forma explicita o implícita para forzar a las personas a cambiar.

Diversos autores han identificado algunos principios para la gestión del cambio al realizar mejoramiento de procesos de software[24][4][45]:

- Planear y adoptar cambios de forma incremental
- Diagnósticar la situación actual
- Crear soluciones a partir de elementos existentes
- Mantener a las personas informadas sobre los cambios
- Involucrar a las personas
- Utilizar mentores y no solamente documentación escrita
- Enfocarse en problemas existentes y experiencias dolorosas del pasado debido a al proceso actual
- Experimentar informalmente, utilizar pilotos
- Cambios organizacionales deben empezar en la alta gerencia
- Evaluar y reforzar los cambios periódicamente
- Gestionar iniciativa de cambio como un proyecto

Una manera sencilla de “vender” los cambios en el proceso de software es enfocarse en los riesgos usuales de los proyectos de la organización. Sobre esta idea es bueno citar a James Bach[4]: “Si se crean procesos para riesgos imaginados antes que riesgos claros y presentes, no llegará a ningún lado. Es mas, los riesgos deben ser aceptados como tal por el equipo.[...].La manera de determinar si un riesgo es real es mirar los problemas pasados y determinar que tanto miedo tiene la organización de que un riesgo vuelva a ocurrir. La gente aprende mejor desde su propia experiencia dolorosa. Utilice esto como un motor para la evolución”.

4.5 MODELOS DE MEJORAMIENTO DEL PROCESO DE V & V

4.5.1 MODELOS DE MEJORAMIENTO

Creamos modelos para representar entidades reales o abstractas y poder entenderlas mejor. Al campo de la ingeniería le concierne crear modelos para representar problemas y ofrecer soluciones técnicas que sean de utilidad práctica para uno o mas grupos interesados (Ej.: el cliente). Un modelo de mejoramiento de procesos describe una solución al problema de cómo lograr definir e institucionalizar mejores procesos que sean eficaces y eficientes en el desarrollo de productos y la prestación de servicios para lograr la satisfacción del cliente.

Los modelos de mejoramiento de procesos de software proporcionan guías para que las organizaciones evolucionen en su manera de realizar algunas o todas las actividades de desarrollo y/o mantenimiento de software. Las mejoras en los procesos usualmente se enfocan en las tres dimensiones fundamentales del desarrollo de software: costos, tiempo y calidad. Dicho de otra forma las mejoras deben tener como objetivo ultimo mejorar nuestra capacidad de desarrollar software de mejor calidad, a menor costo y en menor tiempo.

Utilizar modelos de mejoramiento de procesos de software proporciona beneficios como[38]:

- **Establecen un lenguaje común:** Un modelo esta basado en un conjunto de premisas y define conceptos y términos que todos pueden compartir. Es decir, proporciona un punto de referencia.
- **Están basados en las mejores prácticas de la industria:** El modelo, especialmente si fue desarrollado involucrando personas de la comunidad académica y de las empresas, incorpora varias de las mejores prácticas comúnmente aceptadas.
- **Permiten realizar comparaciones con otras organizaciones o "benchmarking":** El modelo al ser un estándar, permite que una organización de software pueda evaluar su proceso y compararlo con el de otras organizaciones
- **Permiten realizar valoraciones de forma consistente:** Un modelo de mejoramiento del proceso de software define criterios específicos que pueden ser utilizados para evaluar o valorar el proceso de software de una organización.
- **Permiten priorizar las acciones:** Al realizar valoraciones, es frecuente detectar una número considerable de problemas. Los modelos de mejoramiento del proceso de software definen prioridades que pueden ser utilizadas para decidir que problemas "atacar" primero.

A su vez, al utilizar modelos de mejoramiento se debe tener en cuenta riesgos como:

- **Interpretar el modelo literalmente:** Los modelos son representaciones de la realidad que se basan en supuestos específicos no necesariamente validos en todos los contextos. Un modelo de mejoramiento debe ser interpretado según las características de la organización incluyendo sus objetivos, su estrategia y modelo de negocio, sus procesos existentes, su estructura y tamaño, y su cultura organizacional.
- **Considerar que el modelo contiene todo lo necesario:** Los modelos buscan simplificar la realidad para poder representarla. Ningún modelo de mejoramiento puede cubrir todos lo "detalles" y aspectos que se deben mejorar
- **Interpretar y adaptar el modelo incorrectamente:** Los modelos se basan en supuestos específicos y utilizan términos y conceptos que pueden ser fácilmente mal interpretados. Al utilizar un modelo de mejoramiento se debe recurrir a ayuda profesional.

Los modelos de mejoramiento del proceso de Verificación y Validación se enfocan en mejorar las actividades que utilizan las organizaciones para evaluar la calidad en el software que desarrollan y mantienen. Los modelos existentes se pueden clasificar en específicos y generales según estén dirigidos a mejorar el proceso de verificación y validación o a mejorar todo el proceso de desarrollo de software. Vale la pena aclarar que si bien la mayoría de estos modelos utilizan en sus nombres el termino "pruebas", al estudiarlos se observa que el termino es utilizado de manera general para incluir pruebas dinámicas(o de ejecución) y pruebas estáticas(o revisiones), por lo que pueden ser considerados modelos de mejoramiento del proceso de verificación y validación.

Algunos modelos de mejoramiento específicos para verificación y validación son:

- **Test Process Improvement (TPI) [29]:** Este modelo fue desarrollado por Tim Koomen y Martín Pol, basado en las experiencias de la compañía IQUIP. El modelo ofrece, no solamente un método relativamente sencillo para identificar las fortalezas y debilidades, sino que permite el mejoramiento gradual y controlado del proceso de pruebas con sugerencias muy concretas. El modelo contiene 4 componentes: áreas clave(20 en total), niveles, puntos de verificación y sugerencias de mejoramiento. Cada área clave contiene mínimo 1 y máximo 4 niveles que miden la madurez de esa área clave. En cada nivel esta definido un conjunto de criterios de verificación que permiten determinar si la organización o el proyecto se encuentra en ese nivel para esa área. Además, cada nivel describe sugerencias de mejoramiento sobre como alcanzar ese nivel.

- **Testing Maturity Model [18]:** Este modelo fue desarrollado por el Illinois Institute of Technology como una guía para el mejoramiento del proceso de pruebas y como un complemento al SW-CMM. La estructura del modelo es igual a la de CMM, por lo que también tiene 5 niveles que toman los siguientes nombres: Inicial, Definición, Integración, Gestión y Medición, Optimización. Los niveles están altamente relacionados con SW-CMM y de hecho cada nivel depende de ciertas áreas de proceso de ese modelo.

- **Test Improvement Model [21]:** Este modelo, inspirado también en SW-CMM, tiene 2 grandes componentes: un marco de trabajo compuesto por 4 niveles que define objetivos y prácticas en 5 áreas claves de mejoramiento, y un procedimiento de valoración. Los niveles que define el modelo son: Línea de Base, Efectividad en Costos, Disminución de Riesgos y Optimización. Las 5 áreas de proceso son: organización, planeación y seguimiento, casos de prueba, material/productos de trabajo de pruebas("testware") y revisiones. Según los autores, el modelo solo tiene como objetivo ayudar a los roles responsables de las pruebas en como mejorar su trabajo. Puede ser utilizado para identificar fortalezas y debilidades en las prácticas actuales y para elaborar planes de mejoramiento

Algunos modelos de mejoramiento del proceso completo de desarrollo de software que incluyen mejoras para el proceso de verificación y validación son:

- **CMMI[19]:** El modelo CMMI, que ya describimos previamente, incluye las áreas de proceso de Verificación y Validación en el nivel de madurez 3 o "definido". En este modelo Verificación y Validación son dos áreas de proceso separadas, aunque se establecen dependencias entre las dos. Los objetivos específicos para el área de proceso de Verificación son tres: "Preparar Verificación", "Realizar Revisiones de Compañeros" y "Verificar Productos de Trabajo Seleccionados". Los objetivos específicos para el área de proceso de Validación son dos: "Preparar Validación" y "Validar Producto o Componente de Producto".

- **SPICE:** El modelo SPICE, que ya describimos previamente, no incluye explícitamente un componente para verificación y validación. Sin embargo, las actividades de verificación y validación aparecen "dispersas" en las áreas de proceso de Cliente-Proveedor, Ingeniería y Soporte. Además, las revisiones de compañeros son una práctica genérica aplicable para cualquier producto de trabajo de cualquier proceso.

A continuación se describen con mas detalle los modelos TPI, TMM y CMMI. En los modelos específicos TPI y TMM se describen su estructura y componentes. En el modelo CMMI solo se describen las áreas de proceso de Verificación y Validación.

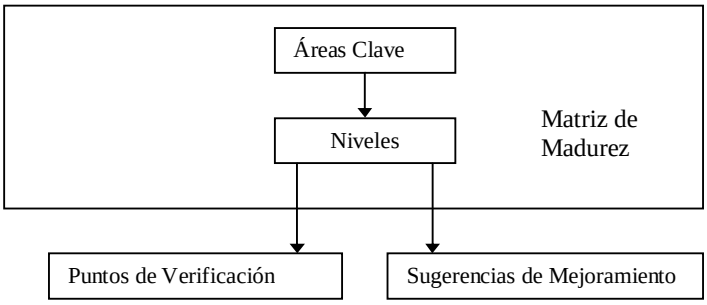
4.5.2 MODELO TEST PROCESS IMPROVEMENT(TPI)

El modelo TPI permite analizar la situación actual del proceso de pruebas y determinar sus fortalezas y debilidades. Para esto define un conjunto de áreas clave que tienen que ver cada una con un aspecto diferente del proceso de pruebas. Cada área clave contiene mínimo 1 y máximo 4 niveles (denotados con las letras A,B,C o D) que miden la madurez en esa área clave. La relación de las áreas clave y niveles se pueden ver en una "matriz de madurez de pruebas".

En cada nivel existen 2 componentes: puntos de verificación y sugerencias de mejoramiento. Los puntos de verificación (en ingles, “checkpoints”), actúan como requerimientos para alcanzar cada nivel. Las sugerencias de mejoramientos son recomendaciones para alcanzar cada nivel.

La siguiente figura muestra la estructura y componentes del modelo:

Figura 3. Estructura Modelo TPI



Fuente: Libro “Test Process Improvement” [29]

Las áreas clave que define el modelo son las siguientes:

- **Estrategia de pruebas:** La estrategia de pruebas define como se utilizan los diferentes niveles de prueba para cubrir los requerimientos y riesgos de calidad
- **Modelo de ciclo de vida:** Definición de las fases del proceso de pruebas como planeación, especificación, ejecución, entre otras
- **Momento de involucramiento:** A medida que se mejora el proceso de pruebas, estas comienzan mas temprano en el ciclo de vida
- **Estimación y planeación:** Definición de las actividades de pruebas a realizar y los parámetros de planeación como esfuerzo, recursos, costo, entre otros
- **Técnicas de especificación de pruebas:** Una técnica de especificación de pruebas define como obtener casos de prueba a partir de información de base como los requerimientos
- **Técnicas de pruebas estáticas:** Inspecciones de productos de trabajo utilizando listas de verificación
- **Métricas:** El uso de métricas permite controlar el proceso de pruebas y además evaluar potenciales nuevas mejoras
- **Herramientas de prueba:** Las herramientas de prueba permiten automatizar varias actividades del proceso de pruebas, en especial la ejecución. La automatización es particularmente ventajosa para pruebas de regresión
- **Ambiente de pruebas:** El ambiente de pruebas incluye componentes como: hardware, software, bases de datos y parámetros de configuración entre otros
- **Ambiente de oficina:** Elementos de oficina necesarios para el personal que realiza las pruebas como computadores, papelería, escritorios, impresoras, teléfonos entre otros

- **Compromiso y motivación:** El personal de pruebas debe estar motivado y comprometido. Para esto es importante que la gerencia les proporcione los recursos necesarios
- **Funciones y entrenamiento de pruebas:** El personal de pruebas debe estar capacitado en el proceso de pruebas y en el dominio de la aplicación, entre otros
- **Alcance de la metodología:** Al definir la metodología debe buscarse que sea genérica pero a la vez detallada para que pueda ser utilizada con facilidad.
- **Comunicación:** La comunicación es importante entre el personal de pruebas y con los diferentes participantes y afectados como el cliente, el director del proyecto, el equipo de desarrollo, entre otros.
- **Elaboración de reportes:** Los reportes de defectos constituyen la principal salida del proceso de pruebas. A medida que se mejora el proceso de pruebas, los reportes permiten comprender mejor la calidad del producto
- **Gestión de defectos:** La gestión de defectos permite realizar seguimiento al estado de cada defecto, desde que es asignado hasta que se resuelve
- **Gestión de testware:** Los productos de trabajo de pruebas deben ser manejados de tal forma que puedan ser reutilizados. Algunos productos de trabajo deben estar bajo gestión de configuración
- **Gestión del proceso de pruebas:** El proceso de pruebas se gestiona realizando planeación, ejecución y monitoreo del proceso
- **Evaluación:** Evaluación es la revisión de productos de trabajo diferentes al software que se ejecuta
- **Pruebas de bajo-nivel:** Las pruebas de unidad e integración son las pruebas que realizan los desarrolladores. El realizar estas pruebas adecuadamente permite detectar errores de forma temprana

A continuación se muestra la matriz de madurez que relaciona las áreas clave y los niveles:

Área Clave/ Escala	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1. Estrategia de Pruebas		A					B				C		D	
2. Modelo de Ciclo de Vida		A			B									
3. Momento de Involucramiento			A				B				C		D	
4. Estimación y Planeación				A							B			
5. Técnicas de Especificación de Pruebas		A		B										
6. Técnicas de Pruebas Estáticas					A		B							
7. Métricas						A			B			C		D
8. Herramientas de Pruebas					A			B			C			
9. Ambiente de Pruebas				A				B						C
10. Ambiente de Oficina				A										
11. Compromiso y Motivación		A				B						C		
12. Entrenamiento y Funciones de Prueba				A			B				C			
13. Alcance de la Metodología					A						B			C
14. Comunicación			A		B							C		
15. Reportes		A			B		C					D		

16. Gestión de Defectos		A				B		C						
17. Gestión de "Testware"			A			B				C				D
18. Gestión del Proceso de Pruebas		A		B								C		
19. Evaluación							A			B				
20. Pruebas de Bajo Nivel					A		B		C					

4.5.3 MODELO TEST MATURITY MODEL

Este modelo fue desarrollado por el Illinois Institute of Technology como una guía para el mejoramiento del proceso de pruebas y como un complemento al modelo SW-CMM [18]. El modelo usa el termino "pruebas" para referirse tanto a pruebas dinámicas como estáticas (revisiones).

Este modelo copia la estructura de SW-CMM, por lo que define niveles de madurez que contienen áreas de proceso.

Los niveles de madurez que define el modelo son:

Nivel de Madurez	**Descripción**
Inicial	Pruebas caóticas, proceso no definido. Pruebas y depuración se consideran lo mismo. Se libera software sin tener visibilidad de la calidad del mismo.
Definición	Proceso de pruebas definido y separado de "depuración". Pruebas son planeadas. Objetivo de las pruebas es verificar que el software cumple requerimientos
Integración	Proceso de pruebas integrado al ciclo de vida del software. Planeación de las pruebas comienza temprano en el ciclo de vida. Existe una organización de pruebas
Gestión y Medición	Proceso de pruebas bien definido y medido. Revisiones e inspecciones se realizan a lo largo del ciclo de vida. Pruebas son consideradas tanto dinámicas como estáticas.
Optimización	Proceso de pruebas en permanente optimización. Existe un proceso para evaluar y adoptar herramientas de pruebas. Objetivo de las pruebas es prevenir defectos

Nivel de Madurez	Área de Proceso
Inicial	No hay áreas de proceso
Definición	Políticas y Objetivos de Pruebas, Planeación de Pruebas, Métodos y Técnicas de Pruebas, Ambiente de Pruebas
Integración	Organización de Pruebas, Programa de Entrenamiento de Pruebas, Integración y Ciclo de Vida de Pruebas, Control y Monitoreo
Gestión y Medición	Revisiones de Compañeros, Medición de Pruebas, Evaluación de Calidad del Software
Optimización	Prevención de Defectos, Control de Calidad, Optimización de Pruebas

Los niveles de TMM están altamente relacionados con SW-CMM y de hecho cada nivel depende de ciertas áreas de proceso de ese modelo. Por ejemplo, el nivel 4 requiere que se hayan adoptado las siguientes áreas de proceso de nivel 3 y 4 de SW-CMM: Coordinación Intergrupos(Nivel 3), Revisión de Compañeros(Nivel 3), Gestión de Procesos Cuantitativa(Nivel 4), Gestión de Calidad de Software(Nivel 4).

4.5.4 ÁREAS DE PROCESO DE V&V EN CMMI

El modelo CMMI incluye las áreas de proceso de verificación y validación, las cuales se encuentran en el nivel 3 de la representación escalonada. A continuación se describen dichas áreas de proceso.

Área de Proceso Verificación

Esta área de proceso tiene como propósito "garantizar que productos de trabajo **seleccionados** cumplen los requerimientos especificados"[19]. La verificación se realiza a lo largo de todo el ciclo de vida para evaluar no solo el producto final (o un componente de este) sino los diferentes productos de trabajo. La verificación se realiza con respecto a los requerimientos.

El área de proceso espera que las verificaciones sean llevadas a cabo realizando primero actividades de preparación como la identificación de los productos de trabajo a verificar, la preparación del ambiente de verificación(Ej.: ambiente de pruebas funcionales, de desempeño) y la elaboración de procedimientos y criterios de verificación(Ej.: casos de prueba, listas de verificación). Luego de esta preparación entonces se procede a realizar las verificaciones siguiendo los procedimientos definidos y por ultimo se analizan los resultados y se identifican correcciones.

Esta área de proceso incluye explícitamente la práctica de revisiones de compañeros. En el modelo anterior, SW-CMM, la revisión de compañeros era una KPA completa, por lo que puede pensarse que el área de proceso Verificación es el reemplazo. Sin embargo, esta área de proceso considera como método de verificación no solo la revisión de

compañeros (siendo las inspecciones, la implementación mas formal) sino también las pruebas, las simulaciones y las demostraciones. En realidad, el área de proceso espera que la organización aprenda a seleccionar los métodos mas apropiados de acuerdo al tipo de producto de trabajo a verificar.

La planeación y el seguimiento de las actividades de verificación, no aparecen como objetivos o prácticas especificas, debido a que todas las áreas de proceso incluyen las prácticas genéricas "planear proceso" y "monitorear y controlar el proceso". La gestión de la configuración del "testware" tampoco aparece explícitamente ya que también existe una práctica genérica de gestión de la configuración.

A continuación se listan los objetivos específicos del área de proceso y sus prácticas asociadas:

Objetivos Específicos	**Prácticas Especificas**
1.Se realiza preparación para la verificación	1. Seleccionar productos de trabajo para verificación 2. Establecer ambiente de verificación 3. Establecer procedimientos y criterios de verificación
2. Se realizan revisiones de compañeros en productos de trabajo seleccionados	1. Preparar revisiones de compañeros 2. Realizar revisiones de compañeros 3. Analizar datos de revisiones de compañeros
3. Los productos de trabajo seleccionados son verificados con respecto a sus requerimientos especificados	1. Realizar verificación 2. Analizar resultados de verificación e identificar acciones correctivas

Área de proceso Validación

Esta área de proceso tiene como propósito "demostrar que el producto o componente de producto satisface el uso esperado en el ambiente en que se espera sea utilizado". La validación utiliza prácticas similares a las utilizadas en el área de proceso de verificación. Sin embargo, las actividades de validación se diferencian por 2 características:

1. Se busca evaluar el producto o componente de producto operando en el ambiente en que se espera sea utilizado
2. Se busca involucrar a los usuarios finales del producto o componente de producto. Por usuario final se entiende no la persona que comisiona o realiza la adquisición del producto sino quien realmente la va a utilizar(que puede o no ser la misma persona).

Se pueden realizar actividades de validación a lo largo de todo el ciclo de vida seleccionando productos de trabajo con base en su capacidad para determinar la satisfacción de los usuarios.

El área de proceso espera que las validaciones sean llevadas a cabo realizando primero actividades de preparación como la selección del producto o componentes del producto (Ej.: módulos de una aplicación), la preparación del ambiente de validación y la elaboración de procedimientos y criterios de validación. Luego de esta preparación, se realizaría la validación de acuerdo a los procedimientos definidos y finalmente se analizarían los resultados de esa validación.

Tal como en el área de proceso Verificación, la planeación y seguimiento de las actividades aparecen como prácticas genéricas.

A continuación se lista los objetivos específicos del área de proceso y sus prácticas asociadas:

Objetivos Específicos	**Prácticas Especificas**
1. Se realiza preparación para la validación	1.Seleccionar productos para validación 2. Establecer ambiente de validación 3. Establecer procedimientos y criterios de validación
2. Los productos y componentes de producto son validados para asegurar que son apropiados para su uso en el ambiente operativo esperado	1. Realizar validación 2. Analizar resultados de validación

Objetivos Específicos:

SG 1: Se realiza preparación para la validación

SG 2: El producto o componente de producto son validados para determinar si son apropiados para su uso en el ambiente operativo definido

4.6 CONCLUSIÓN

En esta sección se ha presentado una síntesis de los siguientes temas:

1. Definiciones de Calidad de Software
2. El Proceso de Verificación y Validación
3. Mejoramiento del Proceso de Software
4. Modelos de Mejoramiento del Proceso de Verificación y Validación

En las definiciones de calidad de software, se precisa que se entiende por calidad y se realiza un análisis económico de los costos de calidad.

A continuación se presenta el proceso de V&V en el contexto del aseguramiento de calidad, se describen las diferencias entre verificación y validación y los estándares existentes en la industria.

Enseguida se trata el tema del mejoramiento del proceso de software comenzando con una introducción general a la teoría de gestión de procesos, para luego continuar con una descripción de los modelos mas conocidos de mejoramiento del proceso de software: CMMI y SPICE. Finalmente, se realiza una introducción a la teoría de gestión del cambio organizacional.

Por ultimo, se presentan 3 modelos de mejoramiento del proceso de Verificación y Validación: TPI, TMM y CMMI en las áreas de proceso de V&V.

5. ESTRATEGIA DE MEJORAMIENTO

5.1 INTRODUCCIÓN

Para lograr mejorar el proceso de Verificación y Validación y por lo tanto reducir los costos totales de calidad del software en las empresas colombianas de software, se propone una estrategia de mejoramiento. Una estrategia de mejoramiento es un programa que define qué prácticas adoptar y qué orden, y que proporciona lineamientos concretos sobre cómo se deben realizar esas prácticas[13].

Previo a la descripción de la estrategia, se establece un conjunto de requerimientos a tener en cuenta. Estos requerimientos fueron identificados por el autor a partir de la revisión del estado del arte, y de las necesidades de la industria del software en Colombia. En la descripción de cada requerimiento se incluye una justificación de cada uno.

Posteriormente, se realiza la descripción de la estrategia. Primero se explica como cumple con cada uno de los requerimientos identificados y luego su estructura y componentes.

La descripción detallada de las prácticas, sin embargo, se presenta en la siguiente sección titulada “Etapas de Mejoramiento”.

5.2 REQUERIMIENTOS A TENER EN CUENTA

Para elaborar la estrategia, se estableció que debía cumplir con los siguientes requerimientos y restricciones:

5.2.1 DEFINIR PRÁCTICAS A ADOPTAR

La estrategia debe indicar explícitamente qué prácticas de Verificación y Validación debe adoptar la organización. Debe existir una descripción de las prácticas con el suficiente detalle para que pueda ser adoptada por una organización sin requerir un gran trabajo de interpretación. Los modelos de mejoramiento de procesos como CMMI y SPICE describen prácticas de manera muy general. Esto permite que los modelos sean aplicables en un mayor número de organizaciones y contextos diferentes pero también hace que sean difíciles de interpretar y adoptar. Para que la estrategia sea útil para las organizaciones se requiere que sea mas detallada que los modelos de mejoramiento.

5.2.2 ESPECIFICAR PRÁCTICAS PARA EL TODO CICLO DE VIDA

La estrategia debe incluir prácticas de V&V para todo el ciclo de vida. Si se realizan prácticas de verificación y validación desde el inicio del desarrollo, es posible disminuir los costos de fallas internas (costos de solucionar defectos luego de las

[13] En cierta forma, la estrategia también definiría que prácticas NO adoptar.

pruebas pero antes de implantar en producción), logrando disminuir aún mas el costo total de la calidad.

5.2.3 ADOPCIÓN INCREMENTAL DE PRÁCTICAS

La estrategia debe indicar cómo adoptar de forma incremental las prácticas que se proponen. Debe incluir un programa con etapas claramente definidas. Adoptar e institucionalizar cambios de forma incremental es uno de los principios claves de la teoría de gestión del cambio organizacional.

5.2.4 APOYAR OPTIMIZACIÓN DE COSTOS TOTALES DE CALIDAD

Las iniciativas de mejoramiento del proceso de Verificación y Validación deliberadamente aumentan los costos de prevención y detección(costos de conformidad). Sin embargo, estos costos no pueden aumentar de forma ilimitada, aun cuando los costos de fallas disminuyan mas, porque entonces los costos totales de calidad aumentan nuevamente(Ver figura 4). La estrategia debe incluir prácticas que apoyen la optimización de los costos totales.

5.2.5 CUMPLIR CON UN ESTÁNDAR INTERNACIONAL DE PROCESO

Para facilitar el acceso a mercados internacionales, el gremio de empresas de desarrollo de software Colombianas ha identificado la adopción de estándares internacionales como una de las estrategias principales. Para apoyar esta iniciativa de adopción de estándares internacionales de proceso, la estrategia debe utilizar como marco de referencia un estándar internacional.

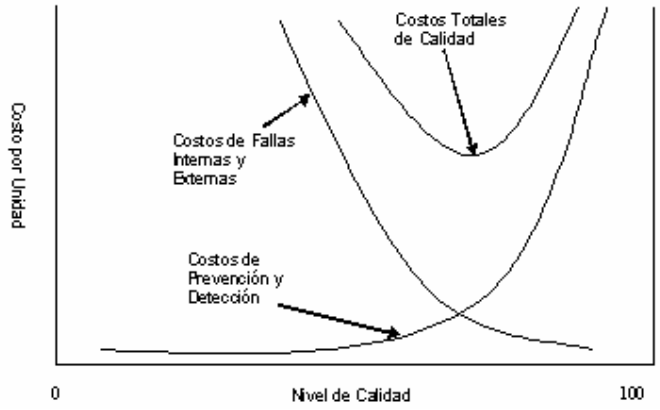
5.2.6 PROPORCIONAR INSTRUMENTOS PRECISOS DE APLICACIÓN

Los modelos de mejoramiento de procesos como CMMI y SPICE al ser generales no proporcionan instrumentos precisos para realizar las prácticas que proponen. Esto dificulta la aplicación de los modelos. Según Priscilla Fowler: “La adopción real en la practica de nuevas soluciones y productos basados en tecnología es lenta comparada con la rapidez con que las soluciones son propuestas y desarrolladas.... El conocimiento de como aplicar tecnología y producto no siempre esta disponible y las organizaciones no pueden adoptar cualquier tecnología o producto que luce atractiva”[22]. El proceso de verificación y validación es una “tecnología” relativamente nueva para la mayoría de las organizaciones que desarrollan y mantienen software. La estrategia debe proporcionar instrumentos precisos que ayuden a estar organizaciones a aplicar la “tecnología” de verificación y validación.

5.2.7 ADAPTACIÓN AL CONTEXTO

La estrategia debe estar diseñada teniendo en cuenta el contexto de uso: empresas colombianas de desarrollo y consultoría de software de tamaño mediano. Las practicas incluidas en la estrategia, y el grado de formalidad, deben ser apropiadas a las características de estas empresas, en especial el tamaño de los proyectos (en número de personas, duración y presupuesto).

Figura 4. Comportamiento de Costos de Calidad



Fuente: "Zero Defects Strategy in Software Development:The Costs of Being Bug-Free" [20]

5.3 DESCRIPCIÓN DE LA ESTRATEGIA

Con base en los requerimientos y restricciones identificados previamente se propone una estrategia de 11 prácticas organizadas en 4 etapas. Las prácticas son las siguientes:

1. Preparar Ambiente de Pruebas
2. Realizar Pruebas de Aceptación
3. Planificación del Proceso de Pruebas
4. Verificación de Software
5. Gestión del Proceso
6. Identificación de Riesgos de Calidad
7. Verificación Unidades/Componentes de Software
8. Institucionalizar Proceso Definido
9. Elaboración Plan de Riesgos de Calidad
10. Revisiones de Productos de Trabajo
11. Elaborar Prototipos de Requerimientos

Estas prácticas se organizan en un programa de 4 etapas:

1. Validación Básica

2. Verificación Básica
3. Verificación Intermedia
4. Verificación y Validación Temprana

Antes de explicar cada una de las etapas y prácticas es necesario describir como la estrategia cumple con cada uno de los requerimientos y restricciones identificadas. La explicación de la estrategia con base en los requerimientos también permite justificar la selección de las prácticas. Los requerimientos que justifican la selección de las prácticas son los requerimientos 2,4,5 y 7.

Requerimiento # 1: Definir prácticas a adoptar

La estrategia define 11 prácticas concretas que se deben adoptar para mejorar el proceso de verificación y validación. Cada práctica tiene una descripción y sugerencias de mejoramiento detalladas.

Requerimiento # 2: Especificar prácticas para todo el ciclo de vida

La estrategia incluye prácticas como revisiones de productos de trabajo(incluye especificaciones de requerimientos) y elaboración de prototipos que permiten realizar verificación y validación temprana en el ciclo de vida.

Requerimiento #3:Adopción incremental de prácticas

La estrategia propone la adopción de las 11 prácticas a través de un programa de 4 etapas.
En la descripción de cada práctica se definen dos o más metas de mejoramiento, de tal forma que cada práctica, a su vez, se adopta de forma gradual.

Requerimiento # 4: Apoyar optimización de costos totales de calidad

Las prácticas de la estrategia son de tres aspectos: verificación y validación, gestión de proceso y gestión de riesgos. Las prácticas de gestión de proceso y gestión de riesgos apoyan el control de los costos totales de calidad. En particular, las prácticas de gestión de riesgos, permite que se prioricen los esfuerzos de verificación y validación al prevenir o mitigar los problemas de calidad mas importantes. Este es un método práctico de evitar un aumento excesivo en los costos de conformidad.

El enfoque a riesgos implica que se debe realizar una evaluación de los riesgos de calidad y dar prioridad a las funciones y características de mas riesgo. No se deben verificar y validar todas las características o funciones de un producto con igual esfuerzo y/o profundidad. De hecho, algunos autores han encontrado que la ley de Pareto también se cumple para los defectos de software (“El 20% de los módulos es responsable del 80% de los defectos”)[12]. Las actividades de V & V harían parte del **plan de mitigación de riesgos de calidad** del producto[40][3], permitiendo identificar la presencia o ausencia de un riesgo.

Un riesgo de calidad, es cualquier elemento, situación o condición que puede implicar la no-satisfacción del usuario final. Por ejemplo, si se detecta que la seguridad es muy importante para los usuarios finales de la aplicación, un riesgo de calidad podría ser "el producto no verifica adecuadamente los privilegios de acuerdo al rol del usuario". Por otro lado, si se identifica que la eficiencia en el uso de recursos computacionales es un factor critico para la aceptación del producto, entonces debería considerarse un riesgo de calidad una situación como "el producto consume demasiada memoria al realizar la transacción X". El enfoque a riesgos de calidad permite que el proceso de V & V sea tanto eficaz como eficiente ya que se concentran los esfuerzos y recursos en mitigar los riesgos mas importantes.

Requerimiento #5: Cumplir con un estándar internacional de proceso

En la elaboración de la estrategia se tuvo en cuenta que las prácticas apoyaran el cumplimiento de los objetivos específicos y genéricos de las áreas de proceso Verificación y Validación del modelo CMMI. Por ejemplo, las prácticas de gestión del proceso permiten cumplir con los objetivos genéricos de nivel 3[14]. En cambio, las prácticas de verificación y validación permiten cumplir con todos los objetivos específicos en las dos áreas de proceso.

Requerimiento #6: Proporcionar instrumentos precisos de aplicación

La estrategia viene acompañada de un "paquete de transición"[15]. Un paquete de transición es una lista de materiales a ser utilizados en la introducción y adopción de nuevas tecnologías o prácticas [22]. Un paquete de transición contiene ejemplos, plantillas, listas de verificación, guías y en general cualquier elemento que facilite la adopción de nuevas tecnologías o prácticas. Priscilla Fowler, quien acuñó el termino "paquete de transición", sostiene que "un paquete de transición para cualquier tecnología de software—ya sean inspecciones de software "Fagan", diseño orientado a objetos, o una KPA de CMM como Gestión de Requerimientos(RM)—es un conjunto diseñado e integrado de componentes a ser utilizados en la introducción y aplicación de esa tecnología de software".

El paquete de transición proporcionado incluye:

- Formatos y Plantillas de ejemplo
- Procedimientos de ejemplo
- Listas de Verificación
- Materiales de capacitación sobre Verificación y Validación. Los materiales vienen en documentos y presentaciones en formato "Power Point".
- Herramientas de software libre

El paquete de transición se entrega con una interfaz web. De estas forma las organizaciones que deseen adoptar la estrategia pueden colocar un sitio en su Intranet

[14] Las áreas de proceso Verificación y Validación se encuentran en el nivel 3 de la representación escalonada del modelo CMMI.

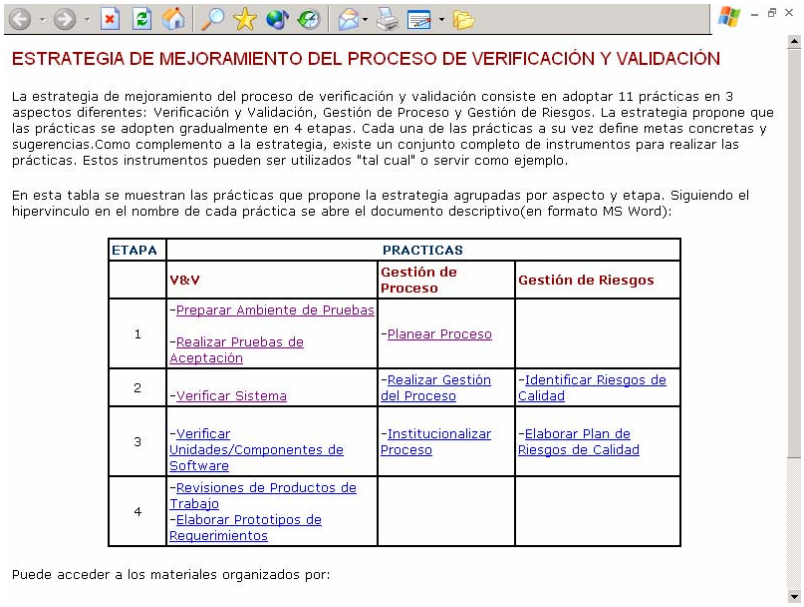
[15] El paquete de transición viene en el CD que acompaña este trabajo.

para facilitar el acceso de los participantes en el programa de mejoramiento. Las figuras 5 y 6 muestran imágenes de la interfaz web al paquete de transición.

Requerimiento #7: Adaptación al contexto

La estrategia se diseñó con criterios de simplicidad apropiados al contexto de uso esperado. Las descripciones de las prácticas son cortas (máximo 3 o 4 páginas). Los elementos del paquete de transición también fueron elaborados con estos criterios. En las formas y plantillas, por ejemplo, se evitó incluir un gran número de campos.

Figura 5. Pagina Principal Paquete de Transición



ESTRATEGIA DE MEJORAMIENTO DEL PROCESO DE VERIFICACIÓN Y VALIDACIÓN

La estrategia de mejoramiento del proceso de verificación y validación consiste en adoptar 11 prácticas en 3 aspectos diferentes: Verificación y Validación, Gestión de Proceso y Gestión de Riesgos. La estrategia propone que las prácticas se adopten gradualmente en 4 etapas. Cada una de las prácticas a su vez define metas concretas y sugerencias.Como complemento a la estrategia, existe un conjunto completo de instrumentos para realizar las prácticas. Estos instrumentos pueden ser utilizados "tal cual" o servir como ejemplo.

En esta tabla se muestran las prácticas que propone la estrategia agrupadas por aspecto y etapa. Siguiendo el hipervinculo en el nombre de cada práctica se abre el documento descriptivo(en formato MS Word):

ETAPA	PRACTICAS		
	V&V	Gestión de Proceso	Gestión de Riesgos
1	-Preparar Ambiente de Pruebas -Realizar Pruebas de Aceptación	-Planear Proceso	
2	-Verificar Sistema	-Realizar Gestión del Proceso	-Identificar Riesgos de Calidad
3	-Verificar Unidades/Componentes de Software	-Institucionalizar Proceso	-Elaborar Plan de Riesgos de Calidad
4	-Revisiones de Productos de Trabajo -Elaborar Prototipos de Requerimientos		

Puede acceder a los materiales organizados por:

Figura 6. Elementos del Paquete de Transición organizados por tipo

LISTA DE MATERIALES POR TIPO

En esta pagina esta la lista de materiales del paquete de transición organizadas por tipo.

Formatos/Plantilllas

- Ejemplo Plantilla Plan de Pruebas Aceptación
- Ejemplo Plan de Pruebas Funcionales
- Ejemplo Plantilla Plan de Pruebas Funcionales 1
- Ejemplo Plantilla Plan de Pruebas Funcionales 2
- Ejemplo Registro Pruebas de Sistema
- Ejemplo Formato Reporte de Errores
- Ejemplo Cronograma de Pruebas
- Ejemplo Formato de Casos de Prueba
- Ejemplo Formato Casos de Prueba de Unidad
- Ejemplo Casos de Prueba 1
- Ejemplo Casos de Prueba 2
- Ejemplo Lista de Riesgos de Calidad
- Ejemplo Plan Mitigación Riesgos de Calidad

Procedimientos

- Ejemplo Procedimiento Pruebas de Sistema
- Ejemplo Procedimiento Validación Prototipos

Listas de Verificación

- Lista de Verificación Especificación de Requerimientos
- Lista de Verificación Pruebas Funcionales
- Lista de Verificación Evaluación Requerimientos No Funcionales
- Lista Verificación Facilidad de Uso
- Lista de Verificación Código Fuente

Para comprender la estructura de la estrategia se debe ver como se relacionan las etapas, los aspectos de mejoramiento y las prácticas. En la siguiente matriz se puede ver estas relaciones:

	Prácticas		
Etapas /Aspecto de Mejoramiento	**V&V**	**Gestión de Proceso**	**Gestión de Riesgos**
Validación Básica	-Preparar Ambiente de Pruebas -Realizar Pruebas de Aceptación	-Planificación del Proceso de Pruebas	
Verificación Básica	-Verificación de Software	-Gestión del Proceso	-Identificación de Riesgos de Calidad
Verificación Intermedia	-Verificación Unidades/Componentes de Software	-Institucionalizar Proceso Definido	-Elaboración Plan de Riesgos de Calidad
Verificación y Validación Temprana	-Revisiones de Productos de Trabajo -Elaborar Prototipos de Requerimientos		

Las filas de la matriz indican las etapas mientras que las columnas indican los aspectos de mejoramiento. En cada celda se encuentran las prácticas correspondientes a una etapa y un aspecto de mejoramiento especifico.

DESCRIPCION ASPECTOS DE MEJORAMIENTO

Tal como mencionamos en la sección anterior, en la estrategia existen tres aspectos de mejoramiento:

1. Mejoramiento de las actividades de verificación y validación: El orden de las actividades de verificación y validación en las 4 etapas esta basado en el supuesto de que se debe comenzar a mejorar las pruebas de alto nivel (aceptación y de sistema) para luego continuar con las pruebas de bajo nivel y las revisiones e inspecciones de productos de trabajo. Este supuesto esta basado en el modelo TPI. Según los autores del modelo: “En un proceso de pruebas inmaduro el estimulo por el mejoramiento usualmente viene de los usuarios y gerentes de sistema…Los niveles de madurez de las áreas clave están conectadas al hecho de que en la práctica una organización usualmente comienza a enfocarse en las pruebas de alto nivel. Inicialmente los niveles del modelo están concentrados mas en estos niveles de prueba mientras que en un proceso de pruebas mas maduro estos están dirigidos a pruebas de bajo nivel, evaluaciones (**Nota:** Los autores de TPI utilizan el termino “evaluación” para referirse a revisiones técnicas

e inspecciones) y especialmente la integración y afinamiento entre las diferentes clases de prueba y niveles de evaluación"[29].

2. Mejoramiento hacia un enfoque de gestión de riesgos (de calidad): Se incluyeron prácticas de gestión de riesgos como un método práctico para evitar un aumento excesivo en los costos de conformidad(costos de prevención y costos de detección). En particular, las prácticas de gestión de riesgos, permite que se prioricen los esfuerzos de verificación y validación al prevenir o mitigar los problemas de calidad mas importantes.

3. Mejoramiento de las prácticas de gestión de proceso: Las prácticas de gestión de procesos permiten control del proceso y establecen una base para el mejoramiento continuo al institucionalizar un proceso definido de verificación y validación

Prácticas de verificación y validación

Para el mejoramiento de las actividades de verificación y validación se incluyen las siguientes prácticas:

Preparar Ambiente de Pruebas: Con esta práctica la organización aprende a preparar ambientes de pruebas diferentes de los ambientes de desarrollo y de producción. El ambiente de prueba, por ejemplo tiene su propia base de datos y utiliza una copia del software que esta en desarrollo. Los elementos necesarios para configurar el ambiente de pruebas son definidos previamente y preparados.

Realizar Pruebas de Aceptación: Con esta práctica la organización aprende a planear, diseñar y ejecutar pruebas de aceptación con el cliente y los usuarios de la aplicación.

Verificación de Software: Con esta práctica la organización aprende a planear, especificar y ejecutar pruebas de sistema de la aplicación

Verificación Unidades/Componentes de Software : Con esta práctica la organización aprende a realizar pruebas de los programas/componentes/unidades del software.

Revisiones de Productos de Trabajo: Con esta práctica la organización aprende a revisar productos de trabajo intermedios de la metodología de desarrollo utilizada. Ej: Especificaciones de Requerimientos, Modelos de Diseño, Código Fuente.

Elaborar Prototipos de Requerimientos: Con esta práctica la organización aprende a realizar prototipos con el propósito de validar el análisis y especificación de requerimientos.

Prácticas de gestión de riesgos

Las prácticas de mejoramiento hacia un enfoque de gestión de riesgos incluyen:

Identificación de Riesgos de Calidad: Con esta práctica la organización aprende a identificar y priorizar riesgos de calidad de la aplicación. Los riesgos solo se consideran como "aspectos a tener en cuenta" pero no se identifican acciones de mitigación. Esta identificación se realiza justo antes de las pruebas.

Elaboración Plan de Riesgos de Calidad: Con esta práctica la organización aprende a identificar acciones de mitigación para los riesgos de calidad, principalmente a través de pruebas. Se comienza a producir un cambio de mentalidad sobre lo que son las pruebas. Este plan se debe elaborar por lo menos antes de las pruebas de sistema pero preferiblemente en etapas mas tempranas como la especificación de requerimientos.

Prácticas de gestión del proceso

Las prácticas de gestión del proceso son:

Planificación del Proceso de Pruebas: Con esta práctica la organización aprende a planear las pruebas. Las actividades de pruebas son incluidas explícitamente en el tiempo del proyecto. Se identifican los recursos y se estima el tiempo de las pruebas.

Gestión del Proceso: Con esta práctica la organización aprende a realizar seguimiento y control al proceso a través de mediciones y revisiones formales frente a lo planeado.

Institucionalizar Proceso Definido: Con esta práctica el proceso de V & V esta formalmente definido. Se proporcionan recursos y se capacita a las personas para realizar el proceso. La existencia de un proceso definido permite contar con una línea de base para mejorar el proceso continuamente

LA ESTRATEGIA Y EL COSTO TOTAL DE CALIDAD

Al examinar las prácticas vale la pena revisar que efecto tienen en el costo total de calidad que es analizado en la sección 4.2.2 ("Costo de la calidad"). Las prácticas de verificación y validación son un factor de costo de evaluación o detección. Al aplicar la estrategia las empresas deliberadamente aumentan este costo. Sin embargo, el comportamiento del costo de calidad (ver figura 4, "Comportamiento de Costos de Calidad") implica que al aumentar los costos de prevención y/o evaluación/detección, comienzan a disminuir los costos de fallas internas y externas (el costo de no conformidad). Dado que inicialmente, los costos de no-conformidad son mas altos que los costos de conformidad, al comenzar a disminuir los primeros, disminuye también el costo total de calidad.

Naturalmente, aumentar ilimitadamente los costos de conformidad eventualmente conduce a aumentar de nuevo el costo total de calidad(ver figura 4, "Comportamiento de Costos de Calidad"). Por esta razón, se incluyen prácticas de gestión de proceso y gestión de riesgos de calidad. Las primeras permiten el control y medición del proceso,

los cuales son necesarios para determinar los costos de conformidad. Las segundas permiten lograr eficiencia ya que las organizaciones concentran sus recursos en los requerimientos y componentes mas importantes. Según investigaciones de Boehm [12], la Ley de Pareto también se cumple para los defectos de software. Es decir, en la mayoría de los casos "el 20% de los módulos/componentes es responsable del 80% de los defectos". Por consiguiente, intentar verificar y validar todas las partes del software por igual, produce rendimientos decrecientes, donde a medida que invierto mas tiempo y esfuerzo, menores beneficios obtengo. El enfoque a riesgos de calidad, permite concentrar los recursos en ese 20% critico donde se obtienen las grandes reducciones en el costo total de calidad.

5.4 CONCLUSIÓN

En esta sección se realiza una descripción de la estrategia de mejoramiento propuesta. Inicialmente, se define un conjunto de requerimientos a tener en cuenta. En la siguiente sección se describe la estrategia, explicando cada una de sus etapas.

6.ETAPAS DE MEJORAMIENTO

6.1 INTRODUCCIÓN

En esta sección se explican cada una de las 4 etapas de la estrategia y las prácticas definidas en cada una. Para cada práctica se definen metas de mejoramiento y sugerencias concretas sobre como implementarla. En el texto de las prácticas se realizan referencias en letra itálica a elementos del paquete de transición que acompaña la estrategia.

6.2 ETAPA 1: VALIDACIÓN BÁSICA

6.2.1 PREPARAR AMBIENTE DE PRUEBAS

El ambiente de pruebas es el conjunto de elementos necesarios para realizar las pruebas de la aplicación. Cuando se cuenta con un ambiente de pruebas separado se pueden realizar pruebas de forma independiente del equipo de desarrollo.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Preparar ambiente de prueba controlado
2. Preparar ambiente de prueba similar a producción

Metas:

1. **Preparar ambiente de prueba controlado:** Asegure que todos los proyectos de software tengan disponible o establezcan un ambiente de pruebas separado del ambiente de desarrollo. Para esta mejora considere las siguientes recomendaciones:

- Identifique claramente la versión o fecha del software que se va a copiar en el ambiente de pruebas. Por ejemplo, en la carpeta donde guarde el software coloque la fecha o la versión como parte del software (“ACME_V0.8”, “ACME_12_NOV_2003”). Al identificar la versión o fecha del software, se puede saber qué errores corresponden a qué programa.

- Identifique o solicite la configuración del ambiente de prueba utilizado. De acuerdo a la aplicación que va a probar considere: Sistema Operativo y versión, actualizaciones (“Services Packs” o parches) y Especificaciones de Hardware(procesador, memoria, disco, tarjetas de video y sonido, tarjeta de red, dirección IP), Motor de Base de Datos utilizado y versión, entre otros. Documente el ambiente de prueba en el plan de pruebas del software(*Ver ejemplo de plantillas de planes de prueba en el paquete de transición*)

- Utilice un esquema de base de datos con uno o mas usuarios de pruebas. Ejecute los scripts de creación de objetos con estos usuarios, de tal forma que el personal de pruebas trabaje con sus propios datos. Si es posible, utilice una base de datos diferente para las pruebas.

- Durante el tiempo en que se realicen las pruebas trate de no modificar ningún parámetro de la configuración del ambiente de pruebas (elementos de hardware, nuevas aplicaciones o nuevas versiones de aplicaciones existentes). Si requiere modificar el ambiente de pruebas, el personal de pruebas, el Director de Pruebas o Director de Proyecto debe ser informado. Considere documentar los cambios realizados. De esta forma si el software presenta problemas puede identificarse el cambio de la configuración como una posible causa.

- El ambiente de prueba debe incluir datos de prueba. Al preparar el ambiente de prueba defina datos de prueba suficientes para realizar una prueba real. Durante la preparación del ambiente de prueba realiza la carga de datos paramétricos y tablas de base de datos atribuitivas. Considere utilizar datos de los clientes para que se trabaje con información real.

2. **Preparar Ambiente de Pruebas Similar a Producción:** Asegure que los proyectos tengan un ambiente de pruebas con iguales o similares especificaciones de hardware y software. Para la mayoría de las organizaciones será muy costoso poder replicar cada ambiente de producción de un cliente. Sin embargo, es perfectamente viable recrear algunos de los elementos del ambiente de producción (Ej: el sistema operativo). Para esta mejora considere las siguientes recomendaciones:

   - Trate de identificar con el cliente de forma temprana el ambiente de producción donde se implantará el software. Esta información puede ser obtenida en las reuniones iniciales del proyecto o puede ser parte de un documento de especificación o de términos de referencia. La identificación temprana permite que el Director de Proyecto pueda negociar la disponibilidad de recursos con el área encargada de la administración de los recursos informáticos (“Dpto de Sistemas, Tecnología, Informática”). Si el proyecto tiene amplios recursos o el cliente es importante podrían tomarse decisiones como adquirir nuevos equipos o rentar temporalmente algunos. Es posible que muchas veces el cliente no haya tomado la decisión de cómo será el ambiente de producción. En este caso pueden identificarse elementos parciales de cómo será el ambiente de producción o rangos (Ej: “El tamaño de la memoria puede ser entre 256 MB y 1024 MB). La identificación temprana también permite saber cuestiones como cuantos usuarios utilizarán concurrentemente la aplicación y que funcionalidad será la mas importante para los usuarios.

- Cuando sea posible, trate de documentar la especificación del ambiente de producción como parte del documento de especificación del sistema o documento de especificación de requerimientos. Realizar esta práctica permitiría que el equipo de desarrollo conociera desde el inicio como sería el ambiente de producción en especial, las características de hardware, pero también de los usuarios.

6.2.2 REALIZAR PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son las pruebas que realizan los usuarios (o un representante de estos) en la organización del cliente para determinar si el software puede ser aceptado y puesto en producción. Las pruebas de aceptación necesariamente implican que el cliente tenga presente ciertos criterios de aceptación. Estos criterios preferiblemente están explícitos en algún tipo de documento.
Dentro de la organización del cliente, si existe alguna área o departamento encargado de la administración de los recursos informáticos y de las aplicaciones, puede decidir realizar también pruebas de aceptación para evaluar características de calidad como la portabilidad (se puede instalar en el ambiente de producción establecido?), la seguridad(cumple las políticas de seguridad de la organización?) además de verificar los procedimientos de instalación.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Realizar Pruebas de Aceptación
2. Realizar Pruebas Beta

Metas:

1. **Realizar Pruebas de Aceptación:** Asegure que en todos los proyectos el cliente realiza pruebas de aceptación antes de poner el software en producción. Para esta mejora considere las siguientes recomendaciones:

- Acuerde con el cliente desde el inicio del proyecto la realización de pruebas de aceptación e inclúyalas en el cronograma del proyecto
- Ayude a los usuarios a realizar las pruebas de aceptación proporcionándoles una guía de las características a probar
- Ayude a los usuarios a realizar las pruebas de aceptación proporcionándoles plantillas para elaborar escenarios/casos de pruebas.
- Informe al cliente de la importancia de las pruebas de aceptación como mecanismo formal para lograr la entrega final del software
- Incluya una actividad llamada “Pruebas de Aceptación” en su metodología de desarrollo
- Diseñe mecanismos en contratos o acuerdos de servicio que incentiven la realización de pruebas de aceptación y alguna forma de penalización por no realizar las pruebas

- Defina un formato de pruebas de aceptación. Incluya criterios de aceptación. Incluya también campos para firmas del usuario y del desarrollador
- Utilice herramientas de software para automatizar las pruebas de aceptación *(Vea herramientas de software libre para pruebas de aceptación en el paquete de transición)*
- Utilice un plan de pruebas de aceptación para definir actividades, recursos y responsabilidades (*Ver un ejemplo de una plantilla de plan de pruebas de aceptación en el paquete de transición*)

2. **Realizar Pruebas Beta:** Asegure que en todos los proyectos el cliente realiza pruebas de uso del software por un período de tiempo considerable (Pruebas Beta). Las pruebas beta deben ser realizadas directamente por los usuarios de forma natural, es decir, los usuarios utilizan la aplicación normalmente como un piloto y reportan los errores. Para esta mejora considere las siguientes recomendaciones:

- Acuerde con el cliente desde el inicio del proyecto la realización de pruebas beta e inclúyalas en el cronograma del proyecto
- Durante la realización de las pruebas beta, proporcione asistencia a los usuarios en el uso de la aplicación
- Defina el tiempo de pruebas beta de tal forma que permita probar funciones que dependan de eventos(Ej: “cierres mensuales de contabilidad”, “cierres mensuales de nomina”, “cierres diarios de fondos de valores”) y/o de procesar información de períodos de tiempo
- Evalué con los usuarios la facilidad de uso de la aplicación para determinar posibles modificaciones en la interfaz de usuario. Las modificaciones se pueden planear de una vez o para una próxima versión.
- Si su organización tiene una línea de reporte de errores (“Help Desk”) solicítele al cliente que reporte los errores encontrados a través de ese medio.

6.2.3 PLANIFICACIÓN DEL PROCESO DE PRUEBAS

Planear un proceso implica determinar de antemano lo que se requiere para realizar dicho proceso. Planear requiere identificar las actividades y recursos necesarios para lograr un objetivo. La planeación de un proceso puede realizarse como parte del plan general de un proyecto o puede realizarse de manera independiente. Igualmente, la formalidad de la planeación puede variar desde un simple cronograma realizado por el director o líder de un proyecto hasta un documento con múltiples secciones que es revisado y aprobado por varias personas.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Incluir Pruebas en Cronograma del Proyecto
2. Elaborar Cronograma Detallado de Pruebas
3. Elaborar Plan de Pruebas Formal

Metas:

1. **Incluir Pruebas en Cronograma del Proyecto:** Asegure que todos los proyectos definan **explícitamente** tiempo y fechas para realizar las pruebas dentro del cronograma del proyecto. Para esta mejora considere las siguientes recomendaciones:

- Considere el tiempo para solucionar errores y que este tiempo sea diferente del tiempo de las pruebas. Idealmente en las pruebas no se encontrarían errores, pero es algo muy improbable.
- Considere el tiempo para "re-probar" el software luego de que se corrigieron los defectos. Es realista incluirlo en el cronograma del proyecto.
- Tenga en cuenta el tamaño del proyecto, a la hora de definir las pruebas. Defina el numero de horas para pruebas como un porcentaje del tiempo total del proyecto (Ej: 10%, 15%, 20%, 25%). Es lógico que proyectos mas grandes requieran mas tiempo para pruebas.
- Tenga cuidado en definir un limite máximo de tiempo las pruebas. Si se define demasiado tiempo es bastante probable que no sea cumplido porque el tiempo para desarrollo usualmente es subestimado. Trate de definir un porcentaje de tiempo que sea "realista".

2. **Elaborar Cronograma Detallado de Pruebas:** Asegure que los proyectos definan un cronograma explicito para pruebas con tiempos y fechas. Para esta mejora considere las siguientes recomendaciones:

- Organice el cronograma de pruebas de acuerdo a la funcionalidad de la aplicación o los procesos que se realizan con la aplicación((*Ver un ejemplo de cronograma de pruebas en el paquete de transición*) Los nombres de las tareas serían "Probar opción/característica/proceso X" La funcionalidad mas importante (para el cliente) sería probada en las fechas iniciales
- Si tiene definido casos /escenarios de prueba también puede organizar el cronograma de acuerdo a estos casos.

3. **Elaborar Plan de Pruebas Formal:** Asegure que los proyectos elaboren un documento de plan formal de pruebas. Este documento puede contener:

    - Características/Funciones que se probarán
    - Características/Funciones que NO se probarán
    - Especificación del ambiente de prueba(reqs. de hardware y software)
    - Entregables Esperados
    - Cronograma (puede ser una referencia)

Tenga en cuenta las siguientes recomendaciones:

- Como la elaboración de documentación escrita usualmente es resistida por los desarrolladores, considere inicialmente adoptar un documento de plan mas "liviano" con solo el cronograma y alguna de las secciones mencionadas. Con el tiempo puede ir agregando otras secciones según los resultados de la práctica(*Ver ejemplos de plantillas de planes de prueba en el paquete de transición*)

6.3 ETAPA 2: VERIFICACIÓN BÁSICA

6.3.1 VERIFICACIÓN DE SOFTWARE

La verificación de software en esencia consiste en evaluar si el producto final de software cumple con las especificaciones. Verificar implica determinar corrección con base en una referencia. Esta referencia son las especificaciones que definen lo que **"debe ser o hacer"** el software. Ese "**deber ser o hacer** " se compara con lo que "es", típicamente ejecutando el software para observar su comportamiento proporcionándole ciertas entradas y observando los resultados o salidas. La verificación del software es realizada por la organización o el grupo que desarrolla el software y no por el cliente. Esta verificación tiene como objetivo evaluar la calidad del software para determinar si puede ser liberado o si se requiere realizar ajustes.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Realizar Pruebas Funcionales de forma exploratoria
2. Realizar Pruebas Funcionales con casos de prueba

Metas:

1. **Realizar Pruebas Funcionales de forma exploratoria** [7]**:** Pruebas exploratorias es un método informal de realizar pruebas. Consiste en realizar sesiones de pruebas con objetivos bien definidos (probar una característica de la aplicación, probar un proceso de negocio completo, probar la facilidad de uso, probar la seguridad). Al final de cada sesión se presenta un reporte o informe donde se documentan los errores encontrados y opcionalmente posibles problemas potenciales(Riesgos) de la aplicación. El reporte o informe puede estar estandarizado o puede ser ad-hoc. Para esta mejora considere las siguientes recomendaciones:

- Es importante que el personal de pruebas comprenda el dominio de la aplicación, es decir del tipo de negocio al que esta dirigido. Si la persona que

realizará la prueba carece de ese conocimiento, debe realizarse una capacitación o ser asistido por alguien experto

- Es importante que el personal de pruebas conozca la aplicación antes de realizar las pruebas. El personal debe poder comprender cómo funciona el software, cómo se acceden a las diferentes funciones y características, cómo se parametriza, etc.
- La única documentación que se produce con el método de prueba exploratoria es el reporte de los errores encontrados. Discuta con el personal de pruebas y con el personal de desarrollo si el contenido de este reporte debe ser estándar. Si el formato se define como un estándar, revise periódicamente el contenido y la forma como se están describiendo los errores.

2. **Realizar Pruebas Funcionales con casos de prueba:** Las pruebas funcionales con casos de prueba consiste en realizar las pruebas con base en un conjunto de casos de prueba previamente diseñados. Un caso de prueba es una combinación especifica de entradas, condiciones y resultados o salidas esperadas que se ejecutan para verificar alguna funcionalidad del software. Los casos de prueba pueden ser especificados en un formato estándar que incluya las entradas y condiciones y el resultado esperado. Si al ejecutar el software se encuentra que el resultado es diferente del esperado, entonces esto indicaría un posible defecto. Realizar Pruebas Funcionales con casos de prueba es un proceso de tres pasos: 1) Diseñar los casos de prueba, 2) Ejecutar el software con las entradas y condiciones especificadas de los casos de prueba y 3) Reportar los posible errores. Para adoptar esta mejora considere las siguientes recomendaciones:

   - Es importante que el personal que diseñe los casos de prueba tenga conocimiento del el dominio de la aplicación, es decir conozca sobre el tema y/o del tipo de negocio al que esta dirigido el software. Si la persona que va probar carece de ese conocimiento, debe realizarse una capacitación o ser asistido por alguien experto
   - Si existen especificaciones, es recomendable que los casos de prueba estén basados dichas especificaciones. De todas maneras, es bueno revisar la aplicación de manera “exploratoria”, para pensar como diseñar los casos de prueba.
   - Los casos de prueba diseñados deben combinar o valores de entrada y condiciones tanto correctas como incorrectas. Los valores y condiciones incorrectas están dirigidas a verificar si el software valida correctamente las entradas y puede manejar los errores del usuario. (*Ver ejemplos de casos de prueba funcionales en el paquete de transición*)
   - A medida que el personal de pruebas vaya ganando experiencia, considere entrenar al personal de pruebas en técnicas de diseño de casos de prueba como partición equivalente y valores límite (*Ver documento guía sobre diseño de casos de prueba funcionales en el paquete de transición)*

- Utilice el método taguchi para determinar los casos de prueba *(Ver documento sobre método taguchi en el paquete de transición)*

6.3.2 GESTION DEL PROCESO

La gestión del proceso consiste en planear el proceso, realizar seguimiento y control al proceso con base en el plan y realizar ajustes si es necesario. La gestión del proceso debería realizarla el líder o director del proyecto, o sí existe un equipo o departamento de pruebas el líder o director de dicho grupo o departamento.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Realizar seguimiento y control a actividades de pruebas
2. Realizar seguimiento y control con mediciones

Metas:

1. **Realizar seguimiento a actividades de pruebas:** Inicialmente asegure que los proyectos realicen seguimiento a las actividades de pruebas. Para realizar este seguimiento siempre debe existir un plan o al menos un cronograma (ver práctica Planificación del Proceso en la Etapa 1). La persona encargada de coordinar las pruebas (usualmente el director de proyecto o un director de pruebas) debe adquirir el habito de verificar el cumplimiento de los compromisos de pruebas. Para esta mejora tenga en cuenta las siguientes recomendaciones:

- La revisión del estado de las actividades de pruebas debe realizarse a través de reuniones previamente planeadas. En estas reuniones debe utilizarse el plan o cronograma de pruebas para evaluar el cumplimiento
- Al realizar seguimiento a las actividades de pruebas, cuando existe un desfase con respecto al plan, siempre existen dos alternativas: actualizar el plan o realizar las actividades desfasadas (lo que implicaría trabajar mas rápido o mas horas diarias). Es importante que se consideren siempre estas dos alternativas para tomar la mejor decisión. En algunos casos, actualizar el plan puede ser la opción mas realista y apropiada para el éxito del proyecto.
- Es importante informar y/o acordar las modificaciones al plan de pruebas con el cliente y otros grupos participantes o afectados de alguna forma por el proyecto.

2. **Realizar seguimiento con indicadores de gestión:** Asegure que el seguimiento y control de las pruebas sea realizado con mediciones sobre el

proceso. Las mediciones deben indicar el desempeño en la ejecución del proceso de las pruebas de la aplicación. Algunas mediciones sugeridas son:

- # Casos de pruebas ejecutados / # Casos de pruebas planeados
- Valor ganado en actividades de pruebas
- Cubrimiento de pruebas de la aplicación (# requerimientos probados/ # total de requerimientos, # opciones y características probadas/# total de opciones y características)

Para esta mejora tenga en cuenta las siguientes recomendaciones:

- Para la gestión del proceso defina mediciones relacionadas con el proceso de pruebas y no con la calidad de la aplicación (tales como # de defectos encontrados, etc. Es importante recordar que lo que se quiere medir es el desempeño de las actividades de pruebas y no la calidad del producto. Las mediciones sobre calidad del producto pueden ser un resultado del proceso de pruebas pero como tal no indican el desempeño frente a lo planeado.
- Las mediciones que se definan deben ser fáciles de obtener. Defina mediciones simples que permitan determinar el estado de las pruebas y la toma de decisiones.

6.3.2 IDENTIFICACIÓN DE RIESGOS DE CALIDAD

La identificación de riesgos de calidad consiste en identificar las situaciones y problemas potenciales que de ocurrir afectarían la calidad del producto (o también la calidad del servicio que se presta). Identificar los riesgos mas probables de ocurrir y con mayor impacto permite que se prevengan problemas en la calidad del producto. También permite que se dedique el escaso tiempo y recursos a evaluar aquellas características, funciones, módulos y/o elementos con mayor riesgo de fallar y de afectar la satisfacción de los usuarios. Esta lista de riesgos se utiliza como entrada para realizar pruebas comenzando con los riesgos mas importantes.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Elaborar Lista de Riesgos de Calidad (solo Funcionalidad)
2. Elaborar Lista de Riesgos de Calidad Completa

Metas:

1. **Elaborar Lista de Riesgos de Calidad (solo Funcionalidad):** Inicialmente asegure que los proyectos elaboran una lista de los riesgos funcionales de la aplicación. Los riesgos funcionales son problemas potenciales en la

funcionalidad y opciones de la aplicación evidenciados por que al ser ejecutadas los resultados o salidas son incorrectas. La lista de riesgos no puede ser completa, porque entonces tendría los mismos elementos que la especificación de la aplicación, sino que debe tener los riesgos considerados mas importantes. Para adoptar esta mejora considere las siguientes recomendaciones:

- Es importante que la lista de riesgos sea pequeña. Dependiendo del tamaño del proyecto y/o aplicación, esta lista puede tener de 5 a 20 riesgos. Recuerde que la lista de riesgos esta orientada a identificar únicamente los problemas potenciales mas importantes
- Adopte la perspectiva del usuario en el momento de identificar los riesgos. Esto significa que los riesgos mas importantes son los que mas afectarían las tareas de los usuarios o los procesos de negocio mas críticos.
- No es normal que cada módulo, componente o programa tenga asociado igual numero de riesgos. Algunos de los módulos, componentes o programas implican mas riesgos debido a causas como: son los mas utilizados, son nuevos, tienen gran cantidad de cambios/mejoras, son complejos, tienen mas dependencias con otros módulos, requieren precisión en los cálculos y las salidas, son considerados "estratégicos", entre otras.
- Si se ha elaborado una especificación funcional del producto o al menos una lista de las funciones y características que debe tener entonces, puede identificar los riesgos funcionales revisando los requerimientos más importantes y adoptando una actitud negativa. Es decir, para cada requerimiento funcional piense en "lo que podría salir mal". Por ejemplo, si un requerimiento es "Crear un archivo plano (con cierta estructura)", los riesgos asociados pueden ser: "sistema operativo no encuentra la ruta", "nombre incorrecto del archivo plano", "estructura no cumple con la especificación", "sistema operativo se bloquea creando el archivo". De esa lista de problemas potenciales, solo deben elegirse los que sean mas probables de ocurrir y tengan mayor impacto.

2. **Elaborar Lista de Riesgos de Calidad Completa:** Asegure que los proyectos elaboran una lista de riesgos para otros atributos además de la funcionalidad. Entre los atributos pueden estar: seguridad, facilidad de uso, rendimiento, eficiencia, facilidad de instalación, portabilidad, entre otros Para adoptar esta mejora considere las siguientes recomendaciones:

- Investigue el estándar ISO 9126 que define las características de calidad del software. La lista de características (y subcaracterísticas) puede ayudarle a identificar riesgos de calidad diferentes a los de funcionalidad. Defina una plantilla estándar para listas de riesgos de calidad basándose en las características que define el estándar ISO 9126 (*Ver un ejemplo de lista de riesgos de calidad en el paquete de transición*)
- Concéntrese en los riesgos de calidad relevantes e importantes. Si el producto esta orientado a una plataforma específica, por ejemplo, no tiene sentido trabajar en la identificación de riesgos de portabilidad.

- Escuche a los usuarios finales de la aplicación para determinar qué características del producto son criticas para ellos y por lo tanto deben estar en la lista de riesgos

6.4 ETAPA 3: VERIFICACIÓN INTERMEDIA

6.4.1 VERIFICAR UNIDADES/COMPONENTES DE SOFTWARE

La verificación de unidades de software consiste en evaluar la calidad de los componentes del producto con respecto a las especificaciones del componente. Una unidad es una parte del producto que puede ser probada de forma independiente(o con un mínimo esfuerzo). Esto en la practica implica que la "parte" del software debe tener pocas dependencias (bajo acoplamiento) para que pueda ser probado de forma independiente.

La definición concreta de "una unidad " en un producto de software depende de aspectos como:

- Lenguaje utilizado. Ej: PL/SQL, BASIC, Java, C#
- Herramienta y tecnología de desarrollo. Ej: Oracle Forms/Reports, MS Visual BASIC, JDK.
- Arquitectura del producto. Ej: programas y módulos, clases y paquetes

En un producto construido con Oracle Forms/Reports, por ejemplo, las mínimas unidades son las formas y reportes, pero también de acuerdo al tamaño del producto, varias formas y reportes pueden estar agrupadas bajo un menú principal común lo que puede denominarse un módulo. En este caso las unidades son las formas y reportes individuales así como los "módulos". En un producto construido con el lenguaje Java, las unidades son las clases individuales y los paquetes (o también los componentes J2EE que son archivos EAR)

Las pruebas de unidad son también conocidas como "pruebas del desarrollador". Reciben este nombre, porque deben ser realizadas por el desarrollador de la unidad o componente antes de la integración. El beneficio es que al momento de la integración las "partes" individuales se encuentren probadas y corregidas con lo que solo deberían aparecer defectos de integración de los componentes o en la implementación de requerimientos que involucran varios módulos.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Realizar Pruebas de Unidad
2. Realizar Pruebas de Unidad con Técnicas de Caja Blanca

Metas:

1. **Realizar Pruebas de Unidad:** Inicialmente asegure que los desarrolladores realizan pruebas de unidad de los programas, clases o módulos que les son asignados. Considere las siguientes recomendaciones:

- Establezca una política de responsabilidad individual sobre la calidad de las unidades desarrolladas. Cada desarrollador debe garantizar la calidad del programa, clase, o módulo que le es asignado. Convenza a los desarrolladores de que la calidad "se construye desde abajo".
- Entrene a los desarrolladores para que realicen pruebas con múltiples entradas de tal forma que sea necesario ejercitar toda la lógica (las diferentes condiciones en instrucciones "IF", "WHILE", "UNTIL") del componente
- Entrene a los desarrolladores para que también realicen pruebas con datos inválidos
- Entrene a los desarrolladores para crear componentes de prueba, es decir, componentes que prueban otros componentes. Instrúyalos para que creen rutinas que ejecuten el componente con unos datos y esperan una salidas predeterminados.
- Utilice herramientas o librerías para automatizar pruebas de unidad como Junit, VBUnit, Pl/Sql Unit.(Ver lista de herramientas de pruebas de unidad en paquete de transición)
- Utilice un formato de casos de prueba de unidad para definir entradas, procedimiento de prueba y resultados esperados *(Ver ejemplo formato de casos de prueba de unidad en el paquete de transición)*

2. **Realizar pruebas de unidad con Técnicas de Caja Blanca:** Cuando logre institucionalizar las pruebas de unidad, asegure que los desarrolladores conocen y utilizan técnicas de caja blanca. Considere las siguientes recomendaciones:

- Investigue sobre técnicas de prueba de caja blanca *(Ver guía sobre pruebas de unidad en el paquete de transición)*
- Defina una política de cubrimiento de rama mínimo de las pruebas de unidad. El cubrimiento de rama (en ingles "branch coverage") es cuando todas las ramas del código son probadas al menos una vez.
- Automatice la medición del cubrimiento de pruebas de unidad (*Ver lista de herramientas de medición de cubrimiento en el paquete de transición*)

6.4.2 INSTITUCIONALIZAR PROCESO DEFINIDO

Institucionalizar un proceso consiste en lograr que un proceso se convierta en la forma rutinaria de trabajar de una organización. La institucionalización garantiza que las prácticas de un proceso se mantienen una vez son adoptadas y que son utilizadas incluso en momentos de "crisis" y "urgencia".

La institucionalización de un proceso es difícil y siempre toma mas tiempo del esperado.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Documentar formalmente el proceso
2. Proporcionar recursos
3. Capacitar a las personas que realizan el proceso

Metas:

1. **Documentar formalmente el proceso:** Inicialmente asegure que el proceso de pruebas este documentado. El proceso puede documentarse de manera independiente o como parte la documentación de un proceso (o metodología) de desarrollo de software. Considere las siguientes recomendaciones:

- La documentación ayuda a comunicar como se debe realizar el proceso. Sin embargo, no implica que no se deba capacitar y acompañar a las personas para que aprendan a realizar las prácticas del proceso. La documentación es una "memoria" escrita del proceso que ayuda a compartir el conocimiento.
- La documentación debe ser concisa. Si esta es muy detallada, requerirá muchos cambios. Entre mas corta sea la documentación, es mejor.
- Considere utilizar un cronograma estándar como forma de documentar el proceso. Esto hace mas probable que el proceso sea comunicado y utilizado, ya que se encuentra definido en un documento vivo que se esta trabajando todo el tiempo.

2. **Proporcionar Recursos:** Asegure que existen los recursos necesarios para el proceso. Los recursos incluyen equipo de oficina, software, elementos de papelería, bases de datos, espacio en disco, herramientas de software, entre otros. Considere las siguientes recomendaciones:

- Defina una lista de recursos necesarios para realizar el proceso analizando las tareas que lo componen
- Periódicamente revise la lista de recursos realizando reuniones con las personas que ejecutan el proceso. Intente llegar a un consenso y priorize los recursos de acuerdo a su presupuesto

- Considere herramientas de pruebas de software libre. Estas herramientas están disponibles en Internet. En algunos casos puede equiparse a herramientas comerciales

3. **Capacitar a las personas que realizan el proceso:** Asegure que todas las personas que realizan el proceso son capacitadas periódicamente. La capacitación es el elemento mas importante en lograr mejoras permanentes en el proceso. Tenga en cuenta las siguientes recomendaciones:

- La capacitación no es únicamente a través de clases formales en un salón (con una presentación, por ejemplo). Combine otras formas de comunicación, realizando acompañamiento directo a las personas, por ejemplo.
- La capacitación debe ser lo mas práctica posible. Considere mostrar el mayor número posible de ejemplos.
- Seguramente requerirá que las personas sean capacitadas varias veces. Es difícil que con una sola capacitación las personas logren dominar el proceso.

6.4.3 ELABORACIÓN PLAN DE RIESGOS DE CALIDAD

Elaborar un plan de riesgos de calidad que consiste en identificar tareas concretas para prevenir o mitigar la ocurrencia de un riesgo de calidad. Un plan de riesgos de calidad en la practica es una lista que relaciona tareas a los riesgos de calidad identificados. Las tareas finalmente deben hacer parte del plan del proyecto por lo que puede ser mas practico “mover” las tareas luego de identificadas al cronograma del proyecto. Es necesario evitar duplicación de información.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Elaborar Plan de Riesgos de Calidad Funcionales
2. Elaborar Plan de Riesgos de Calidad

Metas:

1. **Elaborar Plan de Riesgos de Calidad Funcionales:** Inicialmente asegure que al identificar riesgos de calidad, se definan tareas asociadas a cada riesgo de calidad de funcionalidad. Este plan de riesgos de calidad se convierte en el plan de pruebas funcionales del producto. Considere las siguientes recomendaciones:

- Defina las tareas para mitigar cada riesgo como pruebas funcionales
- Trate que las tareas definidas sean muy concretas de tal forma que sea posible estimar razonablemente el tiempo
- Considere este plan de riesgos de calidad como un documento temporal. Copie las tareas al cronograma del proyecto para que tengan fechas de inicio y fin definidas. Evite la duplicación de documentos.

2. **Elaborar Plan de Riesgos de Calidad:** Cuando logre institucionalizar la elaboración de planes de riesgos de calidad de funcionalidad, progresivamente defina tareas para otros riesgos de calidad del producto. Considere las siguientes recomendaciones:

- Concéntrese en los riesgos de calidad relevantes e importantes. No todos los riesgos requieren tanta atención. Defina un mayor número de tareas sobre los riesgos importantes.
- Logre consenso entre el equipo del proyecto sobre qué tareas pueden ayudar a prevenir o mitigar los diferentes riesgos
- Defina una plantilla estándar para planes de riesgos de calidad basándose en las características que define el estándar ISO 9126 (*Ver un ejemplo de plan de riesgos de calidad en el paquete de transición*)

6.5 ETAPA 4: VERIFICACIÓN Y VALIDACIÓN TEMPRANA

6.5.1 REVISIÓN DE PRODUCTOS DE TRABAJO

La revisión de productos de trabajo consiste en la evaluación estática de artefactos creados durante el proceso de desarrollo de software. Entre los artefactos o productos de trabajo pueden estar: especificaciones de requerimientos, modelos de análisis, modelos de diseño, especificaciones de interfaces, casos de prueba y el mas importante, el código fuente.

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Revisar especificaciones y modelos
2. Revisar interfaces de usuario
3. Revisar código fuente

Metas:

1. **Revisar especificaciones y modelos:** Inicialmente asegure que las especificaciones de requerimientos, y los modelos de análisis y diseño son revisados. Considere las siguientes recomendaciones:

- Determine criterios de calidad de las especificaciones a revisar. Los criterios mas importantes son: No-ambigüedad (del texto) y que el requerimiento sea verificable. Otros criterios podrían ser que la especificación sea consistente, esto es, que dos requerimientos no se contradigan, y que sea completo, esto es, que tenga en cuenta todas las necesidades de los usuarios y otros afectados por el proyecto. Documente los criterios de calidad como una lista de verificación (*Ver un ejemplo de lista de verificación de requerimientos en el paquete de transición*)
- Utilice la funcionalidad de anotaciones de comentarios que tienen los procesadores de palabra. Esta es una forma práctica de revisar documentos y realizar anotaciones.
- Seleccione solo los modelos que considere mas importantes. Entre estos podrían estar: Entidad/Relación, Diagrama de Clases(UML), Diagrama de Secuencia(UML), Modelos Dimensionales, Modelos de Procesos de Negocios y Modelos de Flujo de Datos
- En modelos grandes, seleccione solo las partes del modelo mas importantes a revisar.

2. **Revisar interfaces de usuario:** Realice revisiones de las interfaces de usuario para evaluar cumplimiento de criterios de calidad, en especial la facilidad de uso. Considere las siguientes recomendaciones:

- Adopte métodos para evaluar facilidad de uso de las interfaces de usuario. Algunos métodos pueden ser: Cuestionarios a usuarios finales, evaluación de expertos independientes, revisión con lista de verificación (*Ver un ejemplo de Lista de Verificación de Facilidad de Uso en el paquete de transición*)

3. **Revisar código fuente:** Realice revisiones de código fuente para evaluar la calidad de este y detectar posibles errores en la lógica. Considere las siguientes recomendaciones:

- Identifique las unidades de código mas críticas según tenga mayor exposición al riesgo (probabilidad de ocurrencia e impacto) de fallar
- Utilice una lista de verificación.Consulte la lista de verificación de ejemplo de código Java
- Considere utilizar las prácticas de programación en parejas. En esta práctica dos programadores trabajan con un mismo computador sobre el mismo

código. Mientras uno de los dos escribe otro va revisando en busca de errores en el código tanto de sintaxis como de lógica.

- Automatice la revisión utilizando herramientas de software. (*Vea herramientas de revisión automática de código el paquete de transición*)

6.5.2 ELABORAR PROTOTIPOS DE REQUERIMIENTOS

Elaborar prototipos de requerimientos consiste en realizar una implementación parcial de estos o "simular" esta implementación. El objetivo es validar la interpretación de los requerimientos mostrando el prototipo al cliente o usuarios finales y obteniendo retroalimentación de estos.

Los usuarios en general prefieren los prototipos antes que las descripciones textuales. Para el desarrollador, el prototipo también puede ser una "primera aproximación" al producto final[50].

Para realizar la adopción de esta práctica establezca las siguientes metas:

1. Elaborar prototipos de requerimientos

Metas:

1. **Elaborar prototipos de requerimientos:** Asegure que los desarrolladores elaboran prototipos durante el levantamiento de requerimientos para obtener retroalimentación rápida de los usuarios finales. Considere las siguientes recomendaciones:

- Utilice en lo posible prototipos en papel o en un tablero. Estos prototipos son mas rápidos de elaborar y evitan que el usuario piense que el requerimiento esta implementado
- Si elabora prototipos ejecutables considere utilizar lenguajes de desarrollo rápido o de secuencias de comandos(en ingles "scripting") como Visual Basic, JavaScript, Perl, Python.
- Para programar un prototipo recuerde mantenerlo simple:
    i. No defina código para validar entradas ni manejas excepciones
    ii. Si el requerimiento define salidas, haga que el prototipo produzca salida con valores predefinidos constantes
    iii. Si el requerimiento es un reporte o una consulta coloque una lista de resultados constante

- Considere utilizar prototipos evolutivos, es decir, el prototipo no es desechado sino que es un primer borrador de la implementación del requerimiento
- Incluya tiempo en el cronograma proyecto para el desarrollo de prototipos
- Formalice la validación de prototipos utilizando un cuestionario estándar a ser respondido por los usuarios finales
- Defina un procedimiento de validación de prototipos para uso de todos los proyectos (*Ver un ejemplo de procedimiento de validación de prototipos en el paquete de transición*)

6.6 CONCLUSIÓN

En esta sección se presentan las etapas de mejoramiento que define la estrategia. En cada etapa, a su vez, se describen en detalle cada una de las prácticas con metas y sugerencias de mejoramiento concretas. En el texto de cada una de las prácticas se referencian los materiales del paquete de transición.

7.APLICACIÓN DE LA ESTRATEGIA

6.1 INTRODUCCIÓN

En esta sección se presenta un caso de estudio sobre la aplicación de la estrategia en una empresa de software colombiana[16]. Inicialmente se describe la empresa y sus procesos de negocio principales explicando sus requerimientos de verificación y validación. A continuación, se explica el programa de mejoramiento de la empresa. Por ultimo, se presenta el trabajo realizado y los logros alcanzados hasta el momento[17].

6.2 DESCRIPCIÓN DE LA EMPRESA CASO DE ESTUDIO

La misión de la empresa seleccionada incluye ofrecer, desarrollar e implantar soluciones de software integrales para empresas públicas y privadas en Colombia y en Latinoamérica. Las soluciones integrales que ofrece esta empresa incluyen productos estándar de software empresarial[18] junto con servicios profesionales de capacitación, desarrollo de personalizaciones y consultoría. Gran parte de los productos estándar han sido desarrollados por la propia empresa utilizando tecnologías Oracle, en particular, Oracle Developer Forms y Reports y el lenguaje de programación PL/SQL.

Los productos de la empresa, como es usual en aplicaciones empresariales, tienen una arquitectura modular. Esto facilita no solamente el mantenimiento de los productos sino que le da flexibilidad a la empresa para ofrecer soluciones a los clientes que incluyan solamente los módulos que estos necesitan. También es muy conveniente en la implantación, ya que esta se puede realizar de forma incremental y se evita el riesgoso enfoque de implantación “Big Bang”. Esto sin embargo, requiere que la empresa tenga destrezas especiales en pruebas de sistema para lograr identificar cómo ciertos casos de uso y procesos de negocio se ejecutan a través de varios módulos.

El tamaño de la empresa es mediano. La empresa emplea aproximadamente 120 empleados de los cuales cerca del 90% son ingenieros de sistemas. Un gran porcentaje de ingenieros son contratados de forma temporal, de acuerdo a las necesidades de los proyectos. De esta forma la empresa puede controlar sus costos de nomina de acuerdo a la demanda del mercado. Los proyectos que ejecuta la empresa en promedio no superan el tamaño de 10 hombres-mes en esfuerzo.

La empresa tiene cuatro procesos clave relacionados con su misión: Innovación de Productos y Servicios, Gestión de Venta, Desarrollo e Implantación de Soluciones y Servicio Post-Venta. En cada uno de estos procesos, la empresa existe la posibilidad de realizar Verificación y Validación de productos de trabajo relacionados con Software.

---

[16] Por solicitud de la empresa, se omite el nombre o cualquier información que pueda identificarla

[17] Es importante aclarar que la aplicación de la estrategia se esta llevando a cabo actualmente, es decir no se han cubierto totalmente las 4 etapas.

[18] Este tipo de software es conocido en el mercado como “paquetes”

El proceso de Innovación de Productos y Servicios incluye actividades de planeación de nuevos productos y servicios. En este proceso es donde se definen los requerimientos de nuevos productos o de nuevas versiones de los productos actuales. También en este proceso se aprueba el inicio de proyectos para el desarrollo de productos y la liberación de nuevas versiones. Los productos de trabajo críticos a ser verificados y/o validados en este proceso son las especificaciones de requerimientos funcionales bien sea de nuevos productos o de nuevas versiones de productos actuales.

El proceso de Gestión de Venta incluye actividades de identificación de clientes potenciales, identificación preliminar de necesidades, expectativas y restricciones de los clientes, diseño y presentación de ofertas para satisfacer y cubrir las necesidades, expectativas y restricciones previamente identificadas, y finalmente la negociación y cierre de la venta. Este proceso tiene la característica que puede terminar en cualquier momento si el cliente no esta interesado. También es usual que los clientes se tomen varios meses o años antes de tomar una decisión. Los productos de trabajo críticos a ser verificados y/o validados en este proceso son las especificaciones de necesidades y las ofertas de soluciones.

El proceso de Desarrollo e Implantación de Soluciones incluye actividades de especificación de requerimientos, elaboración de modelos de diseño, programación, pruebas unitarias, pruebas de sistema, pruebas de aceptación de usuario, instalación y configuración de software, y capacitación. Las actividades de este proceso se pueden realizar para lograr diferentes objetivos como desarrollar productos nuevos (o nuevas versiones de productos actuales), implantación sola, implantación con desarrollo de personalizaciones o desarrollo totalmente a la medida. Al estar definido como un proceso estándar que puede ser utilizado para diversos objetivos requiere de una apropiada interpretación según el caso. En este proceso existe un número mayor de potenciales productos de trabajo que podrían ser verificados y/o validados incluyendo especificaciones de requerimientos, modelos de diseño, código fuente, código ejecutable, manuales, material de capacitación y en general cualquier entregable considerado crítico en el servicio al cliente.

El proceso de Servicio Post-Venta incluye actividades como soporte y mantenimiento de las soluciones que ya se encuentran en producción en los de clientes. En realidad, las actividades de este proceso son las mismas que el proceso de Desarrollo e Implantación de Soluciones con la gran diferencia que se realizan sobre un sistema en producción. Los productos de trabajo que podrían ser verificados y/o validados en este proceso son los mismos que en el proceso de Desarrollo e Implantación de Soluciones.

6.3 DESCRIPCIÓN PROGRAMA DE MEJORAMIENTO

Como parte de sus iniciativas estratégicas, la empresa decidió iniciar un programa de mejoramiento de procesos basado en el modelo CMMI. La empresa busca mejorar la eficacia y eficiencia de sus procesos clave de negocio[19]: Innovación de Productos y Servicios, Gestión de Venta, Desarrollo e Implantación de Soluciones y Servicio Pos-

[19] La empresa eventualmente utiliza áreas de proceso del modelo para otros procesos cuando puedan ser aplicables. Algunas de áreas de proceso de CMMI con alcance organizacional podrían ser: Medición y Análisis, Enfoque de Procesos Organizacionales, Definición de Procesos Organizacionales, Entrenamiento y Gestión de Riesgos.

Venta. Las mejoras en eficacia apuntan a mejorar la calidad de los productos y servicios y aumentar la satisfacción del cliente. Las mejoras en eficiencia apuntan a aumentar la productividad y a reducir las horas no facturables. Las horas no facturables se deben a re-trabajos en los procesos de Desarrollo e Implantación de Soluciones, y de Servicio de Post-Venta. La causa de los re-trabajos es defectos en el software razón por la cual las horas dedicadas a soluciones los defectos no se pueden facturar a clientes. Estos costos de horas no facturables corresponden a costos de fallas externas en la ecuación de costos de calidad.

La empresa ejecuta el programa de mejoramiento a través de ciclos que siguen una metodología basada en el modelo IDEAL. Cada ciclo de mejoramiento corresponde a cada una de las áreas de proceso del modelo CMMI.

Desde el inicio del programa de mejoramiento la empresa ha definido e institucionalizado nuevas prácticas en las áreas de proceso de nivel 2 del modelo CMMI: Gestión de Requerimientos, Planeación de Proyectos, Monitoreo y Control de Proyectos, Aseguramiento de Calidad y Gestión de Configuración.

La organización del Programa de Mejoramiento incluye un Grupo de Procesos(llamado EPG por las siglas en ingles de “Engineering Process Group”), el cual es el encargado de coordinar las actividades del programa de mejoramiento, y Equipos de Trabajo por cada una de las áreas de proceso(llamados PATS por las siglas en ingles de “Process Action Team”), quienes se encargan de apoyar la definición de nuevas prácticas y nuevos estándares de productos de trabajo(Ej: formas, plantillas,etc). El grupo de Procesos incluye a un ingeniero de sistemas y un ingeniero industrial dedicados de tiempo completo al programa de mejoramiento. Los Equipos de Trabajo son conformados por personal de línea(consultores y directores de proyecto), quienes trabajan tiempo parcial en los ciclos de mejoramiento.

Como en cualquier programa de mejoramiento basado en un modelo, la interpretación apropiada con respecto a las características particulares de la organización no es una tarea fácil. En esta empresa, la labor de interpretación principalmente esta a cargo del Grupo de Procesos quienes tratan de “aterrizar” las prácticas del modelo a las realidades de la organización y encontrar soluciones concretas que puedan ser adoptados.

6.4 CICLO DE MEJORAMIENTO DE V&V Y APLICACIÓN ESTRATEGIA

La empresa donde se esta realizando la aplicación de la estrategia, sigue una metodología basada en el modelo IDEAL para realizar los ciclos de mejoramiento para cada una de las áreas de proceso del modelo CMMI.

La metodología tiene los siguientes pasos:

1. **Elaborar Plan y Cuestionario de Diagnóstico:** En esta actividad se elabora un cuestionario de diagnóstico sobre un área de proceso y se define a quiénes se le entregará el cuestionario.
2. **Realizar Diagnóstico:** En esta actividad se lleva a cabo el diagnóstico bien sea como entrevistas personales a los consultores y directores de proyecto o a través de encuestas

3. **Elaboración Resultados Diagnóstico:** En esta actividad se elabora un documento con los resultados del diagnóstico. El documento describe las fortalezas y problemas encontrados y define recomendaciones para cada uno de los problemas
4. **Reunión Inicial PAT:** Esta es la reunión inicial del PAT del área de proceso respectiva. El objetivo de esta reunión es presentar el área de proceso y los resultados del diagnóstico.
5. **Reunión Lluvia de Ideas:** En esta reunión se priorizan los problemas encontrados y se identifican posibles soluciones. También se planean pilotos de las soluciones
6. **Reunión de Pilotos y Avances:** En esta reunión se revisan los resultados de los pilotos y los avances logrados hasta el momento. Esta reunión se repite por lo general 3 o mas veces, hasta que se decide proceder a divulgar los nuevos cambios en toda la organización.

Para aplicar la estrategia se realizaron algunos cambios en la metodología:

1. El ciclo de mejoramiento fue definido para Verificación y Validación de forma integrada aunque en el modelo CMMI son dos áreas de proceso diferentes e independientes.
2. El cuestionario de diagnóstico se realizó basándose en las prácticas definidas por la estrategia. Puede ver el cuestionario en el anexo 1.
3. Las recomendaciones a los problemas del diagnóstico se elaboraron siguiendo las recomendaciones de la estrategia.

Los principales problemas encontrados con el diagnóstico fueron:

1. No existe un ambiente de prueba para realizar pruebas
2. Clientes algunas veces no realizan pruebas de aceptación
3. Existe poca planeación y seguimiento de las pruebas
4. Estimación inadecuada del tiempo de pruebas
5. Pruebas de sistema no son basadas en los requerimientos
6. Consultores Técnicos no realizan pruebas técnicas

Luego de los resultados del diagnóstico, se inició la aplicación de la estrategia en su primera etapa llamada “validación básica”. En esta etapa según la estrategia se deben adoptar las siguientes prácticas :

- Preparar ambiente de prueba (práctica de verificación y/o validación)
- Realizar pruebas de aceptación (práctica de validación)
- Planificación del proceso de pruebas (práctica de gestión del proceso)

En la aplicación de esta primera etapa de la estrategia, hasta el momento la empresa ha identificado soluciones como:

1. **Configurar ambientes de pruebas independientes para cada uno de los productos:** Se ha trabajado en crear ambientes de prueba con usuarios de base de datos independientes y copias diferentes de los programas ejecutables. El acceso al ambiente de prueba se esta configurando para ser restringido únicamente al personal de pruebas. Esta solución permite adoptar la práctica "preparar ambiente de prueba".
2. **Preparar Datos de Prueba:** Se ha trabajado en la creación de matrices de casos de prueba para dos de los productos de la empresa. En estas matrices se han seleccionado valores de datos de prueba. Esta solución permite adoptar la práctica "preparar ambiente de prueba".
3. **Diseño Formato Pruebas de Aceptación:** Se diseñó un formato de pruebas de aceptación. El formato incluye información del usuario que realiza las pruebas, criterios de aceptación que se deben acordar con los usuarios y fecha de compromiso para realizar las pruebas. El formato también incluye información para firmas. Esta solución permite adoptar la práctica "realizar pruebas de aceptación".
4. **Pruebas basadas en Requerimientos:** Se rediseño el formato de especificación de requerimientos para establecer la trazabilidad de las pruebas con los requerimientos. Esta solución permite adoptar la práctica "planificación del proceso de pruebas".
5. **Estimación de Pruebas:** Se incluyo explícitamente las actividades de prueba en el formato de plan de trabajo relacionado a cada requerimiento. Con este cambio, se pretende que al realizar las estimaciones de cada requerimiento si se estime el tiempo de pruebas. Esta solución permite adoptar la práctica "preparar ambiente de prueba".

Paralelo a estas soluciones, la empresa decidió crear una nueva área independiente dedicada al control de calidad. Con esta área se busca que el personal de pruebas sea independiente de los desarrolladores. Para esta área se ha venido trabajando en definir políticas, metodologías, responsabilidades, herramientas, productos de trabajo a revisar, listas de verificación y formatos entre otros elementos. La aplicación de la estrategia se esta dirigiendo actualmente a apoyar este objetivo.

6.5 CONCLUSIÓN

En esta sección se presenta un caso de estudio de aplicación de la estrategia de mejoramiento en una empresa colombiana de software. En principio, se describe la empresa y su programa de mejoramiento. Finalmente se describe el trabajo realizado hasta el momento y algunos de los logros alcanzados.

8.CONCLUSIONES

8.1 RESUMEN

En este documento se ha presentado una estrategia para el mejoramiento del proceso de Verificación y Validación.

En la primera parte del documento se analiza el contexto de los problemas de calidad del software así como de posibles soluciones que se proponen a nivel mundial y en la industria del software colombiana.

A continuación, se explica la justificación de trabajar en el mejoramiento del proceso de Verificación y Validación a partir de un análisis económico de los costos de calidad y de la ingeniería de software, además de las necesidades de la industria de software colombiana de obtener certificaciones de calidad para acceder a mercados internacionales.

Enseguida, se realiza una síntesis del estado del arte y de la teoría en 4 grandes temas: Calidad de Software, el Proceso de Verificación y Validación, el Mejoramiento del Proceso de Desarrollo de Software, y de los Modelos de Mejoramiento del Proceso de Verificación y Validación.

En las siguientes dos secciones, se presenta la estrategia propuesta la cual define un conjunto de prácticas, cuya adopción se realiza de forma gradual a través de un programa de 4 etapas de mejoramiento, y la utilización de un paquete de transición. El programa incluye prácticas y sugerencias concretas para mejorar la forma como se realizan las actividades de verificación y validación así como la gestión del proceso y la gestión de riesgos de calidad. Al incluir prácticas de gestión de procesos y gestión de riesgos de calidad, se busca apoyar la optimización de costos de calidad. Con el paquete de transición se proporcionan instrumentos precisos para que una organización de software pueda realizar las prácticas definidas.

Con base en estos requerimientos y restricciones la estrategia define un conjunto de 11 prácticas:

1. Preparar Ambiente de Pruebas
2. Realizar Pruebas de Aceptación
3. Planificación del Proceso
4. Verificación de Software
5. Gestión del Proceso
6. Identificación de Riesgos de Calidad
7. Verificación Unidades/Componentes de Software
8. Institucionalizar Proceso Definido
9. Elaborar Plan de Riesgos de Calidad
10. Revisiones de Productos de Trabajo
11. Elaborar Prototipos de Requerimientos

Por ultimo, luego de la presentación de la estrategia, se describe la aplicación de la misma en una empresa de software colombiana que sigue un programa de mejoramiento basado en CMMI.

8.2 APORTES DEL TRABAJO

Los principales aportes de este trabajo son:

- **Síntesis del estado del arte:** En la descripción del marco teórico se realiza una síntesis del estado del arte en 4 grandes temas. Esta síntesis facilita a otros investigadores la comprensión individual de los mismos y las relaciones existentes entre estos. Los 4 temas son: Calidad de Software, el Proceso de Verificación y Validación, el Mejoramiento del Proceso de Desarrollo de Software, y de los Modelos de Mejoramiento del Proceso de Verificación y Validación.
- **Identificación de requerimientos para una estrategia de mejoramiento del proceso de V & V:** El conjunto de requerimientos y restricciones, identificado a partir de la revisión del marco teórico y el contexto colombiano, permite a académicos y empresas colombianas comprender los aspectos a tener en cuenta al elaborar soluciones para mejorar el proceso de Verificación y Validación. Entre los requerimientos y restricciones identificados están:

    - Definir Prácticas a Adoptar
    - Prácticas para Ciclo de Vida Completo
    - Adopción incremental de practicas
    - Apoyar optimización de costos totales de calidad
    - Cumplir con un estándar internacional de proceso
    - Proporcionar instrumentos precisos de aplicación
    - Adaptación al contexto

- **Elaboración de una estrategia de mejoramiento:** La estrategia de mejoramiento propuesta define un conjunto de prácticas y un programa de etapas incrementales para adoptar mejoras en el proceso de Verificación y Validación. La estrategia sirve a las empresas que requieran definir como realizar un ciclo de mejoramiento del proceso de V & V. Las empresas pueden utilizar la estrategia "tal cual", para acelerar la institucionalización de nuevas prácticas, o pueden utilizarla como ejemplo para definir su propia estrategia. La estrategia también proporciona un ejemplo de "solución" para el conjunto de requerimientos y restricciones identificados. Por ejemplo, la inclusión en la estrategia de prácticas de gestión de procesos y gestión de riesgos de calidad apoya la optimización de costos totales de calidad, un problema que se presenta al institucionalizar nuevas prácticas de Verificación y Validación. También sirve como modelo de implementación de un estándar internacional como CMMI, en dos de sus áreas de proceso[20]. Por ultimo, esta estrategia de mejoramiento permite aplicar dos de las

[20] La carencia de un modelo de implementación es una de las críticas que se realiza sobre CMMI. Ver Marco Teórico.

estrategias identificadas por Kotter para reducir la resistencia la cambio (La estrategia de "educación y comunicación", y la estrategia de "facilitación y soporte") [30].

- **Elaboración de un paquete de transición:** La estrategia viene acompañada de un paquete de transición, con lo que en cierta forma, la estrategia se convierte en una "solución" completa al definir, qué prácticas adoptar y en qué orden, cómo implementar esas prácticas, y qué instrumentos utilizar.
- **Aplicación de la estrategia en una empresa colombiana de software:** En el trabajo se ha presentado la aplicación de la estrategia en una empresa colombiana de software. Se describieron los requerimientos de los procesos de la empresa estudiada en cuanto a Verificación y Validación y se mostró como utilizar la estrategia al realizar un ciclo de mejoramiento basado en el modelo IDEAL.

8.3 TRABAJOS FUTUROS

El trabajo de investigación realizado se puede extender de varias maneras. Algunos de los trabajos futuros pueden ser:

- Desarrollo de otras estrategias a partir de los requerimientos y restricciones identificados. Se podrían plantear estrategias con otras prácticas diferentes a las planteadas, o con iguales prácticas pero con un programa alternativo, pero que igual cumplan los requerimientos aquí identificados.

- "Enriquecimiento" de la estrategia y del paquete de transición. Es posible agregar otras prácticas o agregar metas y sugerencias en las descripciones de las prácticas. El paquete de transición se puede mejorar con materiales adicionales. Ejemplo: Materiales para apoyar prácticas de Verificación y Validación de seguridad y facilidad de mantenimiento.

- Continuación de la aplicación de la estrategia en la empresa estudiada. La aplicación de la estrategia en la empresa estudiada seguramente continuará gracias a los buenos resultados obtenidos hasta el momento. Un trabajo de investigación podría ser evaluar los resultados de la estrategia luego de un período de al menos 2 años de comenzar a ser aplicada en esta empresa.

- Aplicación en otras empresas colombianas de software. Es necesario aplicar la estrategia en otras empresas colombianas de software y eventualmente en empresas de otros países de Latinoamérica. Esto ayudaría a refinar la estrategia para que sea mas aplicable en el contexto de Colombia y de la región.

9. REFERENCIAS

[1] Ambler,Scott, "The Full Life Cycle Object-Oriented Testing (FLOOT) Method" **,** Página web descargada de http://www.ronin-intl.com/publications/floot.html en Febrero de 2003

[2] Ardila, Consuelo, "El Sector de Software en Colombia y su Proyección Internacional", Proyecto de grado para optar al título de Magíster en Administración, Universidad de los Andes,2003

[3] Bach, James, "Risk and Requirements-Based Testing", Columna "Software Realities", Revista "IEEE Computer", 1999 (documento en formato PDF descardo de http://www.satisfice.com en Febrero de 2004

[4] Bach,James, "Process Evolution in a Mad World", documento en formato PDF descargado de http://www.satisfice.com en Febrero de 2004

[5] Bach, James, "The immaturity of CMM", página web descargada de http://www.satisfice.com en Febrero de 2004

[6] Bach,James, "The Challenge of "Good Enough" Software", documento en formato PDF descargado de http://www.satisfice.com

[7] Bach, James, "Exploratory Testing Explained" , documento en formato PDF descargado de http://www.satisfice.com en Febrero de 2003

[8] Beck,Kent, "Una explicación de la programación extrema: Aceptar el cambio", Pearson Educación, Madrid, 2002

[9] Sitio web www.betterproductdesign, "Process Maturity / Capability Maturity", página web descargada de http://www.betterproductdesign.net/maturity.htm en Enero de 2004

[10] Black, Rex, "The Risks to System Quality", página web descargada de http://www.stickyminds.com en Mayo de 2003

[11] Black, Rex, "The Cost of Software Quality", página web descargada de http://www.stickyminds.com en Mayo de 2003

[12] Boehm,Barry, y Basili,Victor, "Software Defect Reduction Top 10 List", documento en formato PDF descargado de http://www.cebase.org/www/researchActivities/defectReduction/ top10/top10defects-computer-2001.pdf

[13] Boehm,Barry, "CMMI and the balance of Discipline and Agility" documento PDF descargado de http://www.dtic.mil/ndia/2002cmmi/boehm.pdf

[14] Grupo Bolívar, “Juntos Hacia el Cambio” , Material de capacitación del Centro de Formación Ejecutiva del Grupo Bolívar

[15] Brooks,Frederick, “No Silver Bullet: Essence and Accidents of Software Engineering”, página web descargada de http://www-inst.eecs.berkeley.edu/~maratb/readings/NoSilverBullet.html en junio de 2003

[16] Bruegge,Bern y Dutoit, Allen, “Ingeniería de Software Orientado a Objetos”, Prentice Hall,2002

[17] Bunnyfoot, “The Business Case For Usability”, página web descargada de http://www.bunnyfoot.com/freestuff/articles/usability/usabilitybusinesscase.html

[18] Burnstein, Ilene, “Developing a Testing Maturity Model: Part I”, página web descargada de http://www.stsc.hill.af.mil/crosstalk/1996/08/developi.asp

[19] CMU/SEI, “Capability Maturity Model Integration (CMMI) Version 1.1. Staged Representation (CMU/SEI-2002-TR-029)”, Agosto de 2002. Documento en formato PDF descargado de http://sei.cmu.edu.co

[20] Endres, Megan, “Zero Defects Strategy in Software Development: The Costs of Being Bug-Free”, Decision Sciences Institute 2002 Annual Meeting Proceedings, documento en formato PDF descargado de http://www.sbaer.uca.edu/Research/2002/dsi/papers/477.pdf en mayo de 2003

[21] Ericson, Thomas, “TIM- A Test Improvement Model”, documento en formato PDF descargado de http://www.lucas.lth.se el 3 de Mayo de 2003

[22] Fowler, Priscila, “Transition Packages for Expediting Technology Adoption: The Prototype Requirements Management Transition Package(CMU/SEI-98-TR-004)”, Septiembre de 1998. Documento en formato PDF descargado de http://sei.cmu.edu.co en octubre de 2003

[23] Goeldner, Rudolf, “A Cost-Benefit Model for Software Testing”, Quality Techniques Newsletter, Julio de 2000, página web descargada de: http://www.soft.com/News/QTN-Online/qtnjul00.html en mayo de 2003

[24] Humphrey, Watts, “Managing the Software Process”, 1989

[25] Instituto Colombiano de Normas Técnicas y Certificación(ICONTEC) “NTC-ISO 9000”, norma equivalente a la norma ISO 9000:2000(traducción certificada), ICONTEC, Diciembre de 2000

[26] Gemba Jennifer y Myers,Chuck, “The IDEAL Model: A practical guide for improvement”, página web descargada de http://www.sei.cmu.edu/ideal/ideal.bridge.html

[27] IEEE, "SWEBOK: Guide to the Software Engineering Body of Knowledge" Trial Version May 2001, documento PDF descargado de http://www.swebok.org en Noviembre de 2003
[28] Kaner, Cem "Quality Cost Analysis: Benefits and Risks", página web descargada de http://www.kaner.com/qualcost.htm en Mayo de 2003

[29] Koomen, Tim, y Pol, Martin, "Test Process Improvement", Addison Wesley, 1999

[30] Kotter, John, y Schlesinger,Leonard, "Choosing strategies for change" Harvard Business Review 57 no. 2 (Marzo-Abril 1979): pp. 106-114

[31] Mann, Charles, "Why Software is so Bad", Technology Review, documento en formato PDF descargado de http://www.technologyreview.com el 4 de Mayo de 2003(requiere compra del articulo)

[32] Mariño, Hernando, "Gerencia de Procesos", Editorial AlfaOmega, 2001

[33] Moore, Jim, "ISO 12207 and Related Software Life-Cycle Standards", documento PDF descargado de http://www.acm.org/tsc/lifecycle.html en Enero de 2004

[34] Muñoz-Seca, Beatriz, y Riverola,Joseph, "Del buen pensar y mejor hacer", Editado por Mcgraw Hill y IESE-Business School (Universidad de Navarra) , 2003
[35] Natwick,Gary, y Cocci,Jim "Understanding the CMMI® Validation Process Area", Presentación Power Point descargada de la web en febrero de 2004

[36] Nickols, Fred, "Change Management 101 : A Primer" página web descargada de http://home.att.net/~nickols/change.htm en febrero de 2004

[37] Olson, Tim "Successful Verification and Validation Based on the CMMI Model", Presentación Power Point descargada de http://www.qic-inc.com en febrero de 2004
[38] Paulk,Mark, "CMM v1.1 Overview", documento PDF descargado de www.sweforum.net/process/cmm_over.pdf

[39] Paulk,Mark, "The Capability Maturity Model for Software", Documento PDF descargado de http://:www.sei.cmu.edu.co

[40] Perry, William, "Effective Methods for Software Testing", Wiley, 2001

[41] Peters,Tom, "Thriving on Caos: Handbook for a Management Revolution", HarperPerennial,1991

[42] Pfleeger, Shari, "Ingeniería del Software: Teoría y Práctica", Editorial Prentice Hall, Enero de 2002

[43] Rico,David: "Return on Investment on Software Process Improvement: Metrics and Models for Software Engineering", documento PDF descargado http//www.davidfrico.com en Mayo de 2003

[44] Rico, David, “CMM Strategic/Tactical Plan”, documento PDF descargado de http://www.davidfrico.com en Mayo de 2003

[45] Spence, Ian, “Principles of Process Improvement”, Página web descargada de http://www.therationaledge.com/content/jan_02/f_orgProcessImprovement_is.html en octubre de 2003

[46] Organización Internacional de Estándares(ISO), “Software Process Improvement and Capability Determination”, documentos formato .DOC descargados de http://www.sqi.gu.edu.au/spice/

[47] Turner,Richard y Jarzombek,Joe, “CMMI and Agile Processes: Can’t we all just get along?” documento PDF descargado de http://www.dtic.mil/ndia/2002cmmi/turner3b2.pdf

[48] Westphal, James, “What causes resistance to change”, documento en formato .DOC descargado de http://www.mccombs.utexas.edu/faculty/James.Westphal/

[49] Widmer,Lori, “When software fails: the risks that isn’t covered-Technology Solutions”, página web descargada de http://www.findarticles.com/p/articles/mi_m0BJK/is_3_13/ai_83993287

[50] Wiegers,Karl, “Software Requirements”, Editorial Microsoft Press, 1999

ANEXO 1

CUESTIONARIO DE DIAGNÓSTICO PROCESO DE VERIFICACIÓN Y VALIDACIÓN

Nombre:
Fecha Entrevista:

1.Existe un ambiente de prueba para la versión estándar?

Si No

2. Qué tan frecuente es que los clientes tengan un ambiente de prueba?

Casi siempre A veces Casi nunca

3.Qué tan frecuente es que el cliente realice pruebas de aceptación de las personalizaciones?

Casi siempre A veces Casi nunca

4. Las personalizaciones son probadas por el consultor funcional?

Casi siempre A veces Casi nunca

5.Las modificaciones que se realizan sobre el producto estándar son probadas por el consultor funcional?

Casi siempre A veces Casi nunca

6. Antes de las pruebas, se definen previamente casos/escenarios de prueba?

Casi siempre A veces Casi nunca

7. Son registrados los errores encontrados en las pruebas?

Casi siempre A veces Casi nunca

8.El cronograma del proyecto incluye tiempo para pruebas?

Casi siempre A veces Casi nunca

9. Se realiza un cronograma específico para las pruebas (detallando las actividades de prueba del cronograma general del proyecto?

Casi siempre A veces Casi nunca

10. Si se realiza un cronograma de pruebas, cómo se organiza?

Requerimiento Opción/Funcionalidad Caso de Prueba Otro

11. Cuándo se quieren realizar pruebas, se identifican las partes mas críticas a ser probadas y a estas se les dedica mas tiempo?

Casi siempre A veces Casi nunca

12. Cuándo se quieren realizar pruebas, se prueba todo por igual?

Casi siempre A veces Casi nunca

13. Los Directores de Proyecto realizan seguimiento a las pruebas frente a lo planeado?

Casi siempre A veces Casi nunca

14. Las formas, reportes y el código PL/SQL es revisado por otros consultores o por el Director de Proyecto para obtener una segunda opinión, encontrar errores?

Casi siempre A veces Casi nunca

15. Los documentos y manuales son revisados por otros consultores o por el Director de Proyecto para obtener una segunda opinión, encontrar errores?

Casi siempre A veces Casi nunca

16. Describa la forma cómo se realizan las pruebas de las personalizaciones?

17. Describa la forma cómo se realizan las pruebas de las modificaciones de la versión estándar?