UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
ACATLAN

EL MODELO DE EFICIENCIA Y MADUREZ (CMM) COMO HERRAMIENTA PARA LA MEJORA DEL PROCESO DE DESARROLLO DE SOFTWARE

T E S I N A

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACION

P R E S E N T A:

GOMEZ BECERRIL LORENA ALEJANDRA

ASESOR:
M. en C. Sara Camacho Cansino

Abril 2004

A mi familia

A la memoria de mis Abuelos y de doña Boni

A todas aquellas personas que han compartido sus experiencias y conocimientos conmigo

Gracias

INTRODUCCIÓN

En mi camino profesional dentro del campo de desarrollo de sistemas, (el camino de la mayoría de los egresados de la licenciatura de MATEMÁTICAS APLICADAS Y COMPUTACIÓN), he tenido la oportunidad de observar como magnificas ideas para la mejora de procesos han tenido que ser dejadas a un lado, debido a lo costoso e ineficiente del proceso de desarrollo del producto. La falta de una administración adecuada ha ocasionado el desperdicio de recursos y su asignación hacia esfuerzos menos redituables e incluso inútiles. Lo anterior repercute en una mala imagen sobre cualquier desarrollo de sistemas, asociándolo a problemas tales como costos elevados, tiempos de entrega atrasados y funcionalidad no solicitada o defectuosa, lo que ocasiona conflictos al usuario final al detectar, suponer o comprobar que el nuevo sistema no le reducirá trabajo, sino que al contrario, ahora tendrá que resolver nuevos problemas. Esto generalmente no se debe a la incapacidad técnica del personal, sino se origina de un mal planteamiento de objetivos, planes de trabajo demasiado ambiciosos y falta de información sobre los posibles obstáculos que puedan presentarse en el proyecto. La mayoría de las causas de estos problemas pueden resumirse a la falta de conocimiento sobre administración de proyectos, pues recopilar información que pueda ser usada posteriormente, es la base para plantear tiempos y funcionalidades realistas y documentar las restricciones que se presentaron en un proyecto puede ayudar a replantear los objetivos y alcances de proyectos similares, obteniendo con ello un producto eficiente y un planteamiento realista del resultado final que se obtendrá basándose en información previa.

Uno de los retos principales para cualquier persona involucrada en la industria de software es mejorar el proceso de desarrollo de los proyectos, para ello existen diversos modelos para ayudar a solventar dicho problema, uno de los más populares -y mejor documentados-, actualmente es el SEI-CMM (SOFTWARE ENGINEERING INSTITUTE - CAPABILITY AND MATURITY MODEL), tema principal del presente trabajo, el cual está dirigido a todas aquellas personas que buscan una guía respecto a las actividades a seguir para un mayor control de proyectos de software, o aquellas personas interesadas en ingeniería de software. Para el aprovechamiento de la lectura del presente trabajo se requieren conocimientos básicos de análisis y diseño de sistemas.

Debido a mi experiencia profesional considero importante analizar las principales causas de los problemas asociados con el desarrollo de software y hacer una revisión general del modelo SEI-CMM para brindar puntos de vista sobre las ventajas y desventajas que conlleva este y como su uso y conocimiento pueden apoyar a minimizar los conflictos derivados de la actividad del desarrollo de software para ofrecer productos de mayor calidad.

Es importante conocer el modelo, el cual no es tan conocido en nuestro país como los esquemas ISO, y es una herramienta mas para mejorar el desarrollo de software. Debido a que su bibliografía es escasa, costosa y en Internet la información referente a él es demasiado diversa, considero valioso concentrar los conceptos generales del modelo en un trabajo que pueda servir de referencia para quien desee una introducción al modelo.

El trabajo consta de 4 capítulos:

La administración de recursos en el desarrollo de software: Donde se describen los principales factores que afectan el proceso de desarrollo de software estableciendo la necesidad de administrarlos de manera correcta para tener un producto exitoso, confiable y de bajo costo.

Definición de los componentes del modelo de eficiencia y madurez (CMM): En este capítulo se definirán los principales conceptos en los que se apoya el modelo de eficiencia y madurez, sus características, componentes y usos generales.

Descripción de los componentes de cada nivel del modelo de eficiencia y madurez (CMM): Se detallará la estructura de cada uno de los niveles que componen al CMM, describiendo sus procesos claves, actividades comunes y salidas, así como también las relaciones entre estos para establecer su importancia en el logro del resultado.

Ventajas, desventajas y recomendaciones sobre el modelo de eficiencia y madurez (CMM): Se brindaran puntos de vista sobre las ventajas y desventajas del modelo, con el fin de obtener el mejor provecho que se pueda del mismo.

El alcance del presente trabajo es solo teórico, pues tiene como objetivo presentar las características y componentes principales del modelo a través de la síntesis de información recopilada a través de diversos artículos publicados en Internet y varios libros especializados. Espero que su lectura sea de utilidad y logre crear conciencia sobre la importancia de administrar los recursos y factores que afectan un proyecto.

I. LA ADMINISTRACIÓN DE RECURSOS EN EL DESARROLLO DE SOFTWARE

En estos tiempos, 'Calidad' es la palabra. Los clientes esperan productos de calidad que satisfagan sus necesidades, resuelvan sus problemas y les generen ganancias. Sin embargo, la calidad no ha sido el punto fuerte de la industria de software.

La mayoría de los sistemas no trabajan como se esperaba. Existen sistemas que cubren las necesidades de los usuarios con un bajo nivel de calidad, y que después de varios años, aún tienen errores, pero se continúan utilizando porque es la herramienta más valiosa para 'sacar la chamba'.

Se estima que el rango de cancelaciones o fallas de los sistemas de software es superior al 20%, el 60% de los sistemas terminados, han rebasado los tiempos de entrega y excedido sus costos y el 60% de los sistemas presentan problemas y tienen un nivel bajo de confiabilidad en su primer año de desarrollo.

De acuerdo a un estudio realizado por T. DeMarco, el promedio de los productos de software tiene errores y el costo de la modificación es tan alto que es mejor volver a hacer el sistema que modificar el código existente. Actualmente, el mantenimiento de software representa una parte muy importante del negocio dentro de esta industria. Entre más tiempo transcurra en la detección de un error, será más costoso corregirlo, por ejemplo, se estima que el costo de corregir un error detectado durante el análisis inicial de un proyecto es la décima parte del costo de corregirlo cuando el sistema fue entregado al cliente.

El costo de la calidad.
Es el costo que se paga por hacer las cosas mal, es decir, la suma de todos los costos que desaparecerían si todo se hubiese hecho correctamente desde el inicio. El costo de la calidad es el esfuerzo y el dinero gastado en la prevención de defectos (tiempo dedicado al análisis de las causas de los defectos) y los costos asociados a las fallas de un producto. El retrabajo asociado a la corrección y prevención de los defectos que han sido detectados, puede ocupar desde un 30% a un 40% del esfuerzo total dedicado a un proyecto.

Aún cuando hay muchos ejemplos de lo costoso que pueden ser las fallas y los riesgos relacionados con el desarrollo de software, estos no se publican debido a diferentes razones, entre ellas:

- Las compañías implicadas no desean que se publiquen datos sobre estos desastres.
- Los gerentes y el personal involucrado no están dispuestos a dar entrevistas.
- Los datos sobre costos y defectos se eliminan cuando el proyecto ha sido cancelado.

- Los datos de algunos desastres de software se encuentran en litigio y se consideran confidenciales.

Un ejemplo de lo costoso que pueden ser las fallas en un proyecto de desarrollo de software es el siguiente:

En los 70's IBM canceló el desarrollo de "Future System" o FS, un sistema operativo estimado para 40,000 puntos funcionales, siendo una de las más grandes liberaciones jamás planeadas para software; sus gastos totales antes de la cancelación excedían 500 millones de dólares.[2]

Los obstáculos, costos excedidos y tiempos de retrasos asociados con el desarrollo de sistemas grandes y complejos son riesgos reales y serios para los administradores experimentados y los ejecutivos en las grandes corporaciones, así como también para organizaciones militares y de gobierno, por lo que es recomendable que su administración sea llevada a cabo por gente con experiencia y preparación.

A continuación se describirán los principales factores que afectan el proceso de desarrollo de software, su administración adecuada es fundamental para obtener un producto exitoso, confiable y de bajo costo.

1.1 FACTORES QUE DEBEN SER ADMINISTRADOS EN EL DESARROLLO DE SOFTWARE.

Existen 5 puntos que deben ser administrados en un proyecto de software:

1. Funcionalidad
2. Calidad
3. Costo
4. Tiempo
5. Personal

Estos factores son dependientes, ya que si, por ejemplo, se añade personal, los tiempos pueden bajar, pero el costo se incrementará. Algo común en un proyecto consiste en acortar los tiempos o agregar funcionalidad y sacrificar calidad. En cada proyecto es necesario determinar cuales son los puntos críticos y como balancearlos entre sí, con la finalidad de alcanzar los objetivos claves del proyecto.

Cada uno de estos puntos puede tomar uno de los tres roles dentro de cualquier proyecto: *conductor, restricción, o grado de libertad.*

1) Un conductor es un objetivo clave, por ejemplo, cuando un producto debe ser entregado a tiempo a fin de cubrir la oportunidad del mercado, el tiempo es el conductor. En el software comercial, tal como procesadores de palabras y hojas de cálculo, la funcionalidad es el conductor.

2) Una restricción es un factor limitante que no está dentro del control del líder del proyecto, si se asigna un equipo de tamaño fijo a un proyecto, el personal es la restricción. El costo es una restricción en un proyecto con un contrato de precio fijo, mientras que la calidad puede ser una restricción para un proyecto que desarrollará una pieza de equipo médico o un sistema de control de vuelo. Algunas veces, se puede tener el costo como conductor y restricción ya que puede ser el principal objetivo y un factor limitante. De manera similar, la funcionalidad puede ser el principal objetivo del proyecto, pero se puede ver como restricción, si esta no es negociable.

3) Cualquier punto del proyecto que no sea conductor, ni restricción se vuelve grado de libertad que son aquellos factores que el líder de proyecto puede balancear o ajustar para alcanzar los objetivos del proyecto. Por ejemplo, en algunos proyectos de sistemas de información interna, si los conductores son la funcionalidad y la calidad, el personal una restricción, los grados de libertad serán el tiempo y el costo. La implicación para este perfil es que si todas las funcionalidades solicitadas por los clientes son incluidas, el tiempo de entrega será mayor al deseado.

En un proyecto no todo debe ser restricciones u objetivos, es importante conocer cuales pueden ser negociados, esto permitirá establecer las reglas y fronteras del proyecto. Debemos tener presente que una restricción elimina flexibilidad para el líder del proyecto, un conductor implica poca flexibilidad y los grados de libertad proveen una cobertura más amplia para balancear ese punto contra los otros cuatro.

Una forma de clasificar cada punto dentro de estas categorías es valorar la flexibilidad que tiene el líder para analizar los márgenes de negociación. Frecuentemente, solo nos enfocamos a un punto e ignoramos el impacto de los demás, por lo que surgen inconformidades, para evitar esto es necesario hacer concientes a los clientes o usuarios de que no pueden tener todas las funcionalidades, sin defectos, entregado en el menor tiempo posible, con bajos costos y desarrollados por un personal escaso. Para determinar esto el usuario o cliente deberá contestar las siguientes preguntas:

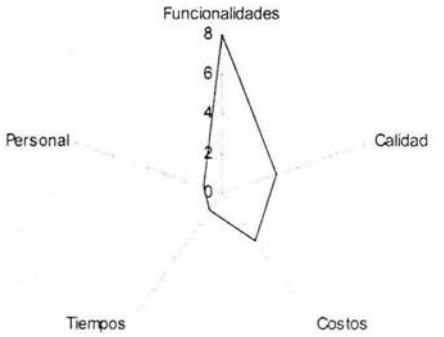
¿Es factible eliminar funcionalidad?
¿Qué pasa sí un proyecto se retraza, por cuanto tiempo?
¿Qué sucede si se eliminarán prácticas de control de calidad, para que el producto este en el mercado en la fecha acordada?

Ante estas preguntas, surgirán discusiones inesperadas, pero clientes y usuarios deben entender el impacto en tales puntos a fin de que puedan tomar las decisiones correctas y se comprometan a tiempos realistas.

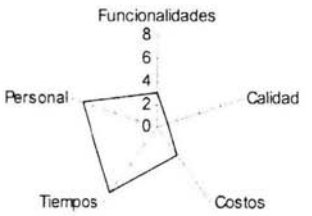
Herramientas como el diagrama de flexibilidad (diagrama de Kiviat), facilitan estas negociaciones. En esta gráfica se pueden diagramar varios valores (cinco en este

caso) como un conjunto de ejes normalizados. La posición de cada punto en el eje indica el grado relativo de flexibilidad que ese punto tiene para un proyecto en particular, desde una escala de 0 (absolutamente restringido) a 10 (completa flexibilidad).

Ejemplo:
La siguiente figura muestra un proyecto donde la restricción es el personal, por lo que el valor del eje para este componente es 0. El proyecto tiene como objetivo cumplir con las fechas de entrega, por lo que este punto también tiene un valor bajo. El proyecto presenta mayor flexibilidad en los otros puntos: funcionalidad, la calidad del producto y los costos, por lo que los valores de estos puntos son mayores en sus ejes. [8]



La siguiente figura muestra el diagrama para un proyecto en el cual la calidad es el objetivo y el tiempo es el de mayor grado de libertad.



Estos polígonos dan un aspecto gráfico de los puntos importantes de cada proyecto. Si se reduce uno de ellos, generalmente se deben ajustar los otros para compensar su impacto, ya que nada es gratis.

1.2 PRINCIPALES PROBLEMAS DE ADMINISTRACIÓN Y SUS CAUSAS EN EL DESARROLLO DE SOFTWARE

Los principales puntos para determinar si un proyecto falla o tiene éxito son los siguientes:

1. Tiempo de entrega.
2. Costos y recursos necesarios para construir las aplicaciones de software.
3. Niveles de calidad y confiabilidad del software cuando se entrega al usuario final.
4. Facilidad de uso y aprendizaje del software.
5. Soporte técnico al cliente y niveles de servicio cuando ocurren problemas.
6. Facilidad de modificación y mantenimiento a medida que la aplicación madura.

Un proyecto falla cuando no satisface los tres primeros puntos.

Los errores en el software, se han vuelto tópicos frecuentes en la literatura reciente, relativa al desarrollo tecnológico. Algunos errores solo son molestos, pero otros cuestan tiempo y dinero.

De acuerdo con una investigación realizada por GAO (Goverment Accounting Office), en 9 organizaciones, 50% de los proyectos tienen sobrecostos, el 60% fue entregado después de tiempo, y solo el 1.8% del software fue usado tal cual fue entregado. También existe el riesgo de que los proyectos sean cancelados debido a costos excesivos, demasiados defectos y una calidad muy pobre.[4]

Los estimados de tiempo y costo en un proyecto están lejos de ser acertados, para darse una idea de este punto, considérense las siguientes 'leyes de administración de proyectos' distribuidas por International Systems Inc.:

- Ningún proyecto grande es instalado a tiempo, dentro de presupuesto y con el mismo personal que lo inició.
- Los proyectos avanzan rápidamente hasta que están completos a un 90%, después permanecerán en el 90% para siempre.
- Una ventaja de los objetivos confusos de un proyecto es que permiten evitar la vergüenza de estimar los costos correspondientes.
- Cuando las cosas vayan bien, algo saldrá mal. Cuando las cosas no puedan empeorar lo harán. Cuando las cosas parezcan ir mejorando, se habrá olvidado algo.

- Si es posible cambiar el contexto del proyecto libremente, el grado de cambios excederá el grado de progreso.
- Ningún proyecto está completamente revisado. Los intentos para revisar un sistema, inevitablemente llevarán a nuevos errores que son aún más difíciles de solucionar
- Un proyecto planeado sin cuidado tomará tres veces mas tiempo de lo esperado en concluirse, uno planeado de manera cuidadosa, tomará solo lo doble de lo estimado.
- Los equipos de proyecto detestan reportar progresos, porque esto manifiesta su falta de progreso.

La actitud de "Preferible con errores a fuera de tiempo, ya que estos los podemos arreglar después" ha creado el mito de que los proyectos grandes significan problemas potenciales. Así como también lo es, que pocos de estos errores son identificados en las primeras etapas de desarrollo para tomar las acciones correctivas necesarias. Pero muchos errores se deben a que se establecen tiempos de entrega no realistas y esto se incentiva con una planeación y seguimiento de gente sin experiencia que solo cuando el tiempo está por vencerse, se dan cuenta de que los estimados son totalmente inadecuados para los compromisos que se han establecido dentro del proyecto.

Causas de los problemas en el desarrollo de software.
Entre las causas más comunes por lo que los proyectos son fallidos, o están en riesgo, se encuentran[4]:

Políticas.
Cuando las políticas se han convertido en una constante en proyectos grandes, se imponen muchísimas restricciones (tiempo, personal, presupuesto, etc.) que pueden evitar el desarrollo del proyecto. Las causas pueden ser muy variadas, pero todas estas imposiciones pueden conducir al fracaso del cumplimiento de los objetivos del proyecto. Estas políticas pueden incluir, trabajar con cierto tipo de gente, equipo o cumplimiento de trámites para liberar o probar un producto.

Promesas no realistas hechas por administradores de proyecto sin experiencia.
La ingenuidad es generalmente asociada con la inexperiencia, por lo que no es sorprendente ver compromisos no realistas hechos por gente que no tiene ni idea de cuanto tiempo o esfuerzo será requerido para construir el sistema deseado, ya que muchos de estos compromisos son hechos por personas fuera del ámbito del desarrollo de una aplicación (ventas, mercadeo, etc.). Y la gerencia se apega a estas promesas de tiempo y presupuesto no importando cuan fuera de la realidad estén.

Una actitud ingenua "Todo podemos hacerlo en el fin de semana".
La administración es uno de los chivos expiatorios más recurrentes de proyectos fallidos, sin embargo, el personal técnico no está del todo exento de culpa.

Muchas veces, se dan estimados optimistas de cuanto llevará desarrollar una aplicación, sin tomar en cuenta factores como la documentación, manejo de errores, edición de las entradas de los usuarios y/o pruebas del software. En la mayoría de estos casos la gente que da estos estimados no tiene experiencia previa en proyectos grandes y complejos que no puedan ser superados por 48 horas seguidas de codificación en el fin de semana.

La mentalidad de compañías que inician.
Las compañías que están iniciando generalmente están desprovistas de personal, experiencia, financiamiento, administración y son demasiado optimistas en lo relativo a sus posibilidades de éxito.

La mentalidad del ejército. Los verdaderos programadores no duermen.
Aunque las compañías que inician son susceptibles a este tipo de comportamiento, es común encontrarlo en grandes compañías tales como EDS y Microsoft que lo han adaptado como parte de una cultura corporativa pues consideran que solo trabajando lo doble pueden hacer frente a la competencia de un mercado difícil.

Intensa competencia creada por la globalización de mercados.
Muchas organizaciones que no han tolerado fallas grandes en sus proyectos se han visto forzadas, en algunos casos, a hacerlo recientemente, debido al incremento de la competencia asociada a la globalización de mercados. Aunque también hay factores secundarios asociados a la tecnología (Internet por ejemplo) y decisiones gubernamentales para abrir mercados protegidos o la eliminación o creación de cuotas y tarifas.

Intensa competencia debido a la aparición de nuevas tecnologías.
La introducción de tecnología nueva para establecer ventajas competitivas, puede causar una respuesta defensiva de una compañía que se ha desempeñado adecuadamente con tecnología diferente ya que muchas veces no hay quien pueda explotar sus potenciales y eliminar sus debilidades o los distribuidores no pueden proporcionar el soporte adecuado a esta tecnología.

Intensa presión causada por regulaciones gubernamentales inesperadas.
Anteriormente se mencionó la decisión del gobierno de eliminar algunas barreras para abrir el mercado a la globalización, muchos proyectos que fracasan son resultado directo de la decisión del gobierno de desregularizar las industrias de las telecomunicaciones, los servicios financieros, la industria aérea, etc. Generalmente estos cambios no son claros, tienen demasiados detalles que no se conocen sino hasta el último momento y tienen la restricción de que deben estar disponibles en una fecha impuesta arbitrariamente, ya que si no lo está, se impondrá una multa muy alta.

Crisis no planeadas o inesperadas.
No es nada sencillo planear y anticipar todas las cosas que sucederán a lo largo del proyecto, pero debemos reconocer que vivimos en un tiempo de caos y que los

proyectos fallidos son su consecuencia. Y aunque se sepa que ocurrirá una crisis que lleve al fracaso del proyecto, frecuentemente la tendencia de la administración es evitarla hasta el último momento posible, porque de que otra manera se explica el pánico que vivieron muchísimas organizaciones de sistemas con respecto al año 2000 si sabíamos que el 1 de enero del 2000 llegaría y que era una fecha que no se podía posponer.

Restricciones.
En los proyectos que fallan (aquellos que han excedido sus tiempos y presupuestos en al menos el 50%), generalmente encontramos la imposición de las siguientes restricciones:

- Los tiempos han sido reducidos a menos de la mitad del estimado por un proceso racional de estimación, de tal manera que el proyecto que normalmente se cumpliría en 12 meses tiene ahora que ser entregado en 6 meses o menos, generalmente esto se debe a las presiones de competencia en un mercado global.

- El personal ha sido reducido a menos de la mitad del número de personas que normalmente estarían asignadas para un proyecto de ese tamaño y alcance, de tal manera que para un proyecto de 10 personas, se le han asignado únicamente 5 personas. Esto puede ser causado por la creencia de que la nueva herramienta doblará la productividad del equipo pese a que no se les ha proporcionado ningún entrenamiento en su uso. Así como también se da debido al *downsizing*, reingenierías y otras formas de reducción de personal.

- El presupuesto y los recursos asociados han sido reducidos. Esto es generalmente el resultado del downsizing y otras medidas de reducción de costos. En lugar de invertir en expertos que son relativamente caros, se contrata personal sin experiencia pero relativamente menos costoso que los especialistas.

- Los requisitos de funcionalidad, acciones y otros aspectos técnicos del proyecto son el doble de lo que serían bajo circunstancias normales. Por lo que el equipo del proyecto está impuesto a usar el espacio en disco, la memoria y el equipo para al menos lo doble de las funciones que su competidor, o que el nuevo sistema debe manejar el doble de transacciones que cualquier sistema comparable. Duplicar la funcionalidad, generalmente significa hacerlo también con el trabajo que debe ser llevado a cabo.

El desarrollo de software es quizás la única industria tecnológica donde, ni los clientes, ni los administradores, ni el personal técnico tiene datos cuantitativos exactos disponibles para proyectos similares cuando se inicia uno nuevo. La falta de datos implica que cada nuevo proyecto tiende a ser tratado como un misterio, a pesar de que se han desarrollado sistemas similares en otras compañías, por lo

que la principal causa de las cancelaciones, retrasos y sobrepaso de costos tiende a ser la falla en recopilar datos históricos sobre desarrollos de proyectos terminados o actuales. Esta falla significa que la gran mayoría de los desarrollos de sistemas de software son iniciados sin que nadie tenga una noción sólida del tiempo requerido, y por lo tanto no tengan las bases de negociación correctas.

La mala administración de proyectos, tiende a ser la principal causa para muchas fallas, pero también los factores técnicos son los que diferencian a los proyectos exitosos de aquellos que han sido cancelados o tienen bastantes problemas. A continuación se enuncian algunos factores que se asocian frecuentemente a los proyectos exitosos y fallidos[2].

PROYECTOS FALLIDOS	PROYECTOS EXITOSOS
No existen datos históricos de medición para el software	Mediciones exactas para el software
No hay uso de herramientas de estimación automatizadas	Uso de herramientas de estimación en las primeras etapas del proyecto
No hay uso de herramientas de planeación	Uso continuo de herramientas de planeación
No se da seguimiento al progreso o a los problemas más fuertes	Se reporta de manera formal el progreso
No se usa una arquitectura adecuada	Planeación formal de la arquitectura
Se falla en el uso de métodos eficientes de desarrollo	Métodos de desarrollo formales
No se hacen revisiones de diseño	Revisiones formales del diseño
No se hacen inspecciones al código	Se hacen inspecciones formales al código
Se falla en la administración formal de riesgos	Administración formal de riesgos
Pruebas informales e inadecuadas	Métodos formales de prueba
Diseño y especificaciones manuales	Diseño y especificaciones automatizadas
No se usa un control formal de la configuración	Control de la configuración automatizado
Mas de 30% de atraso en los requerimientos del usuario	Menos del 10% de retraso en los requerimientos del usuario
Uso inapropiado de lenguajes de 4ª generación (4GL's)	Uso de lenguajes adecuados
Complejidad excesiva y no medida	Complejidad medida y controlada
Poco o ningún reuso de materiales certificados	Reuso significativo de materiales certificados
Errores en la definición de los elementos de base de datos	Planeación formal de los elementos de la base de datos

También hay factores culturales y sociales asociados a los patrones de éxito o falla del software. A continuación se enuncian algunos de ellos:

PROYECTOS FALLIDOS	PROYECTOS EXITOSOS
Demasiada presión de tiempo	Expectativas realistas de tiempo
Rechazo de estimados	Entendimiento de los estimados
Severa fricción con clientes	Cooperación con los clientes
Políticas corporativas divisivas	Metas administrativas congruentes
Pobre comunicación de equipo	Excelente comunicación entre los miembros del equipo
Ejecutivos novatos	Ejecutivos experimentados
Administración errónea del proyecto	Administración adecuada del proyecto
Personal técnico deficiente	Personal técnico capaz
Generalistas a cargo de la medición de la calidad, pruebas, planeación y estimación	Especialistas a cargo de tareas críticas como medición de la calidad, pruebas, planeación y estimación

Es interesante y significativo que los primeros 6 factores son fallas especificas de la administración de proyectos y otros 3 pueden ser asociadas indirectamente con una mala administración.

Todos los factores sociales están asociados con la administración, algunos de ellos no deberían ser inesperados cuando se desarrolla algo grande y costoso con base en estimaciones y planeaciones inadecuadas, tales como la presión de los clientes.

Si algunos sistemas han sido terminados a tiempo o incluso antes, dentro de sus presupuestos y tienen pocos errores después de liberados. ¿Cómo es posible que estos proyectos tengan éxito cuando tantos fallan?. Existen tres conjuntos de factores que distinguen el éxito del fracaso del proyecto: administrativos, sociales y técnicos:

- ✓ Administradores y herramientas de administración para el proyecto.
- ✓ Un control y especialistas de control adecuados.
- ✓ Herramientas y procesos de software adecuados.

Por lo general, estos proyectos hacen uso inicial de un análisis del tamaño que tendrá el nuevo desarrollo (cantidad de código que debe ser escrito, manuales de usuario, casos de prueba, etc.), también se hace necesario saber la capacidad de la gente y la tecnología de la cual se dispone para el desarrollo y otros factores que pueden influenciar las salidas del producto. Todo lo anterior permite controlar de una manera adecuada las entradas, procesos y salidas de los elementos implicados en el desarrollo de software.

Es un fenómeno interesante que, pese a que hay miles de formas de que un proyecto de software fracase, la mayoria de los proyectos que tienen éxito utilizan patrones similares de planeación, estimados y tecnologías de control de calidad.

La administración del desarrollo de software es muy problemática, los retrasos, sobrecostos y la baja calidad son comunes en proyectos grandes, algunas compañías lo han superado a través de una cuidadosa atención al control de calidad usando especialistas capacitados, herramientas de estimación y planeación, la utilización de ambientes y herramientas de desarrollo adecuadas y guardando datos exactos sobre el desarrollo de sus proyectos, para hacer las estimaciones relacionadas con proyectos similares subsecuentes.

1.3 DEFINICIÓN DE INGENIERÍA DE SOFTWARE

La ingeniería de software es considerada un subdisciplina de la ciencia de la computación que ofrece métodos y técnicas para desarrollar y dar mantenimiento al software y resolver problemas de manera práctica.

Una de las metas de la ingeniería de software es incluir en el desarrollo, las medidas necesarias para tener control sobre el progreso de un proyecto. Estas medidas ayudan a generar estimados iniciales, guiar el desarrollo a medida que progresa y revisar estos estimados. La ingeniería de software tiene como objetivo diseñar y desarrollar software de alta calidad.

La ingeniería de software se define como la disciplina tecnológica preocupada de la producción sistemática y del mantenimiento de los productos de software que son desarrollados y modificados dentro de un presupuesto y tiempo definido.

Sus metas primordiales son mejorar la calidad de los productos y aumentar la productividad y satisfacción profesional de los implicados en el trabajo.

Los procesos de ingeniería de software tienen tres fases genéricas, las cuales son *definición, desarrollo* y *mantenimiento.*

La fase de *definición* se enfoca en *que.* Esto es, durante la definición se determina la información a ser procesada, cuales son las definiciones y la funcionalidad deseada, las interfaces establecidas, las constantes de diseño y los criterios de validación que existen y se identifican los requerimientos del sistema.

Los tres pasos que se llevan a cabo son:

1. **Análisis del sistema.** Que define el rol de cada elemento dentro de sistema.

2. **Planeación del proyecto**. Una vez que se ha identificado el alcance del proyecto, se definen tareas a desarrollar y tiempos de entrega.

3. **Análisis de requerimientos**. El alcance del proyecto brinda direccionamiento, pero es necesaria una definición detallada de los elementos de la información y la función del software antes de empezar a trabajar.

La fase de *desarrollo* se enfoca en *como*. Esto es, durante el desarrollo se define como se diseñara la estructura de los datos y la arquitectura de software, como se implementarán los detalles de procedimiento y como se traducirá el diseño a un lenguaje de programación y como se implantarán las pruebas[6].

Los tres pasos que se llevan a cabo son:

1. **Diseño de software.** El diseño traduce los requerimientos a un conjunto de representaciones - algunas gráficas, otras tabulares o de lenguaje - que describen la estructura de los datos, la arquitectura y el detalle de los procedimientos.

2. **Codificación.** Los procesos deben ser traducidos a un lenguaje artificial que da como resultado instrucciones que pueden ser ejecutadas por una computadora.

3. **Pruebas de software.** Una vez que el software ha sido implementado, debe ser probado para descubrir fallas en la función, lógica o implantación.

El *mantenimiento* ocurre en el cambio que es asociado con la corrección de errores, las adaptaciones requeridas que involucra el ambiente del software y los cambios debidos a mejoras que implican los requerimientos del usuario. Esta fase reaplica los pasos de definición y desarrollo, pero a un contexto ya existente.

Los tres tipos de cambios encontrados en una fase de mantenimiento son:

1. **Corrección.** Los usuarios descubren defectos en el software.

2. **Adaptación.** A través del tiempo, el ambiente original para el software cambia. Este mantenimiento resulta en la modificación del software para contemplar los cambios en el ambiente externo.

3. **Mejora.** A medida de que el software es usado, el usuario pedirá funciones adicionales que le proporcionarán beneficios que van mas allá de los requerimientos iniciales.

Modelos de ciclo de vida

Los ciclos de vida de los productos de software se caracterizan por diferentes etapas o fases. Una fase bien definida comprende actividades (un conjunto de entregables y controles de calidad para esa fase).

El ciclo de vida puede ser dividido en desarrollo y mantenimiento. El desarrollo agrupa las actividades hasta que el software se vuelve operacional. El mantenimiento cubre la vida del sistema desde que se instaló hasta que deja de usarse. Las etapas mas usadas en el ciclo de vida del desarrollo de software son

definición del problema, análisis de requerimientos, especificaciones, diseño, codificación, pruebas, operación y mantenimiento. Todos pueden ser definidos bajo un esquema de cascada puesto que la meta una vez completada una etapa, es no regresar a la fase previa.
Este esquema es ilustrado en la siguiente figura:

Metas de cada una de estas fases:

Definición del problema.
La meta es definir el problema en la medida que sea posible en términos del usuario, es muy importante entender el problema a través de preguntar que y porque, además de hacer la evaluación del problema que conlleva a su objetivo, las características de una solución viable. Todo esto para establecer un criterio de solución y las prioridades para partes del problema y su solución.

Análisis de requerimientos.
Una vez que el problema ha sido entendido, los requerimientos y la factibilidad de la solución deben ser establecidos. El análisis de requerimientos busca determinar las características de una solución aceptable y de las herramientas, recursos y gente disponible para el desarrollo de la solución. Los requerimientos frecuentemente tienen conflictos entre si y son o no factibles debido a las herramientas y técnicas disponibles. Tiene tres objetivos principales:

- ✓ Debe existir un entendimiento claro entre el usuario y el equipo de desarrollo de que debe dar una solución.

- ✓ Debe existir un acuerdo del rango de aceptación y posibles negociaciones, resultantes de un criterio de aceptación basado en objetivos y constantes.

- ✓ Debe hacerse un plan del proyecto para implantar la solución con un presupuesto, tiempo y requerimientos que deben ser cumplidos en cada fase. Este plan indica como serán desarrolladas las actividades su funcionalidad y sus propiedades.

Especificaciones.
Su objetivo es describir que aspecto tendrá la solución. Describiendo que tipo de entradas procesará el sistema, que funciones desarrollará para cada entrada, cuales serán las salidas correspondientes y si el sistema cumple los requerimientos de acuerdo al plan, o si debe ser modificado.

Diseño
Mientras que las especificaciones concentran el qué de la solución, el diseño describe el cómo. El diseño estructura el sistema en sus partes lógicas y funcionales, así como también selecciona las estructuras de datos y algoritmos adecuados para la implantación de las entradas, salidas y funciones del sistema.

Codificación
Su objetivo es traducir la solución a código. Ningún código debe ser escrito hasta que la fase de diseño está terminada. Esta fase termina, cuando todo el código para el sistema está escrito y documentado, las compilaciones no tienen errores y siguen las reglas y estándares del proyecto.

Pruebas.
El código escrito debe ser probado de manera rigurosa basado en las características de calidad requeridas. Las pruebas generalmente conllevan varios pasos. Tan pronto como el código ha sido escrito debe ser probado. Las primeras piezas son probadas por separado, esto se conoce como pruebas unitarias. Después se prueban en grupos para ver si interactúan de manera apropiada, estas pruebas son llamadas pruebas de integración. Por último se realiza la prueba del sistema.

Una prueba de aceptación determina si el usuario está convencido de que sus requerimientos son cumplidos.

Operación y Mantenimiento
Después de que el software ha sido instalado de manera exitosa, debe mantenerse operando. El mantenimiento corrige errores, adapta el software a nuevos ambientes, e implementa nuevos usos o los modifica así como mejora el software. Maneja los problemas y les da seguimiento.

En la siguiente tabla se listan los entregables del ciclo de vida del desarrollo de software:

FASE		ENTREGABLES
Definición del problema	PORQUE	Documento de la definición del problema, propuesta, objetivos y restricciones principales.
Análisis de requerimientos	QUE	Documento de requerimientos, tiempos (plan), presupuesto preliminar, objetivos, restricciones, criterios de pruebas de aceptación, reportes de los estudios de factibilidad
Especificaciones	QUE	Documentos de especificaciones (manual de usuario preliminar), casos de prueba, plan de desarrollo
Diseño	COMO	Documento de diseño (arquitectura, diseño detallado) casos de prueba de diseño
Codificación	COMO	Documentación del código, compilaciones sin errores de acuerdo a los estándares, casos de prueba, plan final de pruebas
Pruebas	QUE TAN BIEN	Código de pruebas, documentación finalizada, que incluye manuales de capacitación y de usuario, un acuerdo de aceptación firmado, contrato de mantenimiento y evaluación post proyecto
Mantenimiento	Los descritos anteriormente	Bitácora de problemas, documento de control de versiones, evaluaciones de calidad.

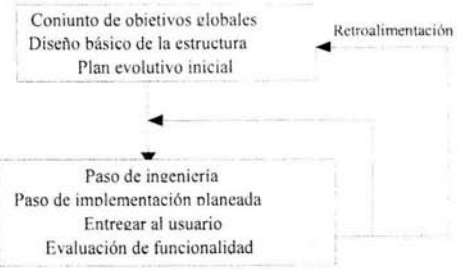
Modelos de ciclo de vida no secuenciales

Estos son modelos que no requieren conocimiento completo de todos los requerimientos. Permiten ciclos iterativos dentro del desarrollo de software y enfatizan la retroalimentación para mejorar los pasos del desarrollo.

El ciclo de vida evolutivo

Este modelo inicia con un conjunto de objetivos que establecen la calidad y los costos asociados con las metas. Todas las fases grandes del modelo lineal de ciclo de vida son divididas en partes más pequeñas, esta división se hace en consideración a la siguiente filosofía: ¿Cuál es la forma más sencilla y económica de desarrollar algo que sea útil para alcanzar el objetivo final?. Se eligen actividades críticas de éxito, aquellas sin las cuales el sistema no tendría éxito y en subsecuentes iteraciones se añadirán otras que sean sumamente útiles. Después se construye un sistema parcial, partiendo de un diseño inicial que sea fácil de cambiar y adaptar. En ciclos posteriores, se modificará el sistema, pero la idea es que los cambios nunca sean muy grandes. La ventaja de esta aproximación iterativa es que los usuarios ven algo mucho antes de que el sistema entero este construido.

En un desarrollo evolutivo, los cuatro pasos de cada iteración a través del ciclo están relacionados a la especificación, diseño, código, pruebas y un monitoreo continuo. La diferencia entre este y el modelo de cascada se encuentra en la retroalimentación y el tamaño. Los entregables son similares, pero no sus tamaños.



Ciclo de vida evolutivo.

El ciclo evolutivo es una técnica para estructurar el proceso de desarrollo y una mejora iterativa que recomienda estructurar el problema para permitir el diseño y la implantación de subconjuntos de soluciones de problemas que conlleven mejores resultados.

El ciclo eterno de desarrollo

Este modelo cierra el ciclo entre la primera y la última fase de desarrollo y considera reevaluaciones y cambios continuos. A diferencia del ciclo de vida evolutivo, la retroalimentación se lleva a cabo en la última fase y siempre conduce a modificaciones en los objetivos, ya que no pueden aplicarse simplemente en los objetivos actuales. A través del ciclo de vida, se evalúa la calidad y se analiza que estuvo bien y que estuvo mal, y basado en esto, se definen nuevos objetivos y se reinicia el ciclo de vida. De esta manera un producto de software evoluciona y cambia con su ambiente.

Ciclo de vida de sistemas expertos

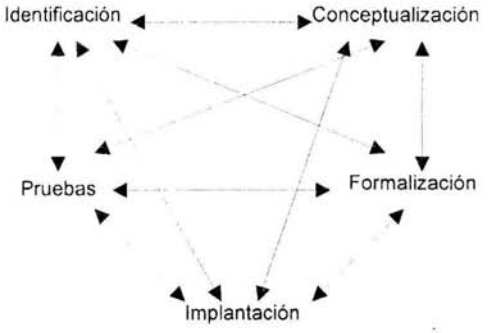
Este ciclo de vida maneja el problema de no saber de antemano que se llevará a cabo dentro del desarrollo y la implantación de un nuevo sistema en una área específica. La solución de este problema requiere frecuente retroalimentación entre todos los pasos del desarrollo y estos no pueden ser separados como en los modelos anteriores. Este tipo de sistema evoluciona gradualmente, necesita mucha experimentación y generalmente son desarrollados de manera evolutiva o paulatina, es decir, a través del tiempo, se añaden soluciones o funcionalidades aun conjunto mínimo o central de funcionalidades.

El primer paso es la identificación del problema, se identifican los recursos necesarios y las metas de desarrollo. Debido a que resolver un problema completo

puede ser muy complejo, se identifica una tarea o un subproblema representativo con lo que se inicia el conocimiento de esta tarea o subproblema. Después se pone el conocimiento en un esquema conceptual y se tratan de encontrar los conceptos el flujo de la información a través del proceso de solución del problema, tras lo cual se desarrolla una representación formal en la cual se pueden mapear estos conceptos, esta representación puede incluir lenguaje y herramientas, el objetivo de este paso es diseñar estructuras formales para organizar el conocimiento. El siguiente paso es la implantación se formulan reglas y estructuras de control que representan los conceptos y el conocimiento formal. Este resultado es un programa prototipo que muestra que tan bien se realizó la conceptualización y formalización del conocimiento del problema. Este paso conlleva diseño, codificación y pruebas. Se evalúa la funcionalidad del prototipo, lo que proporciona retroalimentación para la revisión. Esta retroalimentación corresponde a un objetivo bien definido de la vida operacional del prototipo.

Las etapas en este modelo no están bien definidas ni son independientes. El uso del prototipo puede requerir corregir o revisar los resultados de las etapas preliminares, se puede ir de una etapa a cualquier etapa anterior del modelo pues los ciclos iterativos son parte esencial de este.

A medida que el sistema crece, la base de conceptos se puede hacer inmanejable tanto en tamaño como en organización, lo que en algunos casos requiere una nueva forma de conceptualizar. El conocimiento debe ser reorganizado de tal manera que este estructurado de manera adecuada para el proceso.



Ciclo de vida de sistemas expertos

1.4 MODELOS DE INGENIERÍA DE SOFTWARE

Hay una enorme cantidad de modelos de calidad en el mercado, algunos son para atacar determinados sectores de la industria del software, tales como pruebas o documentación y otros conllevan el desarrollo de un proyecto, proporcionando los procedimientos a seguir a través de cada una de sus fases, así como aquellos que establecen los patrones de conducta no solo para un proyecto en especifico, sino

para todos los que van a ser desarrollados por la organización.

Los objetivos de estos modelos son:

- ✓ Brindar apoyo a todas las personas involucradas en el desarrollo de software para encontrar los errores en este.
- ✓ Mejorar la productividad y las comunicaciones e incrementar su calidad.
- ✓ Dar ideas de como ejecutar los procesos de software.
- ✓ Cubrir diferentes aspectos relativos a los tiempos para las actividades de desarrollo y como deben ser llevadas a cabo estas
- ✓ Brindar información de administración de proyectos.
- ✓ Los estándares, procedimientos y guías dan el fundamento para prácticas de ingeniería de software dentro de una organización.

Un **estándar** es una oración escrita de como debe ejecutarse un trabajo de acuerdo a conjunto específico de reglas. Los estándares generalmente son establecidos por sociedades profesionales tales como el IEEE, un cuerpo oficial de estándares como ISO o instituciones gubernamentales tales como el DoD. Un estándar puede especificar los contenidos de los documentos requeridos, las actividades que deben ser llevadas a cabo.

Un **procedimiento** constituye una fórmula de paso por paso de como llevar a cabo una tarea específica.

Una **guía** es una sugerencia que ayuda al interesado a aplicar técnicas establecidas, sugieren formas prácticas de aplicar ciertos estándares.

A continuación se da una descripción de que son los estándares, guías y procedimientos, y se citarán algunos de los modelos y estándares más conocidos.

Estándares del IEEE

La siguiente tabla lista los estándares de ingeniería de software, prácticas recomendadas y guías que actualmente proporciona el IEEE, muchos de ellos también son aceptados por ANSI. La gran mayoría son prácticos y están claramente escritos.

NÚMERO DE DOCUMENTO IEEE	TITULO
610.12	Glosario de términos de la ingeniería de software
730	Estándares de los planes de medición de la calidad
828	Estándares de planes de la administración configuración de software
829	Estándares para la documentación de pruebas del software
830	Prácticas recomendadas para especificar requerimientos de software
982.1	Diccionario estándar de las medidas para producir software confiable
982.2	Guía para el uso del diccionario estándar de las medidas para producir

NÚMERO DE DOCUMENTO IEEE	TITULO
	software confiable
990	Prácticas recomendadas para Ada como lenguaje de programación
1002	Taxonomía para estándares de ingeniería de software
1008	Estándares para pruebas de software
1012	Estándares para verificación y validación del software
1016	Prácticas recomendadas para descripciones del diseño de software
1016.1	Guía para las descripciones del diseño de software
1028	Estándares para auditorias y revisiones del software
1042	Guía para la administración de la configuración de software
1044	Clasificación estándar para las anomalías del software
1045	Estándar de las métricas de productividad de software
1058.1	Estándares para planes de administración de proyectos de software
1059	Guía para los planes de validación y verificación de software
1061	Estándar de metodología de métricas de la calidad del software
1062	Prácticas recomendadas en la adquisición de software
1063	Estándar para la documentación para el usuario del software
1074	Estándar para desarrollar procesos de ciclo de vida
1209	Prácticas recomendadas para evaluación y selección de herramientas CASE
1219	Estándares para mantenimiento de software
1228	Estándares para planes de seguridad del software
1298	Estándares para los sistemas de administración de la calidad del software 1ª parte: requerimientos

Estándares de ISO

ISO (International Standar Organization) ha desarrollado diversos modelos para la calidad de software, los mas conocidos dentro de la industria de software son los estándares ISO 9000, debido a que se difundieron ampliamente en Europa; resulta ventajoso tener una certificación en ellos, si se desea obtener un contrato de un cliente europeo ya que establecen las reglas mínimas para la certificación de esta organización, aunque también se han desarrollado otros estándares, por parte de esta organización, para el desarrollo de software y los procesos involucrados en esta actividad.

ISO 9000

Su enfoque primario es hacia la relación cliente - proveedor, a fin de reducir el riesgo en la elección de un proveedor, identificando los requisitos mínimos de calidad para un sistema, dando una esquema en el que todos los requisitos deben ser cumplidos para obtener la certificación. La idea básica de ISO 9000 es seguir un conjunto de estándares para hacer que el éxito en los proyectos sea repetible.

ISO 9001
Tiene 2 niveles de evaluaciones: unidad de producción de software y proyecto. Es dependiente del ciclo de vida del desarrollo de software pero es independiente de las funciones, de los atributos y características de estos.

ISO/IEC 12207
Estándar de ciclo de vida que es aplicable a productos y servicios de software. Este estándar intenta agrupar a los primeros del ciclo de vida tales como el IEEE 1074 y DOD 2167a. Está dividido en tres secciones:

- ✓ Los procesos primarios cubren adquisición, provisión, desarrollo, operación y mantenimiento.
- ✓ Los procesos de soporte incluyen documentación, mantenimiento de la configuración, medición de la calidad, auditoria y solución de problemas.
- ✓ Procesos organizacionales que incluyen administración, mejora, infraestructura y entrenamiento.

Este estándar tiene la intención de ser independiente del ciclo de vida, esto es, puede ser aplicado a cualquier modelo de ciclo de vida.

SPICE
(Software Process Improvement Capability and dEtermination) es un conjunto de estándares que combina herramientas, modelos y estándares tales como el CCM, Trillium de Bell, BT de Healthcheck y STD de Compita.

SPICE tiene 5 niveles de madurez, similares a los del CMM: ejecutado, planeado y seguido, definido, administrado y mejorado. Estos niveles se aplican a procesos individuales y las mediciones dan un grado de que también el proceso satisface los requerimientos. Los rangos de satisfacción van de total, ampliamente, parcialmente a de ninguna manera. El uso de los estándares requiere un cambio principal de los procesos de auditoria a uno de medición. La salida de una medición da una visión de adecuación contra cada nivel.

Modelo europeo de calidad total
La Fundación Europea para la Administración de la Calidad y la Fundación Británica de la Calidad promueven la auto medición usando el modelo como un medio de mejorar el negocio. En un corto periodo de tiempo, la auto medición usando el modelo ha generado un gran entusiasmo y una mayor dirección hacia la mejora en muchas compañías europeas.

El modelo hace preguntas desafiantes sobre como se hacen las cosas y los resultados alcanzados en todas las áreas de actividad del negocio.

La auto medición es el proceso que se adopta para contestar esas preguntas para el negocio y revisar los hallazgos dando una retroalimentación sobre la cual basar planes de mejora. Su principio es planear, hacer, verificar y actuar.

El modelo Bootstrap

Consiste de 4 etapas principales: preparación, ejecución de la evaluación, determinación del nivel de madurez y capacidades y la presentación de resultados de la evaluación.

En la etapa de preparación se realizan las siguientes tareas:

- Entrenamiento inicial para tener claros los objetivos.
- Seleccionar los proyectos a ser evaluados para obtener una mejor cobertura de la UPS (Unidad de Producción de Software).
- Definir el personal de evaluación para minimizar la subjetividad de la evaluación.
- Definir el personal a ser evaluado para obtener la mejor cobertura de los roles involucrados en los proyectos seleccionados.
- Realizar el acuerdo de confidencialidad.

En la ejecución las tareas son:

- Una breve reunión de apertura para obtener un enfoque colaborativo con el personal a ser entrevistado.
- El llenado de los cuestionarios con las características generales de la UPS.
- El llenado de los cuestionarios del proyecto elegido, incluyendo la evaluación de como el proceso de producción es aplicado.
- Revisión preliminar de la evaluación.
- Reunión final con el enfoque de presentar los resultados de la evaluación y obtener el consenso para poder pasar a la fase de mejoras.

En la etapa de determinar el nivel de madurez y capacidad, se califica cada pregunta con uno de los cinco valores posibles: nulo, débil, regular, extenso o no aplica. Para cada atributo clave se obtiene un nivel de madurez aplicando un algoritmo numérico dando que da como resultado uno de estos niveles: uno (inicial), dos (repetible), tres (definido), cuatro (administrado) ó cinco (optimizado). Estos niveles de madurez están subdivididos en cuartiles de forma que se obtenga una calificación más exacta. Los procesos de la organización y metodología se califican de uno a cinco, mientras que el de tecnología se califica con solo dos niveles A o B.

Como resultado de la evaluación, la organización recibe dos reportes, uno con los resultados de la evaluación de la UPS y otro con los resultados del proyecto evaluado. El correspondiente a la UPS contiene información como: un resumen ejecutivo, los objetivos de la UPS, los puntos débiles y fuertes, un plan de acción recomendado, etc. El reporte del proyecto contiene: comentarios del proyecto actual detallando lo referente a la organización, metodología y tecnología, los niveles de madurez para el proyecto, el plan de acción recomendado, etc.

Certificación y licencias del personal técnico de software

Un tópico emergente de importancia considerable es la certificación en varias especialidades de software. La razón de que la certificación es importante es que la ingeniería de software no es actualmente una profesión de ingeniería formal y reconocida. El desarrollo de software es un tópico importante pero indisciplinado y frecuentemente no profesional, por lo que la ingeniería de software no puede ser reconocida como una disciplina de ingeniería, por lo que han surgido diversas organizaciones en un intento de dar y facilitar el rumbo para las organizaciones de software.

Las diversas asociaciones tales como DPMA (Data Process Management Association), ACM (Association of Computing Machinery), IEEE Computer Society, IFPUG International Function Point Users Group, SCEA (Society of Cost Estimating and Analisys) y el Instituto para la certificación de profesionales de computación son demasiadas y están tan fragmentadas que es difícil que tengan en un futuro un impacto real sobre la certificación de los profesionistas, puesto que cada una está especializada en diferentes tópicos del proceso de desarrollo de software. Por ejemplo, la única organización que puede dar certificación en puntos funcionales es la IFPUG, por lo que, si una persona certificada bajo este esquema desea usar esta habilidad en una herramienta de estimación de costos de software necesitará otro examen de certificación, dado únicamente por SCEA. Ya que la estimación de la calidad también es necesaria, se debe ir a la ASQC (American Society of Quality Control) y si se hace necesaria la certificación por los estándares de ISO 9000, se debe ir a otra asociación.

Hay poca o ninguna cooperación entre las asociaciones y grupos de certificación, quienes en algunos casos, no tienen conocimiento de los demás e incluso compiten entre sí.

PUNTOS SOBRESALIENTES DEL CAPÍTULO

- Los principales factores de éxito en la industria del software son: el costo, la calidad y la oportunidad de cubrir las necesidades del mercado.

- La industria del software no ha sido muy exitosa en los rubros de calidad y costos. Sus índices de cancelaciones son muy altos y sus productos muy pocas veces son usados tal cual fueron entregados.

- Los problemas más frecuentes de esta industria son: tiempos de entrega, costos y recursos necesarios para construir las aplicaciones de software, niveles de calidad y confiabilidad del software cuando este es entregado, facilidad de uso y aprendizaje cuando se opera el software, soporte técnico al cliente y niveles de servicio cuando ocurren problemas, facilidad de modificación y mantenimiento a medida que la aplicación madura.

- La mala administración de proyectos tiende a ser la principal causa de las fallas de un proyecto de software.

- La ingeniería de software ofrece métodos y técnicas para desarrollar y dar mantenimiento al software y resolver problemas de manera práctica. Una de sus metas es incluir en el desarrollo, medidas para tener mas control sobre el progreso de un proyecto, las cuales ayudan a generar estimados iniciales, guiar el desarrollo a medida que progresa y revisar estos estimados.

- El objetivo de los modelos de ingeniería de software es apoyar a los involucrados en el desarrollo de software para mejorar la productividad y la calidad de sus productos.

II. DEFINICIÓN DE LOS COMPONENTES DEL MODELO DE EFICIENCIA Y MADUREZ

El éxito de cualquier contrato se determina frecuentemente por la eficiencia de una organización para convertir las necesidades de los clientes en productos que las satisfagan, por lo que se necesitan métodos que permitan hacerlo de manera rápida y eficiente. Muchas organizaciones están mejorando sus procesos de ingeniería de software a fin de alcanzar una ventaja competitiva dentro del mercado.

Los procesos de software en uso van desde un individualismo sin forma donde cada quién improvisa y trabaja según sus gustos, a la disciplina de procedimientos estandarizados y medibles en la cual, la calidad, costos y tiempos del proyecto son predecibles. A pesar de que los ingenieros de software y administradores frecuentemente conocen sus problemas a detalle, pueden estar en desacuerdo respecto a las mejoras más importantes que deben realizar. Sin una estrategia organizada, es difícil alcanzar un consenso entre el personal administrativo y profesional sobre la prioridad de las actividades. Para alcanzar resultados duraderos de los esfuerzos de mejora del proceso, es necesario diseñar un camino evolutivo que incremente la madurez del proceso por etapas.

El Instituto de Ingeniería de Software (SEI) define la madurez de proceso como el punto en cual este se encuentra explícitamente definido, administrado, medido y controlado.

La siguiente tabla muestra algunas escalas propuestas para describir la capacidad de una organización para producir software de calidad: a mayor nivel en la escala, mayor calidad y productividad.

Propuesto	**Término**	**Niveles**					
Page-Jones	Etapas		Anarquía	Folklore	Metodología	Métricas	Ingeniería
Weinberg	Conductas	Olvidadizo	Variable	Rutinaria	Manejable	Anticipada	Congruente
Humphrey	Niveles		Inicial	Repetible	Definido	Administrado	Optimizado

Las organizaciones que están en niveles menores tienden a producir software de alto costo con un alto riesgo de falla y retrasos en tiempos de entrega.

Page-Jones describe una secuencia de cinco etapas a través de la cual la organización evoluciona hacia prácticas más eficientes de ingeniería de software.

Weinberg describe los patrones culturales de comportamiento dentro de un grupo de desarrollo de software, reflejando el grado de congruencia entre lo que es dicho y hecho dentro de la organización. En él se incluye la conducta olvidadiza en la que considera que los implicados no saben lo que están haciendo.

Tal vez la mas conocida de la escalas anteriores es el modelo de cinco niveles del SEI (Software Engineering Institute). El modelo de eficiencia y madurez para software (CMM) da a las organizaciones guías de como controlar sus procesos para desarrollar y mantener software y como evolucionar hacia una cultura de ingeniería de software y excelencia administrativa. El CMM fue diseñado para apoyar a las organizaciones en la selección de estrategias de mejoras de sus procesos, determinando la madurez de estos e identificando los puntos mas críticos para la calidad del software y la mejora del proceso.

El SEI considera que la mejora continua del proceso está basada en muchos pasos pequeños y evolutivos mas que en innovaciones revolucionarias. El CMM provee un esquema para organizarlos dentro de cinco niveles de madurez que residen en fundamentos sucesivos para una mejora continua del proceso. Estos cinco niveles definen una escala ordinal para medir la madurez del proceso de software de una organización y para evaluar su eficiencia. Los niveles también ayudan a una organización a dar prioridad a sus esfuerzos de mejora.

Las metas del SEI (Software Institute Engineering) al desarrollar CMM deben:

- Señalar las disciplinas que tienen un impacto sobre el software.
- Proveer modelos de referencia de procesos integrados de mejora.
- Crear un consenso amplio en la comunidad.
- Armonizar estándares.
- Proporcionar disciplinas de mejora para la eficiencia.

2.1 CONCEPTOS DEL PROCESO DE MADUREZ.

Antecedentes.
Los gastos en software se incrementan aproximadamente un 12% cada año, y la demanda por funcionalidades adicionales del software crecen aun más rápido. El Departamento de la Defensa de EU, un gran usuario de software, se hizo intolerante ante el estado de ineptitud en el desarrollo de software, por lo que en 1984 formo el SEI para establecer estándares de excelencia para la ingeniería de software y acelerar la práctica de métodos y tecnologías avanzadas. Consecuentemente, con el apoyo del MITRE, una organización no lucrativa que subsidian agencias gubernamentales de EU, el SEI anuncio su esquema de madurez para procesos de software en 1987.

El CMM describe características que se deben cumplir para realizar correctamente las prácticas de ingeniería de software, no es un proceso, sino un conjunto de características y requerimientos dentro de los procesos de ingeniería de software. Estas características pueden ser usadas como guías para desarrollar, medir y mejorar los procesos de una organización. Es en si, un esquema que describe los elementos claves de un proceso eficiente de software. Describe un sendero evolutivo de procesos a la medida a procesos disciplinados.

Este esquema recibió una retroalimentación positiva tanto de la industria de software como del gobierno, después de diversas mediciones, este esquema de madurez evolucionó al CMM v1.0 en 1991 seguido por CMM v1.1 en 1992.

Se han hecho mediciones pilotos en 1994 y 1995 para ayudar a ajustar el modelo.

La estructura de niveles del CMM está basada en principios de calidad que han existido desde hace 60 años. En los 30's, Walter Shewart, promulgo los principios del control estadístico de calidad. Sus principios fueron desarrollados y exitosamente demostrados en el trabajo de W. Edward Deming y Joseph Juran. Phil Crosby establece en 1979 su esquema de madurez de la administración de la calidad describiendo cinco etapas de la madurez en la administración de la calidad: Incertidumbre, Conciencia, Ilustración, Sabiduría y Certeza. Estos principios han sido adaptados por el SEI dentro de la estructura de madurez que establece una administración de proyectos y fundamentos de ingeniería para el control cuantitativo del proceso de software, el cual es la base para una mejora continua.

El esquema de madurez dentro del cual estos principios de calidad han sido adaptados, fue inspirado primeramente por Philiph Crosby de su libro "La calidad es gratis". El modelo de madurez de Crosby describe cinco etapas evolutivas para adoptar prácticas de calidad. Este esquema de calidad fue adaptado al proceso de software por Ron Radice y sus colegas, trabajado bajo la dirección de Watts Humphrey en IBM. Humphrey trajo este esquema de madurez al Instituto de Ingeniería de Software (SEI) en 1986, añadió el concepto de niveles de madurez y desarrollo los fundamentos para su uso actual en la industria de software. [11]

Esquema de madurez.

Las primeras versiones del esquema de madurez de Humphrey son descritas en los reportes técnicos del SEI y en su libro "Administrando el proceso de software". Un cuestionario preliminar de madurez fue liberado en 1987 como herramienta para proveer a las organizaciones de una forma para identificar la madurez de sus procesos de software. Dos métodos, la evaluación del proceso y de la eficiencia del software, fueron desarrollados para estimar el proceso de madurez de software en 1987. Desde 1990, el SEI, con la ayuda de mucha gente del gobierno y la industria, ha expandido y refinado el modelo basado en muchos años de experiencia en su aplicación a la mejora del proceso.

El esquema de madurez ordena las etapas evolutivas del desarrollo de software de tal manera que las mejoras en cada etapa dan el fundamento para los pasos que deben ser llevados a cabo en las siguientes. Así que, una estrategia de mejora proveniente de un esquema de madurez da un mapa para mejoras continuas del proceso. Esto conduce hacia avances e identifica las deficiencias en la organización, pero no intenta dar un arreglo rápido a proyectos en problemas.

Enfocándose en un limitado conjunto de actividades y trabajando agresivamente para alcanzarlas, una organización puede mejorar constantemente su proceso de software para habilitar mejoras continuas y duraderas en la eficiencia de sus procesos de software.

Características de cada nivel de madurez
A continuación se enlistan las características de cada uno de los cinco niveles de madurez establecidos por el SEI.

1. Inicial
El proceso de software se caracteriza por ser a la medida y en algunas ocasiones caótico. Pocas actividades están definidas y el éxito depende de esfuerzos individuales.

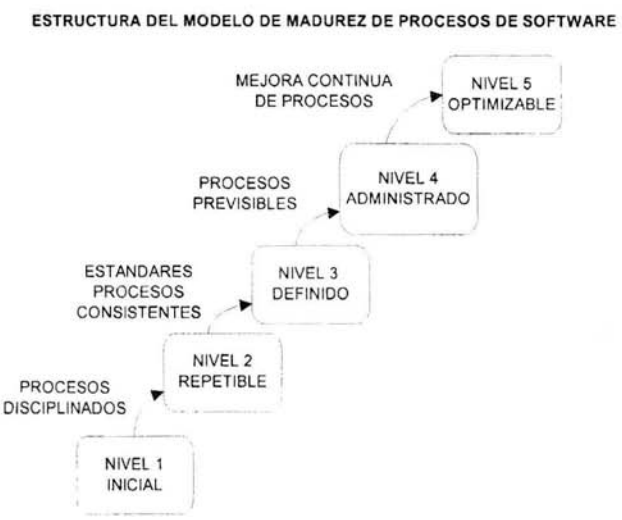
2. Repetible
Se establecen actividades básicas de administración del proyecto para monitorear costos, tiempos y funcionalidad. La disciplina necesaria del proceso se aplica para repetir éxitos preliminares en proyectos similares.

3. Definido
El proceso de software para las actividades de administración e ingeniería se documenta, estandariza e integra dentro de un proceso estándar para la organización. Todos los proyectos usan una versión aprobada y adecuada de este para desarrollar y dar mantenimiento.

4. Administrado
Se recopilan medidas detalladas del proceso de software y la calidad del producto y son cuantitativamente entendidos y controlados.

5. Optimizado
La mejora continua del proceso se establece por una retroalimentación cuantitativa y a través de probar ideas innovativas y tecnologías.

En la siguiente figura se muestra el ciclo de evolución del proceso de madurez:



Proceso

De acuerdo al diccionario Webster, un proceso es "un sistema de operaciones para producir algo... una serie de acciones, cambios o funciones para alcanzar un fin o un resultado": El IEEE define un proceso como "una secuencia de pasos ejecutados para un propósito dado". Un proceso de software se puede definir como un conjunto de actividades, métodos, prácticas y transformaciones que la gente usa para desarrollar y mantener el software y los productos asociados (v.g. planes de proyectos, diseño de documentos, código, casos de prueba y manuales de usuario). A medida que una organización madura, el proceso de software se define mejor y se implanta de forma consistente a través de la organización.

El concepto de proceso de software que será usado en el presente trabajo es: *conjunto de herramientas, métodos y prácticas utilizados para desarrollar un producto de software.*

En la mayoría de las organizaciones, los proyectos se terminan fuera del tiempo y presupuesto planeado. En tales casos, la organización no provee frecuentemente la infraestructura y el soporte necesario para ayudar a evadir estos problemas. Aunque algunos proyectos produzcan excelentes resultados, se debe generalmente a esfuerzos heroicos de un equipo dedicado, y no a repetir métodos probados derivados de un proceso de desarrollo de software definido. En ausencia de dicho proceso, repetir los resultados depende de tener a los mismos individuos disponibles para el siguiente proyecto. El éxito se basa solamente en la

disponibilidad de individuos específicos, no proporciona soporte para una productividad a largo plazo ni mejora la calidad en la organización. La mejora continua solo puede ocurrir a través de esfuerzos enfocados y sostenidos para construir un proceso de infraestructura de ingeniería de software y prácticas de administración.

Las compañías saben que necesitan mejorar, pero no tienen la información respecto a que pasos seguir y frecuentemente son incapaces de priorizar sus problemas, lo que las lleva a ejecutar iniciativas excesivas de mejora sin un enfoque específico. Si una compañía tiene poco dinero para mejoras, no tiene definido a que destinarlo y si dispone de medios los aplica de manera deficiente y sin una dirección determinada.

Organizaciones Maduras e Inmaduras

El establecimiento de metas sensibles para la mejora del proceso requiere entender la diferencia entre las organizaciones de software inmaduras y maduras.

En una organización inmadura, el proceso de desarrollo de software es generalmente improvisado y se administra en el curso del proyecto. Aun cuando un proceso ha sido especificado, generalmente no se sigue. La organización es reaccionaria y los gerentes usualmente están enfocados a la solución de crisis inmediatas, mejor conocidas como 'bomberazos'. Los tiempos y presupuestos son generalmente excedidos debido a que no están basados en estimados realistas. Cuando se imponen metas estrictas, la funcionalidad y calidad del producto se comprometen frecuentemente para apegarse al tiempo. No hay bases objetivas para juzgar la calidad del producto, la cual es difícil de predecir y hay dificultades para resolver los problemas resultantes del proceso. Las actividades encaminadas a mejorar la calidad, tales como revisiones y pruebas se reducen o eliminan frecuentemente cuando los proyectos rebasan los tiempos.

En una organización con un proceso de software maduro existe una habilidad organizacional para administrar el desarrollo de software y procesos de mantenimiento. El proceso está documentado y es comunicado al staff existente y a los nuevos empleados y las actividades se realizan de acuerdo a una planeación. El proceso se ajusta para su uso de acuerdo a las necesidades de cada proyecto y de manera consistente con la forma en que el trabajo se hace actualmente, las mejoras se desarrollan a través de pruebas piloto controladas y análisis de costo beneficio. Las responsabilidades y roles son claros para el proyecto y la organización. Se da seguimiento a la calidad del software y la satisfacción del cliente. Hay un objetivo, así como bases cuantitativas para juzgar la calidad del producto y analizar los problemas de este y el proceso. Los tiempos y presupuestos están basados en históricos y son realistas, los resultados esperados por costo, tiempo, funcionalidad y calidad del producto son generalmente alcanzados.

Hacer énfasis en estas observaciones sobre las organizaciones maduras e inmaduras requiere la construcción de un esquema de madurez de software. El cual describe una vía evolutiva desde un proceso caótico a uno maduro y disciplinado. Sin este esquema, los programas de mejora pueden ser ineficientes ya que los fundamentos necesarios para sostener las mejoras sucesivas no han sido establecidos.

El esquema de madurez del proceso de software surge de integrar conceptos de eficiencia y funcionalidad.

Eficiencia del proceso.

La eficiencia del proceso de software describe el rango de resultados esperados que pueden ser alcanzados siguiéndolo. Permite predecir los resultados del próximo proyecto de la organización.

Funcionalidad del proceso.

La funcionalidad del proceso de software representa los resultados actuales alcanzados a partir de su seguimiento. Mientras que la funcionalidad se enfoca a los resultados alcanzados, la eficiencia se enfoca a los resultados esperados.

Madurez del proceso

La madurez del proceso de software se define como el momento en que este se encuentra definido, administrado, medido y controlado. La madurez implica un crecimiento potencial en eficiencia e indica tanto la riqueza del proceso en la organización como la consistencia con la cual este se aplica en los proyectos. El proceso de software en toda organización madura está documentado, es conocido y entendido, usualmente por entrenamiento y es continuamente monitoreado y mejorado por sus usuarios, por lo que su eficiencia es conocida. La madurez de un proceso de software implica que la productividad y la calidad resultante pueden ser mejoradas en el tiempo por medio de cambios consistentes en la disciplina a través de usarlo.

A medida que una organización mejora la madurez en su proceso, este se institucionaliza vía políticas, estándares y estructuras organizacionales, lo cual involucra construir una infraestructura y una cultura corporativa que soporta métodos, prácticas y procedimientos del negocio de tal manera que durará después de que aquellos que originalmente los definieron se han ido.

El concepto de madurez del proceso está basado en la noción de que algunos procesos de desarrollo proporcionan mas control que otros, ya que ciertos problemas son resueltos por métodos y herramientas (v.g., administración de la configuración).

Nivel de madurez

Un nivel de madurez es una plataforma evolutiva bien definida que sirve de apoyo para alcanzar un proceso de software maduro. Cada nivel provee un conjunto de metas que al cumplirse, establecen un componente importante del proceso, dando como resultado un incremento en la eficiencia de este en la organización.

El nivel de madurez representa de manera razonable las fases actuales de mejora en la organización, así como también una medida de mejora que es razonable de alcanzar desde un nivel anterior. Sugiriendo metas intermedias y medidas de progreso y proporcionando un conjunto de prioridades, una vez que el status de la organización es conocido.

Administración del proceso de software

Como ya se definió un proceso de software es un conjunto de herramientas, métodos y prácticas usadas para generar un producto de software. Sus objetivos son generar productos de acuerdo a un plan y de manera simultanea, mejorar la capacidad de la organización en la generación de mejores productos. Los principios básicos para ello son los de control estadístico del proceso. Se dice que un proceso estable, está bajo control estadístico si su futuro desempeño es predecible dentro de límites estadísticos establecidos.

Mejora del proceso de software

Un paso importante para el manejo de los problemas de software es tratar a toda tarea de software como un proceso que puede ser controlado, medido y mejorado. Para mejorar su proceso, se recomienda que las organizaciones sigan los siguientes pasos:

- Entender el status actual de sus procesos de desarrollo.
- Desarrollar una visión de los procesos deseados.
- Establecer una lista de las acciones requeridas de mejora en orden prioritario.
- Generar un plan para alcanzar las acciones requeridas.
- Iniciar nuevamente en el paso uno.

Para mejorar el proceso dentro de una organización, es conveniente tener una visión clara de las metas a alcanzar y alguna manera de motivar el progreso durante todo el camino.

Evaluación del proceso de software

La evaluación del proceso de software ayuda a la organización a mejorarlo a través de identificar sus problemas más críticos y establecer prioridades de mejora. Los objetivos básicos de la evaluación son:

- Aprender como trabaja la organización.
- Identificar sus principales problemas.
- Involucrar a los lideres en los cambios del proceso.

La evaluación del proceso de software es una revisión de la organización para aconsejar a su gerencia y al personal sobre como pueden mejorar su operación. Es llevada a cabo por un equipo de profesionales de software quienes tienen experiencia en evaluación y capacitación.

Las evaluaciones identifican las áreas para mejorar de mayor prioridad y brindan guías de como realizar las mejoras, están basadas en el principio de que los implicados desean mejorar su operación y que su necesidad principal es que los guíen en que y como hacerlo, aunque esto es generalmente lo que sucede, hay organizaciones que se encuentran bajo mucha presión, sus gerentes son muy inexpertos y su personal muy deficiente, por lo que requieren guía y asistencia externa.

La gente en la optimización del proceso de software

Claramente, cualquier proceso de software es dependiente de la calidad de la gente que lo implemente. Nunca hay suficiente gente buena y aunque se disponga de ella, siempre hay un límite para lo que puedan cumplir.

La optimización del proceso mejora los talentos de la gente en muchas maneras, ya que apoya a los gerentes a determinar donde se necesita apoyo y como brindarlo a las personas que lo requieren. Facilita la transmisión de conocimientos y minimiza el desperdicio de tiempo en problemas que ya han sido resueltos. Brinda un ambiente disciplinado para un trabajo profesional, pero la disciplina debe ser manejada con cuidado, ya que puede convertirse en régimen.

Además de ser un punto de administración, la calidad también es un punto económico, pese a que siempre se pueden hacer mas pruebas y revisiones, estas implican costos tanto en tiempo como en dinero, la optimización del proceso brinda fundamento para avances significativos en la calidad del software y mejoras simultaneas en la productividad. [7]

2.2. CARACTERÍSTICAS DE COMPORTAMIENTO DE LOS NIVELES DE MADUREZ

Los niveles de madurez del dos al cinco pueden ser caracterizados a través de actividades ejecutadas por la organización para establecer o mejorar el proceso de software, por actividades llevadas a cabo en cada proyecto y por la eficiencia del proceso a través de los proyectos. Se incluye una caracterización del nivel uno para establecer una base de comparación para mejoras del proceso a niveles de madurez más altos.

Nivel 1. El nivel inicial

Sus características son: caótico, no es posible predecir costos, tiempos y calidad de los productos a ser entregados.

Las acciones necesarias a llevar a cabo son:
Planeación (estimados de costo, tiempo y tamaño), seguimiento de la funcionalidad, control de cambios, administración de compromisos y medición de la calidad.

En el nivel inicial, la organización típicamente no provee un ambiente estable para desarrollar y mantener software. Como la organización carece de prácticas de administración, los beneficios de buenas prácticas de ingeniería de software son minimizadas por planeación ineficiente y por sistemas reactivos.

Durante una crisis, los proyectos típicamente abandonan procedimientos planeados y se abocan a codificar y probar. El éxito depende enteramente de tener un administrador excepcional y un equipo maduro y eficiente. Ocasionalmente, administradores eficientes pueden soportar las presiones para tomar atajos en el proceso de software, pero cuando dejan el proyecto, su influencia estabilizadora se va con ellos. Aun un proceso de ingeniería fuerte no puede superar la inestabilidad creada por la ausencia de prácticas de mantenimiento.

La eficiencia del proceso de software del nivel uno es impredecible porque este se modifica constantemente a medida que el trabajo avanza -i.e., es a la medida-. Los tiempos, presupuestos, funcionalidad y la calidad del producto son usualmente impredecibles. La funcionalidad depende de la capacidad individual y varia con las habilidades, conocimientos y motivaciones. Hay pocas actividades estables en evidencia y la eficiencia puede ser apreciada solo por capacidad individual en lugar de organizacional.

En este nivel la anarquía prevalece, por ejemplo, los programadores creen que son artistas creativos que no deben estar sujetos a reglas o procedimientos. Pueden existir estándares pero generalmente son ignorados y las herramientas son usadas descuidadamente, pueden existir metodologías como análisis y diseño estructurado, pero son usados de manera 'informal' y a capricho de los programadores, lo que lleva a que sus resultados no sean predecibles, ya que puede que el proyecto se termine antes y dentro del costo o sobrepase significativamente tanto en tiempo como en costo o puede ser que solo necesite una persona o tal vez diez. El resultado del proyecto no depende del proceso usado para desarrollar el software (no hay ninguno) o de la habilidad del gerente, sino enteramente de la capacidad (y del temperamento y animo) de las personas involucradas en él.

Irónicamente, la organización en el nivel uno generalmente sabe que las cosas están mal, pero no tiene ni idea de como mejorar y muchas veces cometen el error de asumir que sus problemas van a ser solucionados contratando mas gente.

En este nivel, los profesionales son llevados de crisis en crisis por prioridades y cambios no planeados. Tales grupos son difíciles de identificar inmediatamente, pero a lo largo del tiempo son fácilmente identificables debido a que no cumplen sus compromisos, pese a que el gerente presenta un historial convincente e impresionante e incluso pueden cumplir con sus revisiones intermedias, pero frecuentemente se presenta una crisis de último minuto que arruina el plan. Para sus clientes, tales organizaciones son toleradas solo porque no hay otra alternativa. De igual manera la gerencia no cree en nada de lo que la gente de sistemas les diga, los gerentes de proyectos son frecuentemente sustituidos, la gerencia hace demandas irracionales a la vez que corta recursos, los tiempos son prioritarios y cuando se entregan los productos, a nadie le gusta el resultado. Existen planes a la medida, los tiempos son arbitrarios, el control del diseño no existe y los recursos siempre son inadecuados. Poca gente aprende de los errores pasados y aquellos que lo hacen, están tan desacreditados que tienen poca influencia.

Aunque la principal causa de un ambiente caótico es la falta de disciplina en los compromisos, también existen otras fuerzas, entre ellas están:

- ✓ Bajo una presión extrema, los gerentes de software frecuentemente suponen en lugar de planear. Cuando esta suposición es demasiado corta, que generalmente lo es, el caos es asegurado. Los compromisos intuitivos pueden ser correctos, pero generalmente lo son solo cuando la escala y función del proyecto son similares a experiencias anteriores del encargado.

- ✓ Cuando el progreso se pone difícil, hay una enorme tentación de creer en la magia, en que aparezca un salvador o en que una nueva tecnología será la respuesta. Debido a que estas esperanzas son excusas por no planear, esto solo pospone el problema.

- ✓ La escala de los proyectos de software tiene un ciclo. Los programas conllevan mas código del esperado, a medida de que se vuelven más largos surgen nuevos aspectos técnicos y administrativos. Debido a que no existe experiencia previa, hay sorpresas y a medida que la escala crece, estas continúan, pero con incremento de costos.

- ✓ Aun cuando se alcanza un nivel alto de madurez, una nueva administración, cambios tecnológicos, o una mayor competencia añaden presión al proceso.

Existen muchos desacuerdos sobre el control de la administración, rara vez la estimación de tiempos y costos es el primer paso. [7]

Nivel 2. El nivel repetible

Sus características son: intuitivo, los costos y la calidad son muy variables, hay control razonable de tiempos, métodos y procedimientos a la medida.

Las acciones necesarias a llevar a cabo son:

- ✓ Desarrollo de procesos estándar y definiciones.
- ✓ Asignación de recursos al proceso
- ✓ Establecimiento de métodos para requerimientos, diseño, revisión y pruebas.

En un nivel repetible, se establecen las políticas para administrar un proyecto de software y los procedimientos para implantarlas. La planeación y administración de nuevos proyectos se basan en experiencias similares. Un objetivo a alcanzar en el nivel dos, es institucionalizar procesos eficientes de administración para proyectos de software que permitan a las organizaciones repetir prácticas exitosas desarrolladas anteriormente, a pesar de que procesos específicos son implantados, los proyectos pueden diferir. Un proceso efectivo puede ser caracterizado como uno practicado, documentado, reforzado, capacitado, medido y posible de mejorar.

Los proyectos en organizaciones en el nivel dos tienen instalados controles de administración básica. Los compromisos realistas de proyectos se basan en los resultados observados previamente y en los requerimientos actuales. Los administradores de software monitorean costos, tiempos y funcionalidad; los problemas para cubrir los compromisos se identifican cuando surgen. Los requerimientos de software y los productos de trabajo desarrollados para satisfacerlos tienen una línea base y su integridad está controlada. Los estándares del proyecto están definidos y la organización se asegura de que sean seguidos fielmente. Si proyecto de software trabaja con contratistas se establece una fuerte relación cliente - proveedor.

La eficiencia del proceso de software de organizaciones en el nivel dos puede ser resumida como una disciplina, ya que la planeación y monitoreo son estables y los éxitos anteriores pueden ser repetidos. Los procesos están bajo un control efectivo de un sistema de administración de proyectos, siguiendo planes realistas basados en la funcionalidad de proyectos previos.

Las características claves de una organización en el nivel dos es que es estable y está bajo control, cumple sus presupuestos y tiempos dentro de una variación estadística aceptable, pero otra característica que tiene es que cumple sus objetivos no a través de herramientas CASE o metodologías de ingeniería de software, sino usando administración de proyectos.

Mientras que una organización de nivel uno es caracterizada por la anarquía, una que está en el nivel dos tiene un consenso general de que todos deben hacer las cosas de la misma manera, pero aún no están escritas las políticas y los

procedimientos de una manera formal. Ha enviado a todo su personal implicado en proyectos de desarrollo de software a cursos sobre una metodología común.

En este nivel, el éxito de un proyecto ya no depende de la habilidad del personal técnico, sino de la habilidad del administrador, pues sobrevivirá a la pérdida de uno de sus miembros del staff técnico, mientras que una organización en el nivel uno no, pero si el administrador del proyecto se marcha, probablemente el proyecto falle.

A medida que un proyecto crece en tamaño y complejidad, la atención hacia asuntos técnicos cambia hacia asuntos administrativos y organizacionales. Este es el enfoque del proceso de madurez. El proceso permite a la gente trabajar mas eficientemente al incorporar, en procesos documentados, las lecciones aprendidas por el mejor staff, desarrollando habilidades necesarias para ejecutar estos procesos eficientemente -usualmente vía entrenamiento-, y continuamente mejorando el aprendizaje de la gente que ejecuta el trabajo.

Una vez que la organización ha realizado una evaluación está en posición de dirigirse a sus prioridades de mejora.

Para alcanzar el nivel dos, la administración debe enfocarse a disciplinar sus propios procesos. El nivel dos proporciona los fundamentos para el nivel tres, debido a que se enfoca en la acción administrativa de mejorar los procesos antes de abordar los puntos técnicos y organizacionales del nivel tres. La administración establece una posición de liderazgo al alcanzar el nivel dos a través de documentar y seguir procesos de administración de proyectos.

Los procesos pueden diferir entre proyectos en organizaciones en el nivel dos, el requerimiento organizacional para alcanzar el nivel dos es el establecimiento de políticas que guíen a los proyectos en la obtención de procedimientos administrativos apropiados. Al estar estos documentados, dan el fundamento para procesos consistentes que puede ser institucionalizados a través de la organización con la ayuda de entrenamiento y seguridad en la calidad del software.

En el nivel repetible, la organización ha alcanzado una funcionalidad estable a través de iniciar una administración rigurosa de costos, tiempos y cambios. Esta estabilidad, por lo general involucra un rango de acciones diseñadas a alcanzar control sobre tiempos y costos. Estas acciones incluyen organización, administración de proyectos y procesos, así como tecnología. [7]

Nivel 3. El nivel definido

Sus características son: Cualitativo tiene costos y tiempos confiables, la calidad de sus productos ha mejorado pero aún es impredecible.

Las acciones necesarias a llevar a cabo son:

- ✓ Establecer medidas para los procesos y metas cuantitativas de calidad, planes, medición y seguimiento.

En el nivel definido, los procesos estándar para desarrollar y mantener software a través de la organización están documentados, incluyendo los de ingeniería de software y procesos de mantenimiento y son integrados dentro de un todo coherente. Este proceso estándar es referido en el CMM como el proceso de software estándar de la organización. Los procesos establecidos en el nivel tres son usados (y cambiados cuando es necesario) para ayudar a los administradores de software y al staff técnico a trabajar mas eficientemente. La organización explota las prácticas de ingeniería de software cuando estandariza sus procesos. Hay un grupo que es responsable de las actividades del proceso de software, por ejemplo, un grupo de procesos de ingeniería de software. Un programa de entrenamiento que abarque a toda la organización es implantado para asegurar que el staff y los administradores tengan los conocimientos y habilidades requeridas para estar a la altura de sus roles asignados.

Los proyectos modelan el proceso de software estándar para desarrollar sus propios procesos de software, los cuales adquieren características únicas en el proyecto. Estos procesos a la medida, son conocidos en el CMM como procesos definidos. Un proceso definido de software contiene un conjunto coherente e integrado y bien definido de actividades de ingeniería de software y administración. Puede ser caracterizado como aquel que incluye entradas, estándares y procedimientos para ejecutar el trabajo, mecanismos de verificación - tales como revisiones- y salidas. Debido a que está bien definido, los administradores tienen una buena percepción dentro del progreso técnico de todos los proyectos.

La eficiencia del proceso de software en el nivel tres puede ser resumida como estándar y consistente, debido a que las actividades de ingeniería de software y de administración son estables y repetibles. Las líneas de producto están bajo un control establecido de costo, tiempo y funcionalidad y la calidad del software es monitoreada. Esta eficiencia del proceso está basada en un entendimiento común dentro de la organización de las actividades, roles y responsabilidades.

El punto clave de una organización en el nivel tres es que su proceso ha sido codificado e institucionalizado, todos en la organización pueden señalarlo y decir "Esta es la forma en que hacemos las cosas aquí". Debido a que es formal y está escrito, es posible mejorarlo. Al final del proyecto, los ingenieros de software y administradores tienen algunas ideas de como hacerlo.

El nivel tres proporciona los fundamentos de administración de proyectos a través de definir, integrar y documentar las entidades del proceso de software. La integración en este caso, significa que las salidas de una tarea dan las entradas de la siguiente. Cuando hay falta de consistencia entre tareas, esta se identifica y se direcciona en etapas planeadas, en lugar de hacerlo cuando son encontradas.

Uno de los retos del nivel tres es construir procesos que fortalezcan a los involucrados en el trabajo sin ser demasiado rigurosos. [7]

Nivel 4. El nivel administrado

Sus características son: cuantitativo, existe un control estadístico razonable sobre la calidad de los productos.

Las acciones necesarias a llevar a cabo son:

- ✓ Establecer planes cuantitativos de productividad y seguimiento, ambientes de procesos e inversiones tecnológicas justificadas.

En el nivel administrado, la organización pone metas de calidad cuantitativas tanto para productos de software y procesos. La productividad y la calidad son medidas para las actividades importantes del proceso de software a través de todo el proyecto, como parte de un programa de medición organizacional. Una base de datos es usada para recopilar y analizar los datos disponibles. Los procesos de software son instrumentados con medidas consistentes y bien definidas en el nivel cuatro. Estas medidas establecen el fundamento cuantitativo para evaluar los procesos y productos del proyecto.

Los proyectos obtienen el control de sus productos y procesos a través de ensanchar la variación en la funcionalidad del proceso para caer dentro de límites cuantitativos aceptables. Variaciones significativas en la funcionalidad pueden ser diferenciadas de la variación aleatoria -ruido-, particularmente dentro de líneas de producto establecidas. El riesgo involucrado de moverse en la curva de aprendizaje de una nueva aplicación es conocida y manejada cuidadosamente.

La eficiencia del proceso de software en el nivel cuatro puede ser resumida como predecible, debido a que es medido y opera dentro de límites medibles. Este nivel de eficiencia permite a una organización predecir tendencias y la calidad de productos dentro de límites cuantitativos. Cuando estos son excedidos, se toma una acción para corregir la situación. Los productos de software son predecibles y de alta calidad.

La organización ha iniciado un proceso de medición del software comprensivo, mas allá de aquellos sobre costos y funcionalidad, ya que son importantes para dar inicio a la mejora de la calidad. Por lo que la característica principal de las organizaciones en el nivel cuatro es que ha iniciado un programa de medición del software (líneas de código o puntos funcionales para ver que tan grande es el producto, el número de personas involucradas y las horas invertidas así como días o meses invertidos en el proyecto), necesario para controlar presupuestos, tiempos y recursos de personal.

Pero también está midiendo el proceso los siguientes puntos:

- ✓ Cuanto tiempo se llevo cada uno de los pasos de la fase de diseño.
- ✓ Cuantos defectos se encontraron en cada fase.

- ✓ Cuanto tiempo fue necesario para revisar cada fase.
- ✓ Que tanto varían los números de proyecto a proyecto, semana a semana, persona a persona.

La primera responsabilidad, y el foco del nivel cuatro es el control del proceso. El proceso de software es administrado de tal manera que opera establemente dentro de la zona de control de calidad. Inevitablemente, hay algún desperdicio crónico, y puede haber algunos picos en los resultados medidos que necesiten ser controlados, pero el sistema generalmente es controlable. Esto es cuando el concepto de controlar causas especiales de variación, entra. Debido a que el proceso es estable y medido, cuando alguna circunstancia excepcional ocurre, la causa especial de la variación puede ser identificada y direccionada.

El énfasis en este nivel es recolectar mediciones para mejorar la calidad tanto del producto como del proceso que se usa para construirlo, por lo que una organización en este nivel tiene un grupo de medición de la calidad frecuentemente encargado de recopilar, analizar y reportar los datos de estas mediciones. [7]

Nivel 5. El nivel optimizado

Sus características son: existen bases cuantitativas para una inversión de capital en la automatización de procesos y de mejoras.

Las acciones necesarias a llevar a cabo son:

- ✓ Enfatizar la medición de procesos y en métodos para la prevención de errores.

En el nivel optimizado, la organización entera está enfocada a un proceso de mejora continua. Tiene los medios para identificar la debilidad y la fuerza de los procesos de manera proactiva, con la meta de prevenir la ocurrencia de defectos.

Los datos sobre la efectividad del proceso de software son usados para ejecutar análisis costo - beneficio de nuevas tecnologías y proponer cambios al proceso de software de la organización. Las innovaciones que explotan las mejores prácticas de ingeniería de software son identificadas y transferidas a través de la organización. Los equipos de proyectos en organizaciones que están en el nivel cinco analizan defectos para determinar sus causas. Los procesos de software son evaluados para prevenir defectos recurrentes conocidos, las lecciones son aprendidas y usadas para otros proyectos.

La eficiencia del proceso de software en organizaciones en el nivel cinco puede caracterizarse como mejora continua, ya que están luchando por mejorar el rango de la eficiencia de su capacidad a través de mejorar la funcionalidad del proceso en sus proyectos.

En este nivel, la organización y la "instrumentación" del proceso a lo largo de esta pueden proporcionar retroalimentación para mejorarlo. Su principal característica

es un énfasis formal en una mejora continua del proceso basada en las métricas descritas en el nivel anterior, de tal manera que la organización posee un mecanismo que describe como debe ser modificado el proceso.

El foco del nivel cinco, es la mejora continua del proceso. El proceso de software es cambiado para mejorar la calidad, y la zona de control de calidad se mueve. Se establece una nueva línea de funcionalidad y se reduce el desperdicio crónico. Las lecciones aprendidas de mejorar tales procesos son aplicadas a la planeación de procesos futuros. Es cuando el concepto de direccionar causas comunes de variación se pone al frente. Hay un desperdicio crónico, en la forma de retrabajo en cualquier sistema debido a la variación aleatoria. El desperdicio no es aceptado, se organizan esfuerzos para remover el desperdicio resultante en cambiar el sistema, i.e., mejorar el proceso puede cambiar las causas comunes de ineficiencia para prevenir la ocurrencia del desperdicio. [7]

2.3 ESTRUCTURA INTERNA DE LOS NIVELES DE MADUREZ

Con la excepción del nivel uno, cada nivel de madurez está compuesto por diversas áreas de proceso claves. Cada área de proceso esta organizada en cinco secciones llamadas tareas comunes. Las tareas comunes especifican las prácticas claves que, cuando son direccionadas, cumplen las metas de las áreas de proceso.

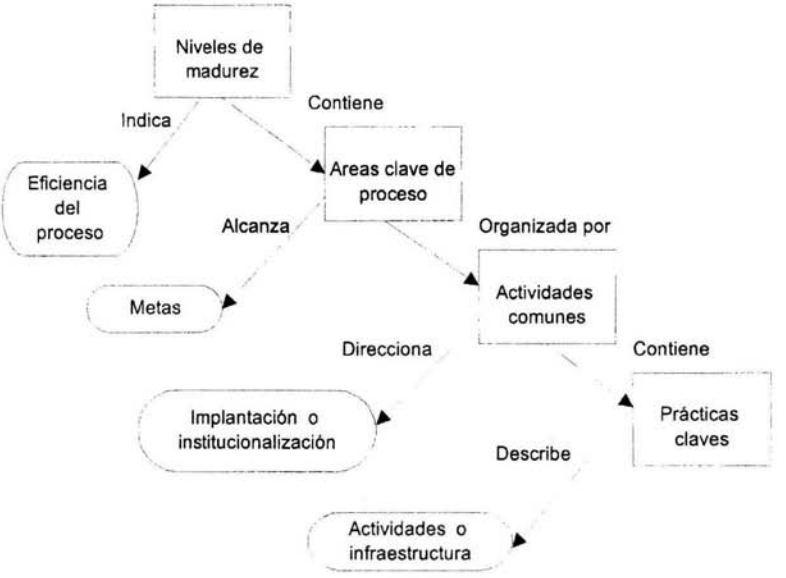
Niveles de madurez

Un nivel de madurez es una plataforma evolutiva bien definida para alcanzar un proceso de software maduro. Cada nivel de madurez indica un nivel de eficiencia del proceso. Por ejemplo, en el nivel dos, la eficiencia del proceso de una organización ha sido elevada de a la medida a disciplinada a través de establecer controles administrativos del proyecto.

Para una referencia sencilla, los cinco niveles del CMM han sido abreviados como inicial, repetible, definido, administrado y optimizable. Estos niveles han sido seleccionados por el SEI debido a que:

- Representan razonablemente fases históricas de mejora evolutiva.
- Brindan pasos de mejora alcanzables en una secuencia razonable.
- Sugieren metas de mejora y medidas de progreso.
- Brindan prioridades inmediatas de mejora una vez que el estatus de la organización es conocido dentro de este esquema.

La siguiente figura muestra los componentes de los niveles de madurez del CMM



Areas claves de proceso (ACP)

Una área clave de proceso contiene las metas que deben ser alcanzadas para mejorar el proceso de software. Se establece que el área esta satisfecha cuando han sido establecidos los procedimientos correspondientes para alcanzar las metas definidas en ella. Una organización de software alcanza un nivel de madurez cuando todas las áreas claves de proceso correspondientes han sido satisfechas. Cada nivel del CMM, excepto el uno, incluye áreas claves de proceso que identifican hacia donde debe enfocarse una organización para llevar el proceso de software a ese nivel. Como las ACP son requerimientos para alcanzar un nivel de madurez, no se define ninguna para el uno.

Cuando una organización lleva a cabo actividades definidas por las ACPs, puede alcanzar metas importantes para mejorar la eficiencia del proceso. Todas incluyen responsabilidades para el proyecto y para la organización.
Cada ACP está dividida en:

- ✓ Metas.
- ✓ Compromiso de ejecución.
- ✓ Capacidad de ejecución.
- ✓ Actividades a ejecutar.
- ✓ Medición y análisis.
- ✓ Verificación de implantación.

Las metas resumen las prácticas más importantes de cada área clave y son usadas para determinar si una organización o proyecto ha cubierto el área clave de proceso. Las metas establecen el alcance y fronteras de cada ACP.

ACTIVIDADES COMUNES

Las prácticas que describen a las áreas clave de proceso están organizadas en actividades comunes, que son atributos que indican si la implantación e institucionalización de un área clave de proceso es: adecuada, repetible y duradera.
Existen cinco actividades comunes:

Compromiso de ejecución.
Describe las acciones que la organización debe tomar con el fin de asegurar que el proceso sea establecido y duradero. Generalmente involucra el establecimiento de políticas organizacionales y liderazgo.

Capacidad de ejecución.
Describe las pre condiciones que deben existir en el proyecto o en la organización para implantar completamente el proceso de software. Generalmente involucra recursos, estructuras organizacionales y capacitación.

Actividades a ejecutar.
Describe las actividades, roles y procedimientos necesarios para implantar una área clave de proceso. Generalmente involucra el establecimiento de planes y procedimientos, ejecución del trabajo, revisión de este y tomar las medidas correctivas necesarias.

Medición y análisis.
Describe las prácticas de medición básicas que son necesarias para determinar el estatus relacionado con el proceso. Estas mediciones son usadas para controlar y mejorar el proceso. Generalmente incluye ejemplos de las medidas que pueden tomarse.

Verificación de la implantación.
Describe los pasos que aseguran que las actividades sean ejecutadas de acuerdo al proceso que ha sido establecido. Generalmente incluye revisiones y auditorias de administradores y equipos de control de calidad.

Las prácticas para las actividades a ejecutar describen que debe ser implantado para establecer la eficiencia del proceso. Las otras prácticas tomadas como un todo, forman la base a través de la cual la organización puede institucionalizar las prácticas descritas en la de actividades a ejecutar.

Prácticas claves

Cada área clave de proceso está descrita en términos de prácticas claves, las cuales describen las actividades y la infraestructura que contribuye mejor a la implantación efectiva y a la institucionalización del área clave de proceso.
Las prácticas claves describen el "que" no el "como" debe hacerse. Se pueden definir prácticas alternativas para alcanzar las metas establecidas en el área clave del proceso. Las prácticas claves deben ser interpretadas racionalmente para juzgar si las metas de las áreas claves son alcanzadas adecuadamente.

Cada práctica consta de un solo enunciado, frecuentemente seguida de una descripción detallada. Estas prácticas claves describen la infraestructura y las actividades que contribuyen a la implantación e institucionalización del área clave de proceso[1].

PUNTOS SOBRESALIENTES DEL CAPÍTULO

- El Departamento de la Defensa de EU, ante los grandes problemas en el desarrollo de software, formó en 1984 el SEI a fin de establecer estándares de excelencia para desarrollos de software y con el apoyo del MITRE, anuncio su esquema de madurez para procesos de software en 1987. El cual está basado en los principios de control de calidad.

- La madurez de proceso es el punto en que este se encuentra explícitamente definido, administrado, medido y controlado.

- El modelo de eficiencia y madurez para software (CMM) da a las organizaciones guías para controlar su desarrollo y mantenimiento de software.

- Un nivel de madurez es una plataforma evolutiva bien definida que sirve de apoyo para alcanzar un proceso de software maduro. Representa de manera razonable las fases actuales de la mejora de las organizaciones de software, así como también una medida de mejora razonable de alcanzar desde un nivel anterior.

- El CMM tiene cinco niveles con las siguientes características:

1. Inicial
El proceso de software está caracterizado por ser a la medida y en algunas ocasiones caótico. Pocas actividades están definidos y el éxito depende de esfuerzos individuales.

2. Repetible
Se establecen las actividades básicas de administración de proyectos para monitorear costos, tiempos y funcionalidad. La disciplina necesaria del proceso es aplicada para repetir éxitos preliminares en proyectos similares.

3. Definido
El proceso de software para las actividades de administración e ingeniería es documentado, estandarizado e integrado dentro de un proceso estándar para la organización. Todos los proyectos usan una versión aprobada y adecuada del proceso estándar de software para desarrollar y dar mantenimiento.

4. Administrado
Se recopilan medidas detalladas del proceso de software y la calidad del producto. El proceso de software y los productos son cuantitativamente entendidos y controlados.

5. Optimizado
La mejora continua de procesos se establece por una retroalimentación cuantitativa y por probar ideas innovadoras y tecnologías.

- Cada nivel de madurez, a partir del nivel dos, está compuesto por diversas áreas de proceso claves.

- Una área clave de proceso contiene las metas que deben ser alcanzadas para mejorar el proceso de software.

- Cada área de proceso esta organizada en cinco secciones llamadas tareas comunes. Las cuales son: Compromiso de ejecución, Capacidad de ejecución, Actividades ejecutadas, Medición y análisis y Verificación de la implantación.

- Las tareas comunes especifican las prácticas claves que, cuando son cubiertas, cumplen las metas de las áreas de proceso.

- Una organización alcanza el nivel de madurez cuando ha cubierto todas las áreas claves de proceso de ese nivel.

III. DESCRIPCIÓN DE LOS COMPONENTES DE CADA NIVEL DEL MODELO DE EFICIENCIA Y MADUREZ CMM

El CMM es una estructura que representa un camino de mejoras recomendadas en una organización de software que quiere mejorar su proceso de desarrollo de software. Esta descripción operacional del CMM está diseñada para apoyar las diferentes formas en que se usará. Hay al menos 4 usos del CMM:

- ✓ Equipos de evaluación usarán el CMM para identificar puntos fuertes y débiles dentro de la organización.
- ✓ Equipos de evaluación usarán el CMM para identificar los riesgos de seleccionar entre diferentes proveedores para ganar negocios y monitorear contratos.
- ✓ Los administradores y el staff técnicos usarán el CMM para entender las actividades necesarias para planear e implantar un programa de mejoras al proceso de software para la organización.
- ✓ Los grupos de mejora del proceso, tales como un GPIS –Grupo de Proceso de Ingeniería de Software-, usarán el CMM como una guía para ayudar a definir y mejorar el proceso de software de la organización.

Debido a los diversos usos del CMM, este debe descomponerse a detalle suficiente para que las recomendaciones del proceso actual puedan ser derivadas de la estructura de los niveles de madurez. Esta descomposición también indica el proceso y su estructura para caracterizar la madurez y eficiencia del proceso de software.

En este capítulo se definirá la estructura de cada uno de los niveles que componen al CMM, describiendo sus procesos claves, las actividades comunes y los resultados de cada uno, así como las relaciones que guardan para establecer su importancia para el logro del resultado.

3.1 AREAS CLAVES DE PROCESO (ACP)

Como ya se había mencionado, una área clave de proceso contiene las metas que deben ser alcanzadas para madurar el proceso. Todas incluyen responsabilidades para el proyecto y la organización.

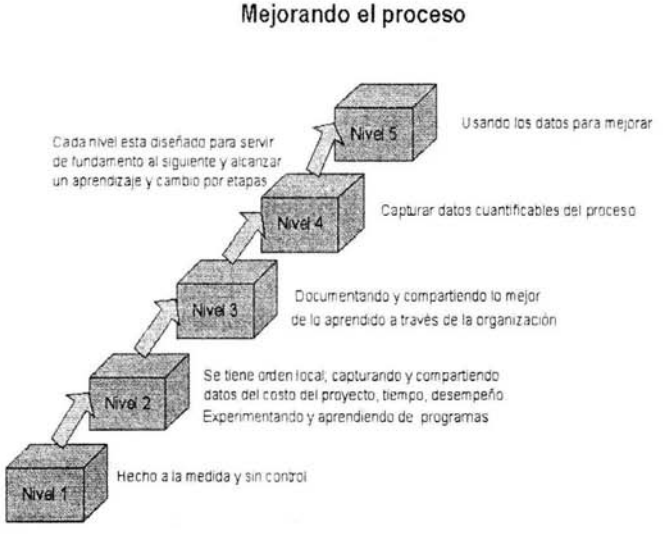
Las áreas claves de proceso para cada nivel son:

Nivel	Objetivo	Áreas claves de proceso
1 Inicial		
2 Repetible	Administración del proceso	Administración de requerimientos Planeación de proyectos de software Seguimiento de proyectos de software Administración de subcontratistas de software Certeza de la calidad del software Administración de la configuración del software
3 Definido	Ingeniería del proceso	Enfoque organizacional del proceso Definición organizacional del proceso Programa de entrenamiento Administración integrada de software Ingeniería de productos de software Coordinación intergrupal Revisiones a fondo
4 Administrado	Calidad del producto y del proceso	Administración cuantitativa Administración de la calidad del software
5 Optimizable	Mejora continua del proceso	Prevención de defectos Administración del cambio de tecnología Administración del proceso de cambios

Actividades comunes.
Las actividades comunes son atributos que indican si la implantación de un área clave de proceso es adecuada, repetible y duradera. Existen cinco actividades comunes:

1. Compromiso de ejecución
2. Eficiencia de ejecución
3. Actividades a ejecutar
4. Medición y análisis
5. Verificación de la implantación

Prácticas claves.
Cada área clave está descrita en términos de prácticas claves: que describen las actividades a realizar para la implantación del área clave de proceso. Las prácticas claves describen el "que" debe hacerse. Cada práctica consta de un solo enunciado, frecuentemente seguido de una descripción mas detallada.



A continuación se definirá la estructura de cada uno de los niveles que componen al CMM, a través del resumen de la descripción de sus áreas claves, de acuerdo a lo expuesto en el libro *'Guidelines for improvement the software process'* publicado por el SEI, donde delinea las metas de cada área y las actividades a seguir para alcanzarlas.

3.2 AREAS CLAVES DE PROCESO PARA EL NIVEL DOS.

En el nivel dos la organización tiene una funcionalidad estable a través de iniciar una administración rigurosa de los compromisos, tiempos y cambios. Esta estabilidad, por lo general involucra un rango de acciones destinadas a obtener control sobre costos y tiempos, tales como organización, administración de proyectos, procesos y tecnología.

Las áreas claves de proceso para cubrir en este nivel son:

Administración de requerimientos.
Su propósito es establecer un entendimiento común entre el cliente y el equipo de trabajo sobre los requerimientos que serán resueltos a través del proyecto de software. Implica establecer y mantener un acuerdo con el cliente sobre sus requerimientos en el proyecto de software.

Metas.

- Controlar los requerimientos de sistemas para establecer la base del uso de ingeniería de software y prácticas básicas administrativas.
- Hacer consistentes los planes, productos y actividades de software con los requerimientos del sistema.

Las prácticas claves a llevar a cabo son:

- Revisar los requerimientos antes de que sean incorporados al proyecto.
- Usar los requerimientos como base para los planes, productos de trabajo y actividades.
- Revisar e incorporar los cambios a los requerimientos dentro del proyecto.

Planeación del proyecto de software
El propósito de la planeación del proyecto de software es establecer planes razonables para ejecutar ingeniería de software y administración de proyectos.

Involucra el desarrollo de estimados para el trabajo a ser ejecutado, estableciendo los compromisos necesarios y la definición del plan de trabajo. Inicia con una definición del trabajo a ser desarrollado, las constantes y metas que definen y limitan el proyecto de software. Incluye los pasos necesarios para estimar el tamaño de productos de software y recursos, determinar tiempos, identificar y evaluar riesgos y negociar compromisos. Este plan da las bases para ejecutar y administrar las actividades del proyecto y direccionar los compromisos de acuerdo a los recursos, restricciones y capacidades del proyecto.

Metas

- Documentar y usar los estimados para planear y dar seguimiento al proyecto.
- Planear y documentar las actividades del proyecto y sus compromisos.

- Acordar entre los grupos afectados los compromisos relacionados con el proyecto.

Las prácticas claves a llevar a cabo son:

- Participación del grupo de ingeniería de software en el diseño del proyecto.
- Revisar con la administración los compromisos hechos para el proyecto por grupos externos a la organización de acuerdo a un procedimiento documentado.
- Identificar o definir un ciclo de vida del software con etapas predefinidas de tamaño manejable.
- Identificar los productos de trabajo que son necesarios para establecer y mantener el control del proyecto – estándares de desarrollo, estrategias de comunicación, esquemas generales de transferencia de información, etc-.
- Calcular los estimados del tamaño de los productos de acuerdo a un procedimiento documentado.
- Calcular los estimados de esfuerzo y costo de acuerdo a un proceso documentado. Por ejemplo, puntos por función.
- Estimar los tiempos del proyecto de acuerdo a un proceso documentado. Por ejemplo, puntos por función.
- Identificar, medir y documentar los riesgos del proyecto asociados a costos, tiempos, recursos y aspectos técnicos.
- Desarrollar el plan del proyecto de acuerdo a un procedimiento documentado.
- Documentar el plan para el proyecto.
- Preparan las herramientas de soporte y de ingeniería de software.
- Registrar los datos de la planeación.
- Participación del grupo de ingeniería de software con otros grupos a través de la vida del proyecto.

Seguimiento del proyecto.

Su propósito es dar una visibilidad adecuada dentro del proceso actual, de tal manera que la administración puede tomar las acciones necesarias cuando la funcionalidad del proyecto se desvíe significativamente de los planes.

Esto implica seguimiento y revisión de los resultados contra los estimados, compromisos y planes documentados y ajustar estos con base a los resultados reales.

Metas

- Dar seguimiento a la funcionalidad y resultados actuales contra el plan del proyecto.
- Tomar acciones correctivas cuando los resultados y la funcionalidad actual se desvían significativamente de los planes de software.
- Acordar cambios a los compromisos por los grupos afectados.

Las prácticas claves a llevar a cabo son:

- Usar un plan de desarrollo documentado para dar seguimiento a las actividades de software y comunicar su estatus.
- Documentar el plan del proyecto de acuerdo a un procedimiento documentado.
- Revisar con la gerencia los compromisos del proyecto y los cambios realizados por grupos externos a la organización de acuerdo a un proceso documentado.
- Comunicar los cambios aprobados.
- Revisar el tamaño de los productos de software (o tamaño de los cambios a los productos) y llevar a cabo las acciones correctivas necesarias.
- Revisar los esfuerzos y costos del proyecto y llevar a cabo las acciones correctivas necesarias.
- Revisar los tiempos del proyecto y llevar a cabo las acciones correctivas necesarias.
- Revisar las actividades técnicas de ingeniería de software y llevar a cabo las acciones correctivas necesarias.
- Revisar los riesgos asociados con costos, recursos, tiempos y aspectos técnicos del proyecto.
- Registrar los nuevos datos de medición y replaneamiento del proyecto.
- Revisar de manera periódica el seguimiento de los progresos técnicos, planes y desempeño contra el plan inicial.
- Llevar a cabo revisiones formales para encaminarse a los resultados del proyecto de acuerdo a un procedimiento documentado.

Administración de contratantes secundarios de software

Su objetivo es seleccionar contratistas secundarios de software adecuados y administrarlos de manera efectiva. Involucra seleccionar al contratista, establecer compromisos y monitorear su funcionalidad y resultados.

Metas

- Seleccionar a los contratistas secundarios por parte del principal contratista.
- Acordar los compromisos entre el contratista principal y los secundarios.
- Establecer comunicación constante entre el contratista principal los secundarios.
- Monitorear el desempeño y resultados de los contratistas secundarios por parte del principal en base a sus compromisos.

Las prácticas claves a llevar a cabo son:

- Definir y planear el trabajo a ser subcontratado de acuerdo a un procedimiento documentado.
- Seleccionar al contratista de software basándose en la evaluación de las habilidades de los participantes para la ejecución del trabajo, de acuerdo a un procedimiento documentado.
- Usar los acuerdos contractuales como base para administrar al contratista.

- Revisar y aprobar un plan documentado de desarrollo del contratista.
- Usar el plan aprobado y documentado del subcontratista para revisar las actividades y estatus del mismo.
- Resolver los cambios a la propuesta del contratista en cuanto a términos y condiciones, así como a otros compromisos de acuerdo a un procedimiento documentado.
- Llevar a cabo revisiones periódicas de estatus con la administración del contratista.
- Llevar a cabo revisiones e intercambios técnicos periódicos con el contratista de software.
- Llevar a cabo revisiones formales del cumplimiento de los procesos de ingeniería de software del contratista, sobre objetivos específicos de acuerdo a un procedimiento documentado.
- Monitorear la calidad de las actividades del contratista de acuerdo a un procedimiento documentado.
- Monitorear las actividades del contratista para la configuración del software de acuerdo a un procedimiento documentado.
- Llevar a cabo pruebas de aceptación como parte de los entregables de productos de software del contratista de acuerdo a un procedimiento documentado.
- Evaluar el desempeño del contratista sobre bases periódicas y revisar esta evaluación con el contratista.

Aseguramiento de la calidad del software

Su propósito es proporcionar a la administración una visibilidad apropiada dentro del procesos usado por el proyecto y de los productos construidos. Involucra la revisión de los productos y actividades para verificar que cumplen con los procedimientos y estándares y brindan los resultados de estas revisiones.

El grupo del aseguramiento de la calidad revisa las actividades del proyecto y audita los productos de software a través de la vida del proyecto y brinda la visibilidad de si el proyecto está cumpliendo con los planes, estándares y procedimientos establecidos.

Metas

- Planear las actividades de aseguramiento de la calidad.
- Verificar de manera objetiva el cumplimiento de los estándares, procedimientos y requerimientos por parte de los productos de software.
- Informar a los grupos afectados de las actividades y resultados del grupo de medición de la calidad.
- Direccionar los problemas que no puedan ser resueltos dentro del proyecto al su gerente.

Las prácticas claves a llevar a cabo son:

- Preparar un plan de aseguramiento de la calidad de software para el proyecto de acuerdo a un procedimiento documentado.

- Preparar y revisar el plan, estándares y procedimientos del proyecto por parte del grupo de aseguramiento de calidad de software.
- Llevar a cabo las actividades del grupo de aseguramiento de la calidad de acuerdo al plan anterior.
- Revisar las actividades de ingeniería de software por parte del grupo de aseguramiento de la calidad para revisar el cumplimiento de los estándares definidos.
- Llevar a cabo auditorias de productos de software designados por parte del grupo de aseguramiento de calidad para revisar el cumplimiento de los estándares definidos.
- Reportar periódicamente los resultados de sus actividades al grupo de ingeniería de software.
- Documentar y manejar las desviaciones identificadas en las actividades de software y en los productos de trabajo de acuerdo a un proceso documentado.
- Llevar a cabo revisiones con el cliente de las actividades y hallazgos por parte del grupo de aseguramiento de calidad.

Administración de la configuración del software
Su objetivo es establecer y mantener la integridad de los productos del proyecto a través de su ciclo de vida. Involucra identificar la configuración del software en intervalos de tiempo determinados así como controlar sistemáticamente los cambios a la configuración y mantener la integridad y seguimiento de la configuración a través del ciclo de vida del proyecto.

Metas
- Planear las actividades de la configuración del software.
- Identificar y controlar productos de software seleccionados.
- Controlar los cambios a los productos de software.
- Informar a los grupos afectados del estatus y contenido de la baseline[1] del software.

Las prácticas claves a llevar a cabo son:
- Preparar un plan de la configuración del software para el proyecto de acuerdo a un procedimiento documentado.
- Usar un plan documentado y aprobado como base para llevar a cabo las actividades de configuración del software.
- Establecer un sistema de librerías de administración de configuración como un repositorio para los baselines del software.
- Identificar los productos de software necesarios para la administración de la configuración.

[1] El baseline para el software debe ser entendido como la especificación revisada de manera formal que sirve como base para desarrollos posteriores y solo puede ser cambiado a través de un procedimiento formal de control de cambios.

- Iniciar, registrar, revisar, aprobar y seguir los cambios requeridos y los reportes de problemas para todos los componentes /unidades a través de un proceso documentado.
- Controlar los cambios a los baselines de acuerdo a un proceso documentado.
- Crear los productos de las librerías de baselines de software y controlar su liberación de acuerdo a un proceso documentado.
- Registrar los estatus de los componentes /unidades de configuración de acuerdo a un procedimiento documentado.
- Desarrollar reportes estándar para documentar las actividades de configuración del software y los contenidos de los baselines de software y ponerlos a disposición para ser modificados por los grupos.
- Llevar a cabo auditorias a los baselines del software de acuerdo a un proceso documentado.

3.3 AREAS CLAVES DE PROCESO PARA EL NIVEL TRES

En este nivel, la organización ha establecido un proceso estándar adaptado para cumplir las necesidades de cada proyecto. Esto brinda bases para manejar las contingencias y permite una mejora ordenada y facilita la introducción de tecnologías de apoyo.

Las áreas claves de proceso para cubrir en este nivel son:

Enfoque organizacional del proceso.
Su propósito es establecer la responsabilidad organizacional para las actividades del proceso de software que mejoren la eficiencia de todo el proceso de software. Involucra desarrollar y mantener un entendimiento de los procesos de software en la organización y los proyectos y coordinar las actividades de medir, desarrollar, mantener y mejorar.

Metas
- Coordinar el desarrollo y mejora de las actividades del proceso de software a través de toda la organización.
- Identificar las fuerzas y debilidades del proceso usado con respecto al proceso estándar.
- Planear actividades de desarrollo y mejora de procesos.

Las prácticas claves a llevar a cabo son:
- Evaluar el proceso de software periódicamente y desarrollar planes de acción para cubrir los hallazgos de las evaluaciones.
- Desarrollar y mantener un plan para el proceso de desarrollo de software y sus actividades de mejora.
- Coordinar a nivel organizacional las actividades para desarrollar y mejorar su proceso de software.

- Coordinar el uso de una base de datos para el proceso de software de la organización.
- Monitorear, evaluar y cuando sea apropiado, transferir a otra parte de la organización, procesos, métodos y herramientas de uso limitado.
- Coordinar la capacitación en los procesos de software y el proyecto a través de la organización.
- Informar a los grupos involucrados en la implementación del proceso de software sobre las actividades de desarrollo y mejora para la organización y el proyecto.

Definición del proceso organizacional.
Su propósito es desarrollar y mantener un conjunto usable de puntos que beneficien la funcionalidad del proceso a través de los proyectos y den base a beneficios acumulativos y a largo plazo para la organización.

Involucra el desarrollo y mantenimiento de estándares y puntos relativos al proceso, tales como descripciones del ciclo de vida, guías y criterios de adecuación, la base de datos del proceso de software de la organización y una librería de la documentación relacionada con el proceso.

Metas
- Desarrollar y mantener un proceso estándar de software para la organización.
- Recopilar, revisar y poner disponible la información relacionada al uso del proceso estándar de software.

Las prácticas claves a llevar a cabo son:
- Desarrollar y mantener el proceso de software estándar para la organización de acuerdo a un procedimiento documentado.
- Documentar el proceso de software estándar de la organización para establecer estándares organizacionales.
- Documentar y mantener descripciones de los ciclos de vida del software aprobados para ser usados por los proyectos.
- Desarrollar y mantener guías y criterios para el ajuste del proceso de software organizacional al proyecto.
- Establecer y mantener la base de datos del proceso de software organizacional.
- Establecer y mantener una librería con la documentación relativa al proceso de software.

Programa de entrenamiento.
Su objetivo es desarrollar las habilidades y conocimientos de las personas de tal manera que puedan ejecutar sus actividades de manera eficiente. Involucra identificar la capacitación necesaria para la organización, proyectos, individuos y después desarrollar un programa de entrenamiento que satisfaga estas necesidades.

Cada proyecto de software identifica sus necesidades de entrenamiento y determina como serán obtenidas estas habilidades. Algunas de ellas son obtenidas de manera informal (asesoría informal, entrenamiento en el trabajo) mientras que otras necesitan vehículos de entrenamiento formales.

Metas

- Planear las actividades de capacitación.
- Proporcionar el entrenamiento para desarrollar las habilidades y conocimientos necesarios para llevar a cabo la administración del software y los roles técnicos.
- Dar la capacitación necesaria al personal en el grupo de ingeniería de software y los grupos relacionados para ejecutar sus tareas.

Las prácticas claves a llevar a cabo son:

- Desarrollar y mantener el plan de capacitación para cada proyecto de acuerdo a sus necesidades.
- Desarrollar y revisar el plan de la organización para capacitación a través de un procedimiento documentado.
- Llevar a cabo la capacitación para la organización de acuerdo con el plan de capacitación organizacional.
- Desarrollar y mantener los cursos de capacitación preparados a un nivel organizacional de acuerdo a los estándares organizacionales.
- Establecer y usar un proceso alterno para la capacitación requerida a fin de determinar si las personas ya tienen el conocimiento y habilidades necesarias para llevar a cabo sus roles.
- Mantener los registros de capacitación.

Administración de la integración del software

Su objetivo es integrar las actividades de ingeniería de software y de administración dentro de un proceso coherente y definido que es adecuado al proceso estándar de la organización.

Involucra desarrollar el proceso definido de software y administrar el proyecto usando un proceso definido.

Metas

- Adecuar el proceso de software estándar de la organización al proceso del proyecto.
- Planear y administrar el proyecto de acuerdo al proceso estándar definido.

Las prácticas claves a llevar a cabo son:

- Desarrollar el proceso de software para el proyecto en base al ajuste del proceso estándar de la organización, de acuerdo a procedimientos documentados.
- Revisar el proceso de software de cada proyecto de acuerdo a un procedimiento documentado.

- Desarrollar y revisar el plan del proyecto que describe el proceso de software del proyecto, de acuerdo a un procedimiento documentado.
- Administrar el proyecto de acuerdo con el proceso predefinido.
- Usar la base de datos de la organización para el proceso de a fin de planear y estimar el software.
- Administrar el tamaño de los productos de software (o cambio en sus tamaños) de acuerdo a un procedimiento documentado.
- Administrar los esfuerzos y costos del proyecto de acuerdo a un procedimiento documentado.
- Administrar los recursos computacionales críticos del proyecto de acuerdo a un procedimiento documentado.
- Administrar las dependencias y rutas críticas de los tiempos del proyecto de acuerdo a un procedimiento documentado.
- Identificar, evaluar, documentar y administrar los riesgos del proyecto de acuerdo a un procedimiento documentado.
- Llevar a cabo revisiones del proyecto de manera periódica para determinar las acciones necesarias para ajustar el desempeño del proyecto y sus resultados a las necesidades del negocio, cliente, usuario final, de manera apropiada.

Ingeniería de productos de software
Su objetivo es ejecutar de manera consistente un proceso bien definido de ingeniería de software que integre todas las actividades para producir productos de software correctos y consistentes.

Involucra ejecutar tareas de ingeniería para construir y mantener el software usando el proceso definido y sus métodos y herramientas. Estas tareas incluyen analizar los requerimientos del sistema, desarrollar los requerimientos y arquitectura, diseñar el software, implementarlo en el código, integrar los componentes y probar el software para verificar que satisfaga los requerimientos definidos.

Metas
- Definir, integrar y ejecutar las tareas de ingeniería de software para producir el software.
- Los productos de trabajo del software son consistentes entre ellos.

Las prácticas claves a llevar a cabo son:
- Integrar métodos apropiados de ingeniería de software y herramientas al proceso definido de software para el proyecto.
- Desarrollar, mantener, documentar y verificar los requerimientos de software a través de su análisis sistemático de acuerdo al proceso de software del proyecto.
- Desarrollar, mantener, documentar y verificar el diseño de software de acuerdo al proceso definido para el proyecto, a fin de incluir los requerimientos del proyecto y formar la infraestructura para codificar.

- Desarrollar, mantener, documentar y verificar la codificación del software de acuerdo al proceso definido para el proyecto, a fin de implementar los requerimientos y el diseño del software.
- Llevar a cabo pruebas del software de acuerdo al proceso definido para el proyecto.
- Planear y ejecutar las pruebas de aceptación e integración para demostrar que el software satisface los requerimientos.
- Desarrollar la documentación que será usada para operar y dar mantenimiento al software de acuerdo al proceso definido para el proyecto.
- Recopilar los datos sobre defectos identificados en revisiones y pruebas y son analizados de acuerdo al proceso definido para el proyecto.
- Mantener la consistencia a través de los productos de trabajo de software, incluyendo planes, descripciones de procesos, requerimientos de software y funcionalidad, diseño, codificación, planes de prueba y procedimientos de prueba.

Coordinación intergrupal.
Su propósito es establecer los medios para que el grupo de ingeniería de software participe activamente con los otros grupos de ingeniería, de tal manera que el proyecto satisfaga correctamente las necesidades del cliente.

Involucra la participación del grupo de ingeniería de software con otros grupos para enfocarse a los requerimientos, objetivos y planes que se convierten en la base de las actividades de ingeniería.

Metas.
- Acordar los requerimientos del cliente por todos los grupos afectados.
- Negociar los compromisos entre los grupos de ingeniería y los grupos afectados.
- Identificación, seguimiento y solución de los problemas intergrupales por parte de los grupos de ingeniería.

Las prácticas claves a llevar a cabo son:
- Establecer los requerimientos del sistema por parte del grupo de ingeniería de software con los clientes y/o usuarios finales.
- Monitoreo y coordinación de actividades técnicas de representantes del grupo de ingeniería de software.
- Usar un plan documentado para comunicar los compromisos intergrupales y coordinan y monitorear el trabajo ejecutado.
- Identificar, negociar y monitorear las dependencias críticas entre grupos de ingeniería de acuerdo a un procedimiento documentado.
- Revisar los productos de trabajo producidos como entrada de otros grupos de ingeniería por parte de representantes de los grupos receptores para asegurarse que los productos cubren sus necesidades.

- Manejar de acuerdo a un proceso documentado los asuntos intergrupales no resueltos por los representantes de los grupos del ingeniería del proyecto.
- Llevar a cabo revisiones e intercambios técnicos de manera periódica por parte de representantes de los grupos de ingeniería del proyecto.

Revisiones a fondo.
Su propósito es remover defectos de los productos temprana y eficientemente. Así como desarrollar un mejor entendimiento de los productos de software y de los defectos que pueden ser prevenidos.

Involucra una revisión metódica de los productos de software para identificar defectos y detectar áreas que necesiten cambios.

Metas
- Planear las revisiones a fondo.
- Identificar y eliminar los defectos en los productos de software.

Las prácticas claves a llevar a cabo son:
- Planear las revisiones y documentar estos planes.
- Ejecutar las revisiones de acuerdo a un procedimiento documentado.
- Registrar los datos de los resultados de las revisiones.

3.4 AREAS CLAVES DE PROCESO PARA EL NIVEL CUATRO

En el nivel cuatro la organización ha iniciado mediciones y análisis comprensivos del proceso. El conocimiento preciso resultante del proceso brinda las bases para una mejora continua en la calidad y productividad.

Las áreas claves de proceso para cubrir en este nivel son:

Administración cuantitativa del proceso.
Su objetivo es controlar la funcionalidad del proceso de software cuantitativamente. Esta funcionalidad presenta los resultados actuales alcanzados de seguir el proceso de software.

Involucra establecer metas de funcionalidad para el proceso, el cual es descrito en la administración de la integración del software. Tomar medidas, analizarlas y hacer ajustes para mantener el desempeño del proyecto dentro de límites razonables.

Metas
- Planear las actividades de administración cuantitativa.
- Controlar de manera cuantitativa la funcionalidad del proceso de software.
- Dar a conocer la eficiencia del proceso estándar de la organización en términos cuantitativos.

Las prácticas claves a llevar a cabo son:

- Desarrollar el plan del proyecto para la administración cuantitativa del proceso de acuerdo a un procedimiento documentado.
- Ejecutar las actividades de administración cuantitativa del proceso de acuerdo al plan desarrollado.
- Determinar la estrategia para la recolección de datos y el análisis cuantitativo a ser llevado a cabo, basándose en el proceso definido de software.
- Recopilar los datos de medición usados para controlar cuantitativamente el proceso de acuerdo a un procedimiento documentado.
- Analizar y someter a un control cuantitativo el proceso de software definido para el proyecto de acuerdo a un procedimiento documentado.
- Preparar y distribuir los resultados que documentan las actividades de administración cuantitativa del proyecto.
- Establecer y mantener el baseline de la eficiencia del proceso de software de la organización de acuerdo a un procedimiento documentado.

Administración de la calidad del software

Su propósito es desarrollar un entendimiento cuantitativo de la calidad de los productos del proyecto de software y alcanzar metas específicas de calidad.

Involucra definir metas de calidad para los productos, establecer planes para alcanzar estas, monitorear y ajustar los planes de software, los productos de trabajo, actividades y metas de calidad para satisfacer las necesidades del cliente o usuario final.

Metas

- Planear las actividades de administración de calidad del proyecto de software.
- Definir metas medibles para la calidad de los productos de software así como sus prioridades.
- Cuantificar y administrar el progreso actual contra las metas de calidad para los productos de software.

Las prácticas claves a llevar a cabo son:

- Desarrollar y mantener el plan de la calidad del proyecto de acuerdo a un procedimiento documentado.
- El plan de la calidad del proyecto de software es la base para las actividades de administración de la calidad del software.
- Definir, monitorear y revisar las metas cuantitativas de calidad para los productos de software a través del ciclo de vida definido.
- Medir, analizar y comparar la calidad de los productos de software del proyecto con las metas cuantitativas de los productos.
- Definir apropiadamente para aquellos contratistas involucrados las metas cuantitativas de calidad del software para sus productos.

3.5 AREAS CLAVES DE PROCESO PARA EL NIVEL CINCO

En el nivel cinco los datos sobre el proceso de software son usados para evaluar la efectividad del proceso y hacer ajustes regulares. Esto brinda un fundamento para una mejora planeada y ordenada de la productividad.

Las áreas claves de proceso para cubrir en este nivel son:

Prevención de defectos.
Su propósito es identificar las causas de defectos y prevenir que vuelvan a ocurrir. Involucra analizar los defectos encontrados y tomar acciones específicas para prevenir su ocurrencia en un futuro.

Metas

- Planear las actividades de prevención de defectos.
- Investigar e identificar las causas comunes de defectos.
- Prioritizar y eliminar las causas comunes de defecto.

Las prácticas claves a llevar a cabo son:

- Desarrollar y mantener el plan para las actividades de prevención de defectos para el proyecto.
- Reunir a los miembros del equipo involucrado en el inicio de una tarea para preparar las actividades relacionadas con la prevención de defectos.
- Llevar a cabo juntas para el análisis de causas de acuerdo a un procedimiento documentado.
- Cada uno de los equipos asignados para coordinar actividades de prevención de defectos se reúnen periódicamente para revisar y coordinar la implementación de acciones derivadas de las juntas de análisis de causas.
- Documentar y monitorear los datos de la prevención de defectos entre los equipos que coordinan estas actividades.
- Incorporar revisiones al proceso de software de la organización derivadas de las acciones de prevención de defectos de acuerdo a un procedimiento documentado.
- Incorporar revisiones al proceso del proyecto derivadas de las acciones de prevención de defectos de acuerdo a un procedimiento documentado.
- Los miembros del grupo de ingeniería de software y de grupos relacionados con software reciben retroalimentación sobre el estatus y resultados de las actividades de prevención de defectos de manera periódica.

Administración de los cambios tecnológicos.
Su propósito es identificar nuevas tecnologías y promocionarlas en la organización ordenadamente. Involucra identificar, seleccionar y evaluar nuevas tecnologías e incorporar las adecuadas a la organización. Su objetivo es mejorar la calidad del

software, incrementar la productividad y decrementar el ciclo de vida del desarrollo de productos.

Metas

- Planear la incorporación de cambios tecnológicos.
- Evaluar nuevas tecnologías para determinar su efecto en la calidad y la productividad.
- Incluir las nuevas tecnologías apropiadas a la organización en la práctica normal a lo largo de la organización.

Las prácticas claves a llevar a cabo son:

- Desarrollar y mantener un plan para la administración de cambio tecnológico dentro de la organización.
- Identificar áreas de cambio por parte del grupo responsable en la organización de los cambios tecnológicos de acuerdo a los proyectos evaluados.
- Informar a los administradores de software y el personal técnico de las nuevas tecnologías.
- Analizar de manera sistemática el proceso estándar de software de la organización para identificar áreas que necesitan o podrían beneficiarse de nueva tecnología.
- Seleccionar y adquirir tecnologías para la organización y los proyectos de software de acuerdo a un procedimiento documentado.
- Llevar a cabo pilotos de mejoras tecnológicas antes de que una nueva tecnología sea introducida a las prácticas normales.
- Incorporar nuevas tecnologías al proceso de software de la organización de acuerdo a un proceso documentado.
- Incorporar las nuevas tecnologías al proyecto de acuerdo a un proceso documentado.

Administración de cambios del proceso.

Su propósito es mejorar de manera continua los procesos de software usados en la organización con la intención de mejorar la calidad del software, incrementar la productividad y decrementar el ciclo de vida del desarrollo de software. Involucra la definición de metas para mejorar el proceso e identificar, evaluar e implementar mejoras a los procesos de software de la organización.

Metas

- Planear las actividades de mejora continua al proceso.
- Participar en las actividades de mejora de los procesos de software de la organización.
- Mejorar el proceso de software de la organización y/o el proyecto de manera continua.

Las prácticas claves a llevar a cabo son:

- Establecer un programa para apoyar la mejora del proceso de la organización.
- Coordinación del grupo responsable de las actividades de mejora del proceso de software de la organización de estas actividades.
- Desarrollar y mantener un plan de mejora del proceso de acuerdo a un procedimiento documentado dentro de la organización.
- Llevar a cabo las actividades de mejora al proceso de acuerdo al plan.
- Manejar las metas de mejora del proceso de acuerdo a un procedimiento documentado.
- Participación activa de los miembros de la organización en equipos para desarrollar mejoras al proceso de software para determinadas áreas de este.
- Instalar las mejoras del proceso en pilotos para determinar sus beneficios y efectividad antes de ser introducidas a las prácticas normales.
- Cuando se ha tomado la decisión de incluir la mejora del proceso dentro de las prácticas normales, su implementación se lleva a cabo de acuerdo a un procedimiento documentado.
- Registrar las actividades de mejora al proceso de software.
- Dar retroalimentación a los administradores de software y el personal técnico sobre el estatus y resultados de las actividades de mejora al proceso de software.

PUNTOS SOBRESALIENTES DEL CAPITULO

En el nivel dos se inicia una administración rigurosa de los compromisos, tiempos y cambios.

Sus áreas claves son:

- ✓ Administración de requerimientos
- ✓ Planeación de proyectos de software
- ✓ Seguimiento de proyectos de software
- ✓ Administración de subcontratistas de software
- ✓ Certeza de la calidad del software
- ✓ Administración de la configuración del software

- En el nivel tres se ha establecido un proceso estándar adaptado para cumplir las necesidades de cada proyecto.

Sus áreas claves son:

- ✓ Enfoque organizacional del proceso
- ✓ Definición organizacional del proceso
- ✓ Programa de entrenamiento
- ✓ Administración integrada de software
- ✓ Ingeniería de productos de software
- ✓ Coordinación intergrupal
- ✓ Revisiones a fondo

- En el nivel cuatro, se han iniciado medidas y análisis comprensivo del proceso.

Sus áreas claves son

- ✓ Administración cuantitativa
- ✓ Administración de la calidad del software

- En el nivel cinco se usan datos sobre el proceso de software para evaluar la efectividad del mismo y hacer ajustes regulares.

Sus áreas claves son

- ✓ Prevención de defectos
- ✓ Administración del cambio de tecnología
- ✓ Administración del proceso de cambios

IV. VENTAJAS, DESVENTAJAS Y RECOMENDACIONES SOBRE EL MODELO DE EFICIENCIA Y MADUREZ (CMM)

El CMM permite una visión global de lo que significa la mejora del proceso de software para una organización, para lograr esto, se basa en un lenguaje común y en la definición de un conjunto de prioridades para atacar los problemas. Es un conjunto de procesos y prácticas desarrolladas en colaboración con opiniones de profesionales que lo revisaron durante su desarrollo.

Las mejoras basadas en un modelo conllevan sus riesgos, "todos los modelos son equivocados aunque algunos son útiles", los modelos deben ser una simplificación del mundo real que representan, el CMM no es una descripción exhaustiva del proceso de software y por lo tanto no toca factores fuera del proceso, tales como la gente o la tecnología, que también afectan el resultado de un proceso de software.

El CMM solo brinda una estructura conceptual para mejorar la administración y el desarrollo de productos de software de una manera disciplinada y consistente. No garantiza que los productos de software se construyan de manera exitosa o que todos los problemas relacionados con su construcción serán resueltos adecuadamente. Es solo una herramienta para apoyar a las organizaciones a mejorar su proceso de desarrollo software y su propósito es describir prácticas adecuadas de ingeniería y administración.

A continuación se brindan puntos de vista sobre las ventajas y desventajas del modelo, con el fin de obtener el mejor provecho que se pueda del mismo.

4.1 VENTAJAS Y DESVENTAJAS GENERALES

Ventajas generales
El CMM tiene tres objetivos principales:

- Ayudar a construir un entendimiento sobre el proceso de software a través de describir las prácticas que componen un nivel de madurez.

- Brindar bases consistentes para generar mediciones del proceso, definiendo escalas que permiten a la organización medir la eficiencia del suyo.

- Servir como una base para la mejora del proceso. Brindando una guía para desarrollar una ruta de donde está la organización y hacia donde quiere ir, pero advierte que no hay atajos, así que toma tiempo establecer una cultura y poder cumplir los compromisos.

Las ventajas de estos objetivos son:

- Permite a las organizaciones evaluar, identificar y mejorar estrategias para el desarrollo de software en todas sus fases, desde la planeación, ingeniería, desarrollo y mantenimiento.

- Sus mediciones son efectivas y eficientes y permite a los participantes obtener control sobre sus procesos de software, así como también una excelencia administrativa.

- Es usado como guía para mejorar los procesos y obtener beneficios en la productividad, la detección temprana de defectos, la calidad del producto y la respuesta a presiones del mercado.

- Reducción en los costos de capacitación y medición.

- Una visión de mejora integrada para todos los elementos de una organización.

- Es un medio para representar información de un proceso

- Forma una cultura de mejora, pues se incluye a todo el personal al definir actividades. El cambio involucra no solo a los desarrolladores, sino también a los líderes y usuarios, ya que el control es lo principal, se debe primero controlar a los usuarios. Los usuarios necesitan ser entrenados para ser pro activos, ya que si todo es una emergencia, esto seguramente es un área de mejora.

- Desarrolla un vocabulario común dentro de la organización para el entendimiento de actividades esenciales, las cuales deben ser ejecutadas para una exitosa ingeniería de sistemas. Permite a la organización hacer evaluaciones rápidas del valor agregado de las mejoras propuestas a través de medir los resultados de la actividad contra el modelo.

Generalmente, las compañías de software reportan beneficios derivados de adoptar modelos de mejora, los beneficios comunes son un cambio cultural positivo, un incremento a la productividad, mejor comunicación y mayor satisfacción del cliente. Estos beneficios generalmente van acompañados de grandes cifras.

Los resultados publicados de algunas compañías de software que han adoptado el CMM, son impresionantes:

Nivel	Costo de desarrollo	Tiempo de desarrollo (meses)	Calidad del producto (defectos * 1000 líneas de código)	Productividad 1000 líneas de código	Productividad ($ por 1000 líneas de código)
1	33	40	9	1	66
2	15	32	3	3	30
3	7	32	1	5	14
4	3	19	0.3	8	6
5	1	16	0.1	12	2

No importando cuanto se haya recuperado de la inversión, se tiene un acuerdo sobre los siguientes beneficios derivados del CMM

- Mejora de la comunicación interna.
- Disminuye el trabajo.
- Disminuye de problemas de integración.
- Disminuye el rango de errores, pese al crecimiento en tamaño de los productos.
- Disminuye los costos de pruebas.
- Los desarrollos que no sobrepasan presupuestos ni tiempos de entrega.
- Mejora la moral entre los empleados, menos presión en el equipo de trabajo y menor tiempo extra.
- Respeto de otras organizaciones y más confianza de los clientes, debido a la puntualidad, eficiencia y documentación de los entregables.

Las evaluaciones de este modelo ayudan a determinar las oportunidades de mejora del proceso, así como también el nivel de madurez de la organización. Los niveles de madurez pueden ser usados como herramientas de mercadeo o como pre requisitos para contratos. Una evaluación del CMM incluye una revisión de políticas, procedimientos y guías organizacionales. Un baseline del proceso actual se desarrolla basado en la revisión de la documentación y entrevistas al personal apropiado del proyecto.

Filosofía. *Controlar el procedimiento de desarrollo te ayudará a controlar el producto. Nuevas técnicas no mejorarán la productividad. El problema es la administración. Enfócate a ella.*

El CMM no impone como hacer las cosas, pero si que cosas hay que hacer, por ejemplo, da por hecho que se tienen estándares, no importa cuales, pero deben de tenerse, porque al seguir estándares, se instituye una estructura básica dentro de la organización.

Para organizaciones con muy poca o ninguna cultura de ingeniería de sistemas, el CMM puede ser usado para establecer un vocabulario común y un entendimiento de ingeniería de sistemas.

Para organizaciones con procesos de ingeniería de software definidos, el CMM puede ser usado para evaluar las fuerzas y debilidades del proceso existente, los resultados de estas evaluaciones pueden ser usadas como la base de la mejora del proceso dentro de la organización.

Para organizaciones ya involucradas en el ciclo de mejora del proceso, el CMM puede ser usado como una métrica periódica para evaluar la efectividad de las mejoras, así como también como una herramienta para vincular estos esfuerzos a las metas estratégicas de la organización.

Las lecciones aprendidas mientras se ejecutan las mediciones, deben ser recopiladas para mejorar las futuras mediciones, estas lecciones caen en tres categorías: capacitación, selección del personal para llevarlas a cabo y técnicas de recopilación de datos.

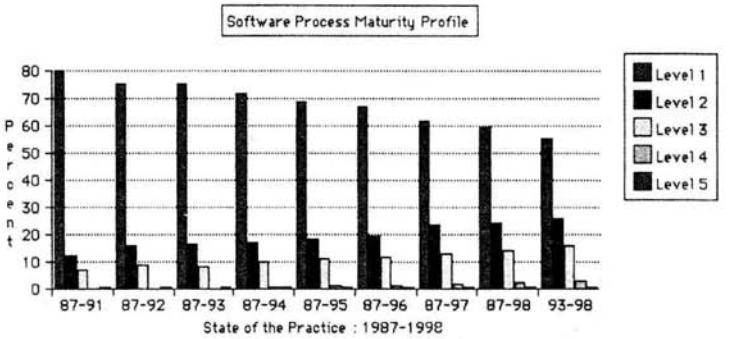
El CMM está enfocado hacia el software y se ha convertido en un conjunto estándar de requerimientos para la mejora del proceso para muchas organizaciones de software. Muchas de ellas están adoptando prácticas del modelo como una forma no solo de reducir costos, sino también para convertirse en proveedores de software elegibles. Este modelo de mejora gradual brinda a las organizaciones la guía que necesitan para moverse hacia las mejores prácticas.

El modelo teóricamente es voluntario, pero el gobierno de EUA impone en algunos casos que sus proveedores demuestren una puntuación de al menos el nivel dos o cuatro. Por lo que una gran compañía se beneficia de calificar en estos niveles pues puede ser elegible para los proyectos del gobierno.

La mejor forma para empezar un programa de mejora es mientras una organización es pequeña, con pocos ingenieros de software y con una estructura de administración simple. El CMM brinda un método en el que todos los roles y las responsabilidades están claramente definidos. Debido a que el CMM es una guía y no una especificación rigurosa de requerimientos, es flexible y fácilmente adaptable a una organización de cualquier tamaño. Las funciones y los roles pueden ser mapeados a los empleados disponibles, donde a cada uno se le asignará cierto número de responsabilidades.

Desventajas generales

La siguiente gráfica muestra las últimas cifras disponibles del estado de madurez de la industria del software:



Como se puede ver, las cifras no son muy alentadoras, ya que la lenta mejora en el proceso de madurez refleja la dificultad experimentada por las organizaciones para alcanzar niveles mas altos.

Esto generalmente se debe a que el compromiso de la administración no es constante, que los agentes de cambio, frecuentemente carecen de la habilidad para hacer que se alcancen las áreas claves de proceso o a que los objetivos del proyecto carezcan del enfoque necesario para adoptar las prácticas de administración e ingeniería que son requeridas. Por lo que es importante tener presente que:

1. El CMM esta diseñado principalmente para organizaciones que trabajan con grandes contratos para el gobierno. Es aplicable a compañías pequeñas, pero se necesita hacer una interpretación extensiva y razonable para esa industria en algunas partes del modelo.

2. El CMM no tiene bases teóricas formales, esta basado en la experiencia de 'gente muy conocedora', por lo que la teoría parece ser que los expertos saben lo que están haciendo, por lo tanto cualquier otro modelo basado en la experiencia de gente conocedora tiene la misma validez.

3. El CMM tiene un vago soporte empírico. El modelo está basado principalmente en la experiencia de grandes contratistas gubernamentales y la experiencia de Watts Humprey en el mundo mainframe. No toma en cuenta el éxito de compañías más pequeñas como Microsoft, Apple o Borland, y los niveles uno, cuatro y cinco no están bien representados por los datos: el primero debido a que está mal representado y los últimos porque hay muy pocas compañías con dichos niveles. Aunque hay varios

reportes de los éxitos del CMM, mas bien son una recopilación de éxitos de la gente que trabaja junta para obtener algo.

4. El CMM reverencia el proceso, pero ignora a la gente, pues consideran que no son fiables y asumen que un proceso definido puede reducir dichas fallas. La idea de que el proceso puede arreglar la mediocridad es el pilar del CMM.

5. El CMM establece la institucionalización del proceso, ya que el CMM establece la habilidad de una organización para cumplir, aunque esto es solo la habilidad del equipo de software para ejecutar. Si los procesos no están formalmente institucionalizados, pueden estar bien colocados debido a la habilidad de los miembros de los equipos de software.

6. El CMM motiva el desplazamiento de la verdadera meta que es mejorar un proceso, por una meta falsa que es alcanzar un nivel de madurez mas alto, lo que tiene como efecto que una organización se ciegue para el uso más eficiente de sus recursos.

Pero el argumento más frecuente en su contra, es que, como una eficiente receta de mejora de procesos, muchas organizaciones exitosas no deberían existir, de acuerdo sus criterios.

La innovación no está contemplada en el modelo, el SEI argumenta que la innovación está fuera del alcance del modelo y solo establece un esquema dentro del cual puede ocurrir, pero debido a que el CMM está demasiado preocupado por la predictibilidad ignora completamente la dinámica de la innovación y la considera como caos, colocando a las organizaciones innovativas en el nivel uno. Su problema principal es que desconfía de las contribuciones individuales e ignora las condiciones necesarias para nutrir ideas no lineales y las sepulta bajo una superestructura. Brindando la ilusión de control administrativo.

Los innovadores aconsejan a las compañías ser flexibles, el CMM les aconseja ser predecibles, donde los innovadores sugieren ignorar a las autoridades, el CMM pide obediencia absoluta, donde se recomienda una constante innovación, el CMM la confunde con el caos del nivel uno, donde la innovación depende de experiencias aprendidas, el CMM depende de prácticas escritas. Además el CMM está descrito en un gran número de páginas, por lo que es pesado y tardado tratar de comprenderlo y aplicarlo.

Para que la organización madure, debe haber capital que apoye el esfuerzo. Muchas organizaciones han gastado grandes cantidades para apoyar sus iniciativas de mejora y tienen muy poco, si es que algún conocimiento de lo que obtendrán de su inversión. Los costos de una mejora de proceso basada en el CMM pueden mostrar incrementos en software, hardware, recolección de datos, diseño y corrección de defectos. Una queja universal es que moverse de un nivel a otro puede costar cientos de miles de dólares e incluso millones.

La innovación no aparece como tal en el CMM, y solo es sugerida en el nivel cinco. Esto es alarmante ya que las firmas más innovadoras en el área de software son pequeñas y están en el nivel uno. Por contraste compañías grandes como IBM, en cuya historia se encuentran desastres como el proyecto de la automatización avanzada de la Administración de Aviación Federal, tienen puntuaciones muy altas de madurez.

Mejorar el proceso de software incluye el incremento de tiempo de entrenamiento y la disminución de tiempo disponible para trabajar en los proyectos. Una organización tiene que seguir trabajando como de costumbre o hacer sacrificios para mejorar su proceso del cual espera obtener mejores productos. Algunas organizaciones, debido a sus compromisos de entrega, tienen dificultad para trabajar en esta mejora.[13]

Puntos de vista de algunos autores sobre el CMM

Algunos autores consideran que muchas organizaciones siguen en el nivel uno debido a que los patrones del CMM son demasiado rígidos, por ejemplo, compañías como Borland, Microsoft, Lotus entre otras, rara vez, si es que lo hacen, manejan sus documentos de requerimientos tan formalmente como lo establece el CMM y esto es indispensable para llegar al nivel dos.

A continuación se describen diversos problemas que algunos autores encuentran en el CMM:

Para **Gerald Weinberg**, madurez no es la palabra adecuada, considera que es muy tentador, cuando se habla sobre culturas caer en juicios, y esa es precisamente la finalidad del CMM, establecer una cultura para la mejora del software. Considera que la palabra madurez no es un hecho, sino un juicio o interpretación de los hechos y en el peor de los casos no se ajusta a estos.

La madurez normalmente va en una sola dirección, y en el lenguaje coloquial significa que se ha alcanzado el pico natural de crecimiento, por lo que sí una organización ha llegado a 'su madurez', significa que se quedara allí, a menos que algo anormal suceda.

Considera que la búsqueda por la perfección no es madura, sino infantil, haciendo una analogía entre las ideas de Hitler sobre la raza superior y las razas menos maduras, pues para este tipo de modelos, se debe admitir en que nivel de madurez se encuentra.

Objeta el término de madurez porque, desde su punto de vista, implica un estado de superioridad sobre los menos maduros. Las diferencias entre las organizaciones están mas en la naturaleza de sus comportamientos y sus culturas, mas que en el progreso de "una búsqueda injustificada de la perfección".

James Bach considera que la mayoría de las compañías están en el nivel uno, pero está falta formal de madurez no evita que algunas de estas progresen a través de producir frecuentemente (pero no siempre) productos de calidad. Considera que debe ponerse mas énfasis en la naturaleza e interacción de las personas involucradas en lugar de definir procesos ideales como única base para un éxito consistente. El mejor desarrollo de software no puede cubrir la brecha de eficiencia entre él mas y el menos capacitado de los programadores. Se debe trabajar continuamente en las habilidades del equipo, a través de mejorar sus habilidades, trabajo de grupo, ambiente de desarrollo y contratación de las mejores personas que se puedan encontrar. [13]

Capers Jones considera que la aplicación de "niveles artificiales de madurez" para caracterizar los procesos, métodos y ciclos de vida del software es un riesgo potencial, se inventan esquemas artificiales de clasificación en un intento de simplificar problemas complejos, frecuentemente con la ausencia de datos cuantitativos. Considera que este modelo es deficiente por las siguientes razones:

- Es incompleto e ignora factores tales como la moral de los participantes, planes de compensación, estructuras organizacionales, métodos de reclutamiento, espacio de oficinas y niveles de ruido, entre otros muchos factores de riesgo para el software.

- No registra datos cuantitativos sobre productividad, calidad, tiempos, etc.

- Es esencialmente manual y más caro de lo necesario.

- Ha sido diseñado con base a la experiencia de grandes empresas y en contra de las pequeñas pues estas carecen la falta de miembros necesarios para alcanzar niveles superiores al dos.

- El SEI ha hecho afirmaciones sobre los grados de calidad y productividad de sus niveles, sin ninguna prueba empírica.

- La estructura binaria de las preguntas del CMM es inadecuada y crea patrones de respuestas forzadas.

- El concepto de madurez del CMM carece de un punto medio o promedio y no es realmente útil para estudios estadísticos ya que muchos datos no son considerados en el nivel 1.

- El concepto del CMM está enfocado a las tecnologías mainframes y tiende a ignorar algunos de los factores asociados con el desarrollo orientado a objetos, cliente-servidor, multimedia y similares.

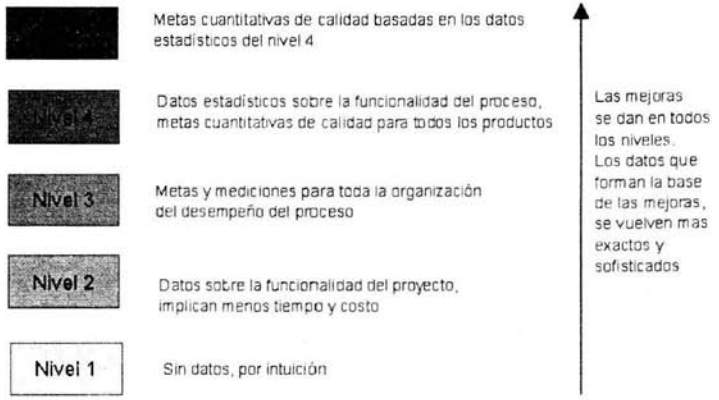
- El CMM no es modernizado tan rápido como las tecnologías de software y no cubre tópicos recientes tales como análisis y diseño orientado a objetos.

- Actualmente, los niveles de madurez son importantes para los contratos de gobierno, y los contratantes están forzando a sus administradores para que den solo patrones de respuesta positivos.

- El enfoque del CMM es ambiguo, pues no hay garantía de que las mediciones elaboradas por diferentes grupos brinden los mismos resultados [11].

4.2 VENTAJAS Y DESVENTAJAS DE CADA NIVEL

En la siguiente figura se muestran los productos resultantes que se obtienen en cada nivel, los cuales son usados como base para medir la mejora del proceso.



Nivel Uno
No existe información.

Nivel Dos
Se establecen procesos básicos de administración para el proyecto a fin de monitorear costos, tiempos y funcionalidad. La disciplina necesaria del proceso es aplicada para repetir éxitos preliminares en proyectos similares.

Ventajas

- Se tiene un consenso sobre lo que se va a hacer entre todos los involucrados por escrito y este es usado para organizar la planeación de las actividades a llevar a cabo.

- Se organiza un grupo de ingeniería de software que será el encargado de revisar el apego al proceso que somete a revisión, con todos los involucrados, los resultados a fin de obtener aquellas desviaciones que pueden afectar al plan a fin de efectuar los ajustes al plan.

- Se define un grupo de aseguramiento de la calidad que valida los productos de trabajo.

- Se asigna la responsabilidad de la configuración del software a un grupo especializado.

- Muchas organizaciones pequeñas tienen la necesidad de subcontratar partes de sus proyectos, o aun proyectos completos a otras compañías. En este nivel se dan recomendaciones que pueden ayudar en la evaluación de estas compañías para elegir al mejor candidato para este contrato. Si el elegido está en el nivel 1 del CMM, los riesgos involucrados con esta compañía pueden ser reducidos a través de un monitoreo constante. O la compañía puede elegir a un candidato que este en el nivel 2 o 3 si el proyecto es mas crítico y delicado.

Desventajas

- Al definir tantas funciones, se requiere a mucha gente. Y la capacitación necesaria para llevar a cabo algunas de estas actividades no se menciona por ningún lado, dando por hecho que la organización tiene todos estos recursos y da por hecho que cuentan con el suficiente conocimiento para llevarlas a cabo.

Nivel Tres
El proceso de software para las actividades de administración e ingeniería es documentado, estandarizado e integrado dentro de un proceso estándar para la organización. Todos los proyectos usan una versión aprobada y adecuada del proceso estándar de software para desarrollar y dar mantenimiento.

Ventajas

- Se evalúa el proceso organizacional a fin de hacerle ajustes de acuerdo a los resultados previamente obtenidos.

- Se adecua el proceso a cada proyecto de acuerdo a las características de este y sus resultados se ponen a disposición de toda la organización para contribuir al reajuste su proceso.

- Se inicia la detección de los planes de capacitación tanto técnica como administrativa para cada rol dentro de los proyectos.

- Cada adecuación del proceso organizacional para el proyecto es documentada y debe ser aprobada.

- Se tiene la identificación de herramientas de ingeniería de software para desarrollar productos.

- Los planes, tareas y productos coinciden con los comprometidos en los requerimientos.

- Hay una coordinación entre todos los grupos participantes para planes y actividades.

- Se revisan productos ya identificados como críticos por el personal calificado pero no se usan sus resultados para evaluar al desarrollador.

Desventajas

- Se requiere la aprobación forzosa de las adecuaciones del proceso organizacional para el proyecto. Lo que implica rigidez, burocracia y costo.

- Define un conjunto común de recursos para la organización, lo cual generalmente no es así, pues el personal de un grupo puede tener las habilidades requeridas en otro grupo y no esta disponible para ese proyecto pues pertenece a un grupo ajeno a dicho proyecto.

- Se asume que un proceso definido claramente va a dar como resultado excelentes productos, sin tomar en cuenta la experiencia del personal involucrado en el proyecto.

Nivel Cuatro

Medidas detalladas del proceso de software y la calidad del producto son recopiladas. Tanto el proceso de software como los productos son entendidos y controlados cuantitativamente.

Ventajas

- Se analizan los procesos para identificar las causas de las variaciones en

los resultados contra los planes, a fin de establecer límites de calidad.

- Se controla el acceso a los datos y en base a ellos se establecen metas de funcionalidad del proceso.

- Cada proyecto define sus metas de calidad para sus productos estableciendo un grupo encargado de su revisión.

- Autocrítica objetiva a través de criterios que apoyarán a estos grupos para determinar el éxito de los productos de software.

Desventajas

- Muy pocas compañías se encuentran en este nivel.

- El modelo exige un patrocinio constante para los continuar los esfuerzos.

- Es necesario contar con habilidades de administración cada vez mas altas.

- Se necesita un compromiso a largo plazo.

Nivel Cinco

La mejora continua de procesos es establecida por una retroalimentación cuantitativa de los procesos y de probar ideas innovadoras y tecnologías.

Ventajas

- Se establecen planes con compromiso de recursos para la prevención de defectos.

- Las actividades de prevención de defectos son incluidas en el plan de cada proyecto. –v.g. revisiones exhaustivas antes de liberar-.

- Se evalúan mejoras a la tecnología organizacional y del proyecto.

- Se definen los objetivos cuantitativos y medibles para la mejora del proceso.

Desventajas

- Muy pocas compañías se encuentran en este nivel
- Solo hasta este nivel hay innovación tecnológica, por lo que no hay flexibilidad para las necesidades de los clientes.

4.3 RECOMENDACIONES

El CMM no guía a métodos avanzados de desarrollo de software. El CMM no se apoya en ningún método que no haya sido conocido hace 20 años. Trabaja a otro nivel, dice en cual de las muchas cosas que son consideradas sobre el desarrollo de software, debe concentrarse, y plantea las preguntas que deben hacerse y no como contestarlas. Puede sonar muy simple y aburrido pero era exactamente lo que la industria del software necesitaba para hacer un progreso real para desarrollar software.

Mitos a ser evitados

- Tener expectativas muy altas.
- Esperar que todas las áreas de proceso inicien en el nivel dos.
- Esperar un avance de nivel por año.
- Esperar que la mejora ocurra por sí misma.

Recomendaciones generales

Toma tiempo moverse de un nivel a otro

Una organización pequeña con cinco o diez ingenieros de software puede moverse del nivel uno a niveles más altos en cuestión de semanas o meses, ya que la cultura de estas organizaciones tiende a motivar la innovación.

El verdadero problema es con las organizaciones grandes, aquellas que tienen a mas de 100 profesionales de software y especialmente con aquellas que tienen mas de 1000, para ellas es razonable esperar de dos a tres años para alcanzar el siguiente nivel. La transacción más difícil es del nivel uno al dos. Como dice Humphrey:

"Para el momento en que una organización invierte en todo el trabajo de ir del nivel uno al nivel dos y lo escribe, frecuentemente encuentra que cuando se encuentra 'oficialmente' en el nivel dos, está realmente en camino a llegar al nivel tres."

De manera similar es relativamente fácil llegar del nivel tres al cuatro, sin embargo, es más difícil alcanzar el nivel cinco, ya que este representa un cambio en los paradigmas, debido a que el énfasis ahora es como prevenir problemas, ya que la organización ha desarrollado procedimientos de convertir listados de fallas en productos de software, tiene que aprender a hacer lo mismo con el proceso. Lo

que significa que la gente ahora se adueñe del proceso y no el proceso de la gente.

Se calcula que para una organización grande, el llegar del nivel uno al nivel cinco lleva alrededor de diez años y esto ha causado terror en las gerencias de organizaciones de software ya que temen quedar fuera del negocio, pero de cualquier manera, organizaciones importantes están tratando de alcanzar estos niveles primero. Aun no hay fórmulas mágicas para acelerar la mejora del proceso, aunque se están buscando ideas que la aceleren, pero obviamente ayuda que haya un fuerte compromiso y conciencia sobre esto, pero no importa que tan grande sea el compromiso, no se puede ignorar que las organizaciones grandes tienen una enorme cantidad de inercia.

También las organizaciones pueden regresar a niveles más bajos, especialmente del nivel dos al uno, esto puede ser el resultado de adquisiciones o uniones corporativas, sobre todo si el proceso no es controlado por la administración corporativa.

No hay muchas organizaciones por encima del nivel uno

En una estadística hecha por el SEI realizada en abril de 1996, los resultados indican que aproximadamente el 68.8% de las organizaciones están en el nivel uno, 18% está en el nivel dos, 11.3 % está en el tres y 1.5% se encuentra en el nivel cuatro y el 0.4% está en el nivel cinco. En cuanto a los proyectos revisados en esta estadística el 88% se encontraba en el nivel uno, 5% en nivel dos, 5% en el nivel tres y 2% en el nivel cinco. Estas estadísticas involucran a toda la organización, no a equipos de proyecto individuales o departamentos de la organización.

Tecnología nueva debe ser evitada en niveles inferiores

En los niveles uno y dos los principales puntos son referentes a la administración, tales como revisión de presupuestos, tiempos y administración de la configuración, de tal manera que la introducción de metodologías y herramientas CASE, no es necesaria, sino que será un desperdicio de dinero, por lo que deben ser prohibidas hasta que la organización alcance el nivel tres.

Lo anterior no es cierto, ya que lo que se desea es que la organización no dependa de una introducción costosa y masiva de nueva tecnología para resolver sus problemas del nivel uno y dos. Un uso inteligente de las herramientas puede acelerar el progreso de la mejora del proceso y el uso de nuevos procesos puede conducir a la necesidad de mejores herramientas. Se pueden usar herramientas sencillas en los niveles inferiores, ya que al usar herramientas complejas la organización que está en nivel uno puede verse abrumada por la formalidad y rigor requerido por estas, desechando su uso lo que significa un costoso error que nadie mencionará.

No es factible que las nuevas organizaciones inicien en el nivel tres

El dilema que enfrentan la mayoría de las organizaciones de software es que la inercia de una cultura existente no puede ser sobrepasada rápida y fácilmente, no importando cuan grande sea el deseo del gerente. ¿Pero que pasa con una organización que inicia y no tiene ningún personal?, ¿Es posible iniciar contrataciones y crear de manera instantánea una organización del nivel tres o cuatro?

La respuesta es que no es probable, aunque el gerente contrate a la gente con una actitud adecuada y si él tiene experiencia trabajando con las metodologías del SEI, las posibilidades son que los nuevos empleados no han trabajado juntos y necesitaran tiempo para adaptar su comportamiento anterior al de la nueva cultura.

Humprey compara esta situación a un equipo deportivo compuesto solo por estrellas. Aunque cada uno de los jugadores haya sido el mejor de su equipo anterior, no necesariamente podrán trabajar juntos de la manera adecuada.

No saltarse los niveles

Frecuentemente, una organización que se encuentra en nivel uno desea saltarse algunos niveles y encontrarse mágicamente en el nivel cuatro o cinco, esto está frecuentemente acompañado por un gasto excesivo con la finalidad de resolver los problemas asociados al comportamiento del nivel uno y dos. Desafortunadamente esto no es fácil, ya que mucho del avance entre un nivel y otro es cultural, por lo que se debe ir de un comportamiento caótico a uno ordenado a través de un consenso informal antes de alcanzar un proceso formal para desarrollar software. Millones de dólares han sido gastados en entrenamiento, pero sin cambios a la cultura, los ingenieros de software, van a los cursos y después regresan a sus escritorios a continuar haciendo lo que hasta ese día han hecho de la misma manera.

Debido a que tiene sus raíces en sistemas de gobierno y militares tiene ciertos supuestos que pueden no cumplirse para el sector comercial.

También no es recomendable incluirlo en proyectos que están muy avanzados y que están a punto de llegar a la fecha límite sin resultados concretos, muchas veces esto se hace para tratar de salvar un caso perdido, ya que la imposición de métodos formales en un proyecto avanzado que se ha desarrollado sin ningún control, implica que este se detenga para que los involucrados lean los manuales, conozcan su metodología y las nuevas herramientas, ya que la mayoría de ellos no sabe que es lo que debe hacer pues si se impone de manera dictatorial y sin un entrenamiento o introducción adecuado seguramente volverán a hacer con los pasos de esta metodología, aspectos que ya tenían cubiertos. Por lo que se recomienda que este sea introducido como parte de una estrategia corporativa a largo término, experimentando con proyectos piloto y proporcionando la capacitación necesaria para ello.

4.4 ORGANIZACIONES QUE UTILIZAN EL CMM[12]

Desde 1987, ha sido medida la eficiencia de 1,500 organizaciones y casi 10,000 proyectos han reportado sus resultados al SEI. Mas de la mitad de las organizaciones que participan en la mejora a través del CMM son casas de software comercial y grupos de desarrollo interno de industrias tales como financieras, aseguradoras, bienes raíces, ventas al menudeo, construcción, transporte, comunicaciones, maquinas industriales, equipo electrónico, instrumentos médicos, entre otras, de los cuales la cuarta parte se encuentra bajo contrato para el gobierno de EUA. El 5% de estas organizaciones son militares o agencias gubernamentales, el tamaño de las organizaciones varía ampliamente y alrededor de la mitad tienen menos de 100 personas en el área de sistemas, una cuarta parte tiene mas de 200 y el otro cuarto menos de 50 personas en el área de sistemas.

A continuación se presenta la lista publicada por el SEI sobre las organizaciones que han sido medidas y en que nivel se encuentran, pero hay que tomar en cuenta las siguientes consideraciones:

- El SEI no certifica a las compañías en niveles de madurez.
- La lista no es exhaustiva, puede haber organizaciones que no estén listadas.
- Las organizaciones listadas dieron su autorización para ser publicadas en ella.
- Ninguna información confidencial fue usada en esta lista.

NIVEL DOS

ORGANIZACIÓN	FECHA DE PUBLICACION
96th Communications Group	Octubre 15, 1999
Accenture, CIO Solution Delivery Madrid Development Center	Julio 26, 2002
Accounting Center of China Aviation Beijing, China	Junio 14, 2002
ACS Government Systems Lexington, KY	Octubre 18, 2002
Advanced Technology Systems, Inc. (ATS*), Enterprise and Management Systems (EAMS) Division	Marzo 2, 2001
Alpha Data Corporation	Febrero 1, 2002
ALSTOM Transport, Montreal Telecite Facility	Febrero 20, 2002
ALSTOM Transport, Rochester, New York Facility	Enero 21, 2002
Anritsu Corporation	Agosto 20, 2001
Anritsu Engineering Co., LTD.	Mayo 31, 2001
Asahigin Systems Co., Ltd. Minami Aoyama, Minato-ku, Tokyo, Japón	Diciembre 6, 2002
AT&T Network Systems Advanced Software Construction Center	Nov/Dic 1995
Boeing Information Services, RCAS Data and Applications Department, Vienna, VA	Junio 25, 1997

ORGANIZACIÓN	FECHA DE PUBLICACION
Booz-Allen & Hamilton, Inc., World wide Technology Business, Civil Market Team	Octubre 1, 1999
Bosch Automotive Systems Corporation, Development Division	Noviembre 6, 2001
CIGNA Corporation, Production Application Center	Marzo 2, 1998
Collective Intelligence, Inc. Headquarters Division	Marzo 2, 2001
Korea First Bank Divisioin : Information and Systems Office Location : 7-11 ShinChun-dong SongPa-gu Seoul, Korea, 138-240	Enero 20, 2003
Korea First Data Systems Division Name : Quality Management Team Location : 7-11 ShinChun-dong SongPa-gu Seoul, Korea, 138-240	Enero 24,2003
DataSource, Inc	Enero. 11, 2000
Dba Engenharia De Sistemas - Software Plant - Rio De Janeiro, Brasil	Julio 27, 2001
Digital Solutions, Inc., 88007 Sudley Road, Suite 112, Manassas, VA 20110, (703)335-5015	Octubre 3, 2002
Dljdirect Subsidiary Inautix Technologies	N/D
Dynamics Research Corporation	Septiembre 17, 1997
EDS (Electronic Data Systems) Spain North East Solution Center	Junio 29, 2002
Extreme Systems Co., Ltd.	Febrero 1, 2001
Fuji Xerox	Mayo 14, 1998
Fuji Xerox (China) Ltd, Fuji Xerox Software Development Center of China	Agosto 3, 2001
Fulton & Associates, Inc.	Marzo 3, 2002
Hewlett Packard	Agosto 1996
Hewlett Packard, Systems Interconnect Solutions Lab	Enero 15
Information Technology Resources	Septiemiebre, 13 2000
Johns Hopkins University, Applied Physics Laboratory, Information Systems Engineering Group of the Power Projection Systems Department, Laurel, Maryland	Diciembre 13, 2002
Korea First Bank, Information and Systems Office 7-11 ShinChun-dong SongPa-gu, Seoul, Korea, 138-240	Enero 20,2003
Korea Securities Computer(KOSCOM) Maket Support Division Seoul, Korea	Noviembre 15, 2002
Link S.A., Gerencia de Proyectos de Software, Santiago, Chile	Enero 8, 2003
Lockheed Martin Aircraft Services Division	Nov/Dic 1995
Madison Research Corporation, Huntsville, AL-Software Engineer Division	Enero 1, 1999
Mellon Financial Corporation - Trust Technology Division	Dicembre 7, 2000
Mellon Financial Corporation, Mellon Global Cash Management	Abril 5, 2001
Mellon Financial Corporation, Retail Financial Services	Enero 25, 2001
Modern Hi-Tech Development Co.,Ltd.(Dalian, China)	Marzo 2002
modis Professional Services, Inc., c modis Incorporated	Julio 28, 1999
Motorola CIG (Cellular Infrastructure Group)	Septiembre 1994
Motorola Transmission Products Group - Canton, Mass.	Abril. 1994
NEC do Brasil S.A.-Switching; Radio and Transmission Divisions	Septiembre 21, 2000
Oerlikon Aerospace	Agosto 3-7, 1997
Ogden SW Development Activity (MSG/SO) at Hill AFB	Nov/Dic 1995

ORGANIZACIÓN	FECHA DE PUBLICACION
PB Farradyne, Inc. A division of Parsons Brinckerhoff Quade and Douglas	Enero 8, 2002
PEC Solutions, Inc., Civilian Government Solutions, Fairfax, Virginia	Enero 13, 2003
PEC Solutions, Inc., Development Operations	Julio 26, 2001
Pentastar Electronics, Inc.	Nov/Dic 1995
PROCASE Corporation	N/D
QSS Group, Inc.	Diciembre 2001
Questra Incorporated, Rochester, NY	Noviembre 6, 2000
Samsung Thales CIS Team, Production Division	Mayo 24, 2002
SimCorp, TMS2000 Development Department	Septiembre 5, 2001
Social Security Adminsitration	Agosto 19, 2002
Software Research Associates, Inc., IT Industry System Division, Tokyo, Japon	Febrero 1, 2000
STMicroelectronics Consumer and Microcontroller Groups Display & TV Division Grenoble and Rousset, France	Abril 26, 2001
Sumaria Systems, Inc 410 Interstate Park Drive Montgomery AL	Julio 8, 2002
Toden Software, Inc.	Noviembre 9, 1999
TranTech, Inc., Alexandria, VA	Septiembre 27, 2002
Truimph International Japón, LTD. - IT Department	Agosto 16, 2000
TYBRIN Corporation	Nov/Dic 1995
United Airlines	Septiembre 14, 1998
United Airlines, Information Services Division	Octubre 24, 1997
Vector Research, Inc.	Junio 12, 1998
Xerox Corporation, Global Software Welwyn Inglaterra Software Center	Octubre 24, 2001

NIVEL TRES

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
AC Technologies, Inc., Fairfax VA	2001
Accenture Teleworks, Minneapolis	Octubre 28, 1997
Accenture Utilities Solutions Center	N/D
AccentureTeleworks, Manila	Octubre 28, 1997
Advanced Information Services Inc.	Noviembre 1999
Air Force, Standard Systems Group	Septiembre 26, 1997
América XXI, Consultoría en Calidad de Software,Chile	Septiembre 13, 2002
American Power Conversión	Marzo 23, 2000
AMS Industrial & Utilities Consulting Systems Group	Noviembre 15, 1999
Anritsu Corporation	Enero 31, 2003
Anritsu Engineering Co., LTD	Diciembre 27,2002
Applications Services and Operations Division, Nedcor Bank Limited	Abril 9, 2002
Applitech Solution Limited, Ahmedabad	Marzo 30, 2001
AT&T	Febrero 25, 1998
Beijing Huahong NEC IC Design Co., Ltd. LSI Software Development Division Beijing, China	Junio 11, 2002
Bellcore Software Systems	N/D
Bharti Telesoft Intl Pvt Ltd. New Delhi, India	Mayo 20, 2002
Blue Cross Blue Shield of Maryland	Agosto 11, 1998
Boeing Company, Special Operations Forces Aerospace Support Center	Junio 1998

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Boeing Wichita Information Systems	Abril 1998
CACI International Inc., Federal Systems Integration Business Group	Mayo 4, 2000
Cap Gemini Ernst & Young Consulting India Pvt Ltd Mumbai India, ADC & AMSC(Application Development, Integration and Application Management).	Enero 26, 2003
Cap Gemini Ernst & Young US LLC New York and Chicago Advanced Development Centers	Mayo 24, 2002
Carderock Division, Naval Surface Warfare Center, Naval Sea Systems Command Ship Systems Engineering Station, Filadelfia,PA	Diciembre 2000
Carrier	Febrero 25, 1998
Citibank Canada, Information Services Division	Septiembre 9, 1999
Codelinks Data Services PVT. LTD. Hyderabad, India	Junio 2001
Company: Covansys (earlier known as Complete Business Solutions, Inc. – CBSI) Division: Columbus DCE	Enero 16, 2001
Computer Sciences Corp (CSC), Chemical Group	Mayo 9, 2002
Computer Sciences Corporation	Diciembre 20, 1999
Computing Devices Canada, A General Dynamics Company	Enero 1999
Conquest Systems Inc. Strategic Software Development Group 1023 15th Street NW Washington DC 20005-2602	Enero 31, 2001
Coritel S.A, Coritel SDC Division	Junio 29, 2002
Corporate Information Solutions, Inc. Solution Centers Computer Services Bldg., Meralco Center Pasig City, Filipnas	Agosto 3, 2002
CTG	Junio 28, 2000
Daiwa Computer Co., Ltd.,36-18, Wakamatsu-cho, Takatsuki, Osaka 569-0054	Enero 24, 2002
Ddemesis	Diciembre 2001
Decision Consultants, Inc.	Octubre 12, 1999
Electronic Data Systems - Information Solutions Division, Brasil Solution Centre (BSC)	Agosto 22, 2001
enherent (Barbados) Ltd.	Diciembre 16, 1997
Ericsson Telecomunicacoes S/A Brazil, Research and Development Center	Marzo 15, 2001
FITEC Corp.	Noviembre 1, 2002
Fuji Xerox, DPC CP&S	Abril 2001
Fujitsu Limited, Systems Integration Business Division the Government and Public Systems Division	Abril 9, 2002
GDE Systems (subsidiary of Tracor, Inc.)	Nov/Dic 1995
General Dynamics - Electric Boat Corporation	Noviembre 29, 2000
General Dynamics Corporation - Computing Devices Co. Ltd.	Junio 20, 2001
General Dynamics-Communication Systems, Needham-Taunton, MA	Octubre 1999
General Dynamics-Electronic Systems, Thousand Oaks, CA	Octubre 1999
Grumman Data	Marzo 1995
GTE Government Systems at Mountain View, CA and Tempe, AZ	Mayo 1996
HCC Infotech Limited, Software Consultancy Design, Development, Installation and Implementation	Septiembre 9, 2002
Healthcare Media Private Limited (HMP)	Octubre 29, 2001

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Honeywell Building Solutions Center, Europa	Enero 1, 1999
HTC Global Services, Inc, Soutfield, MI	Agosto 14, 2000
Hughes Missile Systems Company	Noviembre 1996
Hyundai Information Technology Co., Ltd(HIT) Yong In City, Korea	Febrero 21, 2003
Hyundai Information Technology Co., Ltd(HIT) Yong In City, Korea	Febrero 21, 2003
IBM Global Services - Application Management Services Holanda	Junio 2002
IBM Global Services - Application Managment Services	Diciembre 2000
IBM Global Services, Global Test Organization	Diciembre, 2001
ICICI Infotech Services Ltd.	Septiembre 15, 2001
Idea Integration, National Center for Excellence for RoadMap (TM) Methodolgies, India	Octubre 2, 1999
Information Technology Resources, Inc.	Agosto 24, 2001
Intelligroup Asia Pvt. Ltd., Advanced Development Center (Hyderabad)	Octubre 30, 1999
Keane Inc. Public Service Electric and Gas Outsourcing Contract	Abril 29, 1998
L-3 Communciations Integrated Systems (IS), (formerly Raytheon Aircraft Integration Systems (AIS))	Agosto 5, 2001
L-3 Communications, Communication Systems-East	Julio 09, 2002
Litton Data System Division	Nov/Dic 1995
Litton TASC, Chantilly, VA	Mayo 16, 2000
Lockheed Martin Aeronautical Sytems, Marietta, GA	Diciembre. 4, 1999
Lockheed Martin Information Support Services	Marzo 9, 2000
Lockheed Martin Management & Data Systems	Septiembre 1996
Lockheed Martin Technical Operations (LMTO is a Business Unit of the Technology Services Business Area of Lockheed Martin Corporation)	Febrero 8, 2002
Logicon Advanced Technology	Enero 20, 2000
Magneti Marelli Powertrain S.p.A. - SDC (Software Development Center), Catania, Italia	Julio 26, 2002
Mahindra-British Telecom Limited, Software Development Centres at Mumbai and Prune	Marzo 1999
McDonnell Douglas Aerospace (St. Louis)	Nov/Dic 1995
McDonnell Douglas Space & Defense Systems & Transport Aircraft	Nov/Dic 1995
Mellon Financial Corporation - Trust Technology Division	Enero 14, 2002
Mellon Financial Corporation, Global Cash Management Systems Division	Febrero 21, 2002
MicroPact Engineering, Inc.	Julio 26, 2002
MilSOFT Software Technologies Inc., Ankara Turkey	Septiembre 18, 2002
Modern Hi-Tech Development Co., Ltd.(Dalian, China)	Marzo 6, 2003
Moog-Aircraft Group	Diciembre 20, 1999
Motorola Brazil Design Center, Personal Communication Sector, Latin American Engineering	Diciembre 4, 2001
Naval Surface Warfare Center (NSWC), Port Hueneme	Febrero 1998
NCS Pearson, Government Solutions	Agosto 1, 2001
NEC Communication Systems, Ltd., Engineering Division 1st Software Department	Mayo 19, 2000
Network Systems & Technologies (P) Ltd.	Abril 8,1998
Nihon Unisys, Ltd., E-Technologies & Services, Koto-ku,	Diciembre 19, 2002

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Tokyo, Japón	
NIIT	Diciembre 19, 1997
Northrop Grumman Information Technology, Defense Mission Systems	Diciembre 3, 2001
Northrop Grumman Information Technology, Government Solutions (formerly Logicon Internal Information Services)	Febrero 25, 1998
Ogden Air Logistics Center Software Engineering Division at Hill AFB	Nov/Dic 1995
Orion International Technologies, Inc., Atlantic Fleet Weapons Training Facility (AFWTF), Roosevelt Roads, Puerto Rico	Abril 28, 2000
Paragon Solutions (I) Pvt. Ltd	Marzo 5, 2002
Posdata	Noviembre 12, 2001
pragma Systems Corporation	Septiembre 6, 2000
Pragmatics Inc.	Enero 29, 2001
PricewaterhouseCoppers India, TS SBU	Mayo 16, 2002
PwC Consulting, Washington Consulting Practice (WCP), software Solutions Group (SSG)	Mayo 22, 2002
Raytheon Intelligence and Information Systems State College, Pa	Noviembre, 1997
Raytheon Electronic Systems	Agosto 1995
Robert Bosch India Limited (RBIN), Software Division	Noviembre 11, 2000
Rockwell Collins Government Systems	Febrero 9, 2001
Sacramento Air Logistics Center Softwae Div. @ McClellon AFB	Nov/Dic 1995
SAIC Systems and Software Engineering Support (SSES)	Febrero 1997
Samsung SDS CASE (Center for AdvacnedSoftware Engineering)	Diciembre 28, 2001
Schlumberger Limited - Semiconductor Solutions - Ferndown Product Development Centre	Noviembre 27, 2000
Schlumberger Limited-Semiconductor Solutions-Saint Etienne Center	Febrero 25, 2002
Science Applications International Corporation, Environmental Protection Agency's Systems Development Center	Marzo 10, 1998
SHCIL, IT Division, Mumbai, India	Enero 23, 2002
Singapore Housing and Development Board, Information Services Department	Octubre 19, 2001
Smiths Industries Aerospace & Defense Systems Inc., Grand Rapids Division	Junio 27, 2000
Sodalia	Octubre 4, 1997
Software Eng. Div. of Hughes Aircraft - Fullerton, CA	Julio 1991
Software Research Associates, Inc.	Diciembre 21, 2001
Space and Naval Warfare Systems Center San Diego	Noviembre 10, 2000
Space and Naval Warfare Systems Center, San Diego	Noviembre 10, 2000
SRA International, Inc., Consulting and Systems Integration	Junio 2001
Ssi Technologies a Division Of Ssi Limited Chennai, Tamil Nadu, india	Febrero 12, 2002
STN ATLAS Elektronik GmbH Underwater Vehicles	Abril 4, 2002
Sverdrup/AEDC	Noviembre 15, 2001
System Sophia Co.,Ltd 2-2-10 Utoh Shizuoka-city Japón	Noviembre 13,2002
Systematic Software Engineering A/S, Aarhus Dinamarca	Abril 18, 2002
Tecnologia en Souciones y Sistemas	Abril 3, 2000

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Telos Corporation, Shrewsbury Operations, Shrewsbury, NJ	Enero 1, 2001
Telos Federal Systems TFS & Software Eng. Directorate SED	Nov/Dic 1995
TEPCO Systems Corporation	Noviembre 19, 2001
Texas Instruments Defense Systems and Electronics Group	Nov/Dic 1995
The Kosdaq Stock Market Inc. Kosdaq-it Seul, S.Korea	Diciembre 6, 2002
Toshiba IT-Solutions Corporation, Tokyo Operations, 3-22, Katamachi, Fuchu-shi,	Febrero 8, 2002
Tracor, Inc.	Septiembre 3, 1996
TRW	Enero 23, 1997
TYBRIN Corporation	1996
U.S. Army Information Software Development Center, Ft. Lee, Va	Nov/Dic 1995
United Defense LP, Ground Systems Division, San Jose, CA	Diciembre. 14, 1998
Ushus Technologies Private Limited 311, Nila, Technopark Campus, Trivandrum - 695 581, India.	Mayo, 29, 2002
Vitro Corporation (subsidiary of Tracor, Inc.)	Agosto 1996
Wang Southeast Operation's, Montgomery Alabama	Agosto. 8, 1997
Warner Robins Air Logistics Center Avionics Mgmt. Directorate	Nov/Dic 1995
Xerox Corporate Engineering Center, Centro de Desenvolvimento de Sistemas de Vitoria	Febrero 3, 1998

NIVEL CUATRO

ORGANIZACIÓN	FECHA DE PUBLICACION
ALITEC, Laval, Francia	Noviembre 2000
AmSoft Information Services (India) Pvt.Ltd., Software Development and Services Division, Bangalore, India	Diciembre 18, 2002
ANZ Information Technology Pvt Ltd.	Diciembre 27, 2001
AV-8B Joint System Support Activity; Naval Air Systems Command; Naval Air Warfare Center, Weapons Division; China Lake, CA	Septiembre 26, 2002
Axes Technologies (India) Pvt. Ltd. Wireless Switching Division, Bangalore, India	Enero 22, 2003
Bamboo Guangzhou Bamboo Computer Company Limited Guangzhou, China	Enero 10, 2003
BFL Software Limited	Junio 22,1999
Celstream Technologies Private Limited Prestige Blue Chip Software Park, Block II No.9 Hosur Road, Bangalore - 560 029 India	Enero 7, 2003
Citibank Canada, Information Services Division	Marzo 6, 2001
Citibank, Citicorp Overseas Software Ltd. Division	Julio 9, 1997
CITIL (Citicorp Information Technology Industries Limited)	Septiembre 1996
Cognizant Technology Solutions	Diciembre 28, 1998
Computer Science Corporation Federal Sector Civil Group	Febrero 6, 2001
Computer Sciences Corporation India Pvt. LTD., Indore, Madhya Pradesh, India	N/D
Computer Sciences Corporation's Integrated Systems Division	Junio 18, 1998
Covansys Advanced Technology Center, Milpitas, CA	Marzo 1, 2001
Digital GlobalSoft Limited, Bangalore India	Agosto 8, 2001

ORGANIZACIÓN	FECHA DE PUBLICACION
DSQ	Junio 22, 1998
EDS - Electronic Data Systems, India	Noviembre 14, 2002
eFunds International (India) Private Limited; software development centers, Chennai, India	Abril 19, 2002
Electronic Data Systems (EDS), Solutions Consulting Division - Northern Mexico Solution Centre (NMxSC), Cd. Juárez México	Diciembre 19, 2002
FPT (The Corporation for Financing and Promoting Technology), FPT-SOFT Division, Hanoi, Vietnam	Marzo 31, 2002
Fujitsu, Defense Systems Group	Abril 9, 2002
Future Software Limited, India	N/D
Godrej Infotech Ltd., Mumbai India	Noviembre 20, 2001
Gulf Computers Pvt. Ltd.	Septiembre 24, 2001
HCL Perot Systems, A-10/11, Sector 3, Noida India	Agosto. 30, 1999
Hexaware InfoSystems Limited, Mumbai & Chennai Centers	Octubre 14, 1999
Honeywell India Software Operation	Marzo 24, 1997
Hughes Software Systems Limited, New Delhi and Bangalore Centers	Enero 10, 2000
IBM de Mexico, S.A., Application Management Services, Guadalajara and Cd de México	Junio 28,2002
Information Technology Resources, (whole company), Buena Park, CA	Octubre 10, 2002
Infosys Technologies Limited India	Enero 16, 1998
Infotech Enterprises Limited, Hyderabad, India, Software Development Center	Agosto 29, 2001
iS3C Consultancy Services Ltd.	Marzo 2002
Ivega Corporation	Marzo 2002
K G Information Systems Private Limited, KG Campus, 365, Thudiyalur road, Saravanampatty,Coimbatore- 641035 Tamilnadu, INDIA	Agosto 23, 2002
KPIT Infosystems Ltd.	Agosto 8, 2001
Kumaran Systems Private LTD, Chennai, India	Noviembre 15, 2002
Larsen & Toubro Limited, Mysore	Diciembre 26, 2001
LG CNS Supreme Court Project Seul, Korea	Diciembre 6, 2002
Litton Guidance & Control Systems	Diciembre 18, 1998
Lockheed Martin Aeronautics Company, Fort Worth	Diciembre 17, 1999
Lockheed Martin Air Traffic Management	Diciembre 16, 1999
Lockheed Martin Federal Systems, Gaithersburg	Diciembre 1, 1997
Lockheed Martin Information Systems	Julio 11, 2000
Lockheed Martin Management & Data Systems, King of Prussia, PA	Diciembre 16, 1999
Lockheed Martin Naval Electronics & Surveillance Systems Moorestown	Diciembre 13, 1999
Lockheed Martin Ocean, Radar & Sensor Systems	Diciembre 5, 1997
Loral Federal Systems	Nov/Dic 1995
LUXOFT	Marzo 3, 2002
Mascot Systems Ltd., Bangalore, Chennai, and Pune Development Centres	Junio 8, 2001
Mastek Limited, Mumbai	Septiembre 18, 1999
Momentum India Pvt. Ltd., Noida Development Center	Mayo 13, 2002
NCR Corporation, Teradata Research and Development	Octubre 16, 2000
NEC Communication Systems Ltd., First Software Development Department, Second Network Engineering	Octubre 18, 2002

ORGANIZACIÓN	FECHA DE PUBLICACION
Division, Abiko-shi, Chiba, Japón	
Northrop Grumman ESSS Baltimore Operations	Octubre 28, 1999
Northrop Grumman Integrated Systems and Aerostructures	Enero 18, 2000
Oklahoma City Air Logistics Center Directorate of Aircraft Managment Software Division Test Software and Industrial Plant Equipment Branches	Noviembre 22, 1996
Origin Information Technology India Ltd.	Noviembre. 2, 1999
PCCW Business eSolutions (HK) Ltd	Agosto 20, 2002
Pentamedia Graphics Limited	Julio 8, 2001
Phoenix Global Solutions (India) Private Limited, Bangalore, India	Agosto, 14, 2002
Planetasia.com Limited (Planetasia), Bangalore, India	Abril 10, 2002
PSI Data Systems Limited, Bangalore Development Centres, Bangalore India	Marzo 25, 2002
Ramco Systems Limited, Enterprise Solutions SBU - Development Centers 3 and 4, Anna Salai Chennai India	Julio 30, 2002
Siemens Information Systems Limited (SISL), Software Development	Septiembre 16, 2000
Silverline Technologies Limited, Mumbai India and Chennai India	Enero 31, 2000
Siri Technolgies Pvt Ltd Software Development Division, Bangalore India	Enero 03,2003
Sofil Information Systems Pvt.ltd., india	Septiembre 03, 2002
SPAN Systems Corporation, Bangalore Software Solutions for Business Applications	Agosto 1, 2002
SRA Systems Ltd.	Enero 2001
SSI Technologies	Octubre 12, 2000
Tata Consulting Services (TCS), India, SEEPZ Division J	Junio 1999
Tata Elxsi Limited	Agosto 1999
Tata Infotech Limited, SEEPZ,MIDC, SDF-V Centers (Mumbai), India	Octubre 22, 2001
Tata Infotech Ltd. Banglaore & Pune Centers	N/D
Tata Infotech Ltd. NEPZ, Noida	N/D
Tata Interactive Systems, Division of Tata Industries Ltd., Mumbai, India	Julio 5, 2000
TCG Software Services PVT. Ltd.	Enero 9, 2001
TCS-Ahmedabad Branch	Mayo 12, 2000
Telos Corp., Software Engineering Center Fire Support, Lawton Ok	Noviembre 25, 1997
Trigyn Technologies Ltd., Mumbai	Noviembre 6, 2001
U.S. Army Aviation and Missile Command Life Cycle Software Engineering Center	Abril 20, 2000
Wipro Infotech Group Global R&D Division	Marzo 31, 1997

NIVEL CINCO

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Aithent Inc.	Febrero10, 2002
ANZ Information Technology Bangalore - India	Diciembre 21, 2002.
Applitech Solution Limited, Ahmedabad	Mayo 6, 2002
Azeus Systems Limited, Hong Kong and Manila Divisions	Octubre 9, 2002
BAE SYSTEMS Communication, Navigation and Identification (CNI) Division Wayne, New Jersey	Marzo 27, 2002

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Boeing Company, Boeing Information and Communication Systems	Noviembre 1999
Boeing Company, Phantom Works Long Beach	Noviembre 1999
Boeing Company, Reusable Space Systems	Noviembre 1999
Boeing Company, Seal Beach	Enero 1998
Boeing Defense & Space Group	Agosto 1996
CBS India	Diciembre 20, 1999
CG-Smith Software Limited, Bangalore, India	Octubre 9, 1999
CGI Information Systems and Management Consultants Private Ltd.	Marzo 21, 2002
Citicorp Overseas Software Limited, Mumbai, India	Octubre 29, 1999
Cognizant Technology Solutions (Nasdaq: CTSH)	Octubre 10, 2000
Computer Sciences Corporation (CSC); Systems, Engineering and Analysis Support	Diciembre. 2, 1998
Computer Sciences Corporation, Aerospace Information Technologies Business Unit, Fairborn, OH	Marzo 3, 1999
Computer Sciences Corporation, Defense Group	Abril 9, 2001
CSC Computer Sciences UK Division	Agosto 27, 2002
DCM Technologies	Mayo 20, 2000
Eastern Software Systems Ltd.	Mayo 17, 2002
Engineering Analysis Center of Excellence Pvt. Ltd. (EACoE), India	Marzo 5, 2002
Engineering Analysis Center of Excellence Pvt. Ltd., Bangalore, India	Marzo 5, 2002
Future Software Ltd., Chennai	Mayo 22, 2001
General Dynamics Decision Systems, Communications Systems Business Unit, Scottsdale AZ	Noviembre 27, 2001
General Dynamics Land Systems Sterling Heights, MI	Diciembre 18, 2002
GSG Singapore Software Center Motorola	Febrero 22, 2002
HCL Infosystems Ltd., Professional Services Unit - Kolkata, India	Marzo 10, 2003
HCL Perot Systems, India	Febrero. 21, 2000
HCL Technologies (Mumbai) Ltd. Mumbai Division	Septiembre 28, 2002
HCL Technologies Limited India	Julio 2001
Hewlett Packard India Software Operations Limited, Bangalore	Junio 10, 2000
Hewlett Packard, E Solutions Center, E Solutions Division	Febrero 2001
Hexaware Technologies Limited, Chennai and Mumbai Centres	Diciembre 15, 2000
i-flex solutions limited, IT Services Divisions at Mumbai and Bangalore, India	Diciembre 2000
IBM Federal Systems Company	Noviembre 1994
IBM Global Services, India	Noviembre 26, 1999
IBM Japan Ltd. SVC. ITS/SO division AMS Services Department	Noviembre 2001
Infinite Computer Solutions, Bangalore,India	Marzo 6, 2003
Information Technologies (India) Ltd.	Abril 7, 2001
Infosys Technologies Limited, Bangalore	Diciembre 3, 1999
InfoTech Enterprises Limited, Software Division, INDIA	Abril 10, 2002.
InfoTech Enterprises Limited, Software Division, Madhapur, INDIA	Octunbre 4, 2002
Intelligroup Asia PVT. Ltd., Advanced Development Center	Octubre 21, 2000
Intergraph Consulting Pvt. Ltd., Software development and	Julio 4, 2002

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
services business across all product lines	
International Computers (India)	Febrero 1999
IT Solutions (India) Private Limited, Bangalore and Chennai Development Centers	Noviembre 24, 2001
ITC Infotech India LTD	N/D
L&T Infotech For all offerings at Powai, Vashi, Pune, Chennai,Bangalore & Mysore	Noviembre 14, 2002
L-3 Communications, Integrated Systems, L.P. Greenville, Texas	Noviembre 8, 2002
Larsen & Turbo Infotech Limited, Europe Sbu at Mumbai and Navi Mumbai Centers	Marzo 21, 2002
Larsen and Toubro Infotech Limited	Noviembre 14, 2002
LG Soft India Pvt. Ltd., Bangalore, India	Octubre 30, 2001
Litton PRC Inc.	Abril 3, 2000
Lockheed Martin Federal Systems, Owego	Diciembre 22, 1997
Lockheed Martin Management & Data Systems, King of Prussia	Diciembre 4, 2000
Lockheed Martin Mission Systems, Gaithersburg, MD.	Octubre 14, 1999
Lockheed Martin Ocean, Radar & Sensor Systems, Syracuse, NY	Noviembre. 15, 1999
Lockheed Martin Undersea Systems, Manassas, VA	Marzo 17 1999
Majoris Systems pvt. Ltd, Karnataka, India	Diciembre 16,2002
Mascot Systems Ltd., Bangalore & Chennai Development Centres	Diciembre 23, 2002
Mastek Ltd., Mumbai	N/D
Merrill Lynch, .S. Private Client division Jacksonville, FL	Agosto 23, 2002
Motorola - Global Software Group (GSG) Canada	Enero 24, 2002
Motorola Global Software Group (GSG) - Krakow Software Center, Poland	Febrero 15, 2002
Motorola India Electronic Ltd. (MIEL)	Marzo 1994
Motorola Information Security Systems and Products, ISSPD Division	Enero 19, 2001
Motorola, (NYSE: MOT)Integrated Systems Division, Scottsdale, AZ	Junio 22, 2000
Motorola, GSG St. Petersburg Software Development	Octubre 23, 2001
Motorola, Malaysia Software Center	Diciembre 3, 2001
MphasiS-BFL Limited	Diciembre. 31, 2000
Nasa; Onboard Shuttle Group	Diciembre 1996
NeST Information Technologies (P) Ltd	Noviembre 8, 2001
Network Systems & Technologies (P) Ltd., Trivandrum	Mayo 13, 2000
NIIT, Knowledge Solutions Business (KSB)	Julio 16, 2001
NIIT, Software Solutions	Septiembre 19, 1999
Ogden Air Logistics Center Software Engineering Division at Hill AFB	Agosto. 3, 1998
Patni Computer Systems Ltd	Agosto 31, 2000
Philips Software Centre Private Limited, Philips Innovation Campus	Julio 26, 2000
Productora de Software S.A - PSL - Medellín, Colombia, South America	Diciembre 13, 2002
Quinnox Consultancy Services Ltd. Mumbai, India.	Diciembre 28 2002
Raytheon Missle Systems	Diciembre 5, 2001
Raytheon Systems Company's Command and Control Division, Fullerton, CA	Enero 1999

ORGANIZACIÓN	FECHA DE PUBLICACIÓN
Sasken Communication Technologies Limited., Bangalore, India	Abril 23, 1999
Siemens Information Systems Ltd., Telecom & Major Projects Division	Septiembre 22, 2001
SignalTree Solutions (India) Ltd	Abril 25, 2001
SkyTECH Solutions Pvt Ltd., (Kolkata and Mumbai, India offices) Application Development, Maintenance and Support for Client Server, Web Based, Unisys Mainframe and IBM Mainframe Technology Divisions	Abril 15, 2002
Sonata Software Limited, Bangalore, India	Septiembre 24, 2001
SSI Technologies, Chennai, India	Agosto 8, 2001
Stabilix Solutions Private Limited, Kerala, INDIA	Febrero 15, 2003
STMicroelctronics India Ltd., Corporate Software Fab Noida, India	Diciembre 05, 2002
Syntel	Agosto 8, 2002
Syntel, Inc. (India)	Julio 24, 2001
Tata Consultancy Services, Gurgaon II Centre, Delhi	Febrero 19, 2001
Tata Consultancy Services, NOIDA Centre, Delhi	Marzo 2001
Tata Consultancy Services, TCS U S West	Mayo 27, 1999
Tata Consultancy Services, TCS&HP	Julio 1999
Tata Consultancy Services, TCS, SEEPZ	Agosto 1999
Tata Elxsi Limited	Junio 2000
Tata Interactive Systems, Mohali, India	Octubre 24, 2001
Tata Interactive Systems, Mumbai, India	Octubre 24, 2001
TCG Software Services Pvt. Ltd., Calcutta, India	Junio 3, 2002
TCS - Pune INDIA	Abril 19, 2002
TCS Ahmedabad	Noviembre 10, 2000
TCS Ambattur	Julio 27, 2000
TCS Bangalore	Enero 30, 2000
TCS Calcutta	Enero 22, 2000
TCS GEDC	Agosto 2, 2000
TCS Hyderabad	Mayo 27, 2000
TCS Sholiganallur	Noviembre 17, 1999
TCS-Lucknow	Enero 28, 2000
Telcordia Technologies, Inc.	Mayo 25, 1999
United Defense LP, Ground Systems Division, Santa Clara, CA	Enero 24, 2003
US Technology Resources India, US Software PVT Ltd., Trivandrum, India	Diciembre 20, 2001
Wipro Infotech, Enterprise Solutions Division	Enero 22, 1999

El estatus en México

Aunque no se tienen datos precisos, se dice que en México únicamente cuatro empresas tienen nivel CMM 2 o 3.
A Enero del 2002, de acuerdo a información del SEI, existen 139 organizaciones en el ámbito mundial con un alto nivel de madurez, 73 Nivel 4 y 66 Nivel 5. Ochenta (80) de estas organizaciones se encuentran fuera de Estados Unidos. India tiene 28 organizaciones Nivel 4 y 43 organizaciones Nivel 5.

Recientemente se publico que IBM y Sofftek han alcanzado el nivel cinco, otros ejemplos incluyen a Palacio de Hierro en nivel dos y Banamex en nivel tres. Así como también hay datos en internet de que 18 empresas sinaloenses están participando en un proceso de capacitación para obtener una evaluación en *calidad de Madurez (CMM-(Capability Maturity Model)) nivel 3.*

PUNTOS SOBRESALIENTES DEL CAPITULO

- El CMM tiene tres objetivos principales:
  - ✓ Construir un entendimiento sobre el proceso de software.
  - ✓ Generar mediciones del proceso, definiendo escalas para medir la eficiencia de este.
  - ✓ Servir como una base para la mejora del proceso.

- Las principales ventajas del CMM son:
  - ✓ Mejora de la comunicación interna.
  - ✓ Disminuye el trabajo, problemas de integración, rango de errores y costos de pruebas.
  - ✓ Los desarrollos no sobrepasan presupuestos ni tiempos de entrega.
  - ✓ Mejora la moral entre los empleados y hay menos presión en el equipo de trabajo y menor tiempo extra.

- Las principales desventajas del CMM son:
  - ✓ No hay muchas organizaciones por encima del nivel uno.
  - ✓ Fue diseñado para organizaciones con grandes contratos para el gobierno, y aunque es aplicable a compañías pequeñas, se necesita hacer una interpretación extensiva para ellas.
  - ✓ Considera que la gente no es fiable y asumen que un proceso definido puede reducir las fallas.
  - ✓ En el modelo no esta contemplada la innovación.
  - ✓ Desconfía de las contribuciones individuales.
  - ✓ Sé necesita capital para apoyar el esfuerzo.
  - ✓ Puede incrementarse el tiempo de entrenamiento y la disminución de tiempo disponible para trabajar en los proyectos.
  - ✓ Para ir de un nivel a otro, se requiere una extensa capacitación para que la organización entienda los procesos.

- Al aplicar este modelo se debe tener cuidado con lo siguiente:
  - ✓ Tener expectativas muy altas.
  - ✓ Esperar que todas las áreas de proceso inicien en el nivel dos.
  - ✓ Esperar un avance de nivel por año.
  - ✓ Esperar que la mejora ocurra por sí misma.
  - ✓ No saltarse niveles.
  - ✓ No incluir tecnología nueva en las primeras etapas.

CONCLUSIONES

- Uno de los campos laborales más importante para de los egresados de Matemáticas Aplicadas y Computación, es el desarrollo y/o mantenimiento de sistemas, campo profesional que actualmente tiene una mala imagen asociada a problemas tales como costos elevados, retrasos en tiempos de entrega y funcionalidades no solicitadas o defectuosas; razón por la cual es importante que todos aquellos interesados en este campo profesional conozcan al menos un modelo de ingeniería de software de manera general. Algunas de las prácticas tales como la administración de requerimientos, planeación de proyectos, revisión de entregables y programas de entrenamiento, deberían ser obligatorias para cualquier organización o proyecto, pues son pasos básicos para un desarrollo de software exitoso.

- En un medio tan cambiante como es la industria del software, es importante contar con guías para la administración de proyectos, para lograr esto, existen modelos que brindan ideas que deben ser tomadas en cuenta, pero considero que no se deben seguir rigurosamente, pues además del gasto que pueden implicar tanto de personal como económico, puede no conducir a nada si no hay un cambio de cultura.

- El SEI-CMM es un modelo de ingeniería de software que surge como apoyo para la administración de proyectos, bajo la premisa de que documentar los problemas y restricciones que se presentaron durante un proyecto ayuda a obtener estimados realistas en un futuro.

- El SEI ha desarrollado el CMM para ser usado como un conjunto de criterios para aumentar la madurez de los procesos de desarrollo de software de una organización, considerando que un proceso maduro lleva a productos entregados a tiempo, dentro del presupuesto, que cumplen los requerimientos solicitados y tienen una alta calidad.

- El CMM es un modelo de ingeniería de software enfocado principalmente a brindar guías para un mayor control sobre el proceso de desarrollo de software, tiene puntos útiles para cualquier organización o proyecto, tales como la recopilación y registro de la información que puede ser utilizada como base para la estimación de costos y tiempos en proyectos posteriores con características similares a fin de obtener mejores estimados sobre los resultados que surgirán, pero también tiene algunas prácticas que no son muy comunes y que pueden resultar burocráticas y costosas tales como la administración cualitativa, prevención de defectos o la administración de cambios, así como la constante documentación de todos los pasos del proceso y las adaptaciones de este a los proyectos ocasionan que el modelo se vuelva repetitivo y tedioso.

- El utilizar el CMM ayuda a identificar los puntos fuertes y débiles en la administración de proyectos, así como también es útil para definir el proceso de desarrollo de software que será utilizado, pues para los recién llegados es más fácil integrarse si les brindan guías, documentación y estándares para dar resultados o apoyar en la implantación de un programa de mejora en el proceso de desarrollo.

- Este modelo requiere:

1. Un compromiso a largo plazo, porque no se trata de un cambio tecnológico, sino un cambio de cultura, lo que implica un gran esfuerzo que no siempre es factible debido a la naturaleza cambiante y competitiva de la industria del software.

2. Un cambio cultural para formalizar el registro de las actividades, la explicación para llevar a cabo este registro está descrito en un gran número de páginas, por lo que es considerado pesado y riguroso, pues aunque no dice que metodología usar para cumplir cada nivel, si pide que todas las áreas claves de proceso sean cumplidas para poder avanzar al siguiente nivel, por lo que algunas de las tareas que deben ejecutarse pueden no ser cumplidas debido a la falta de presupuesto, personal o compromiso de los involucrados.

- El CMM se basa en la premisa de que los mayores problemas de desarrollo de software, y por ende, las causas de las fallas del proyecto son más administrativas que técnicas.

- EL CMM no involucra prácticas de administración e ingeniería importantes para el éxito del proyecto, por ejemplo, no cubre la experiencia en el manejo de una aplicación en particular, métodos o tecnología de software o asuntos relacionados con recursos humanos (tales como seleccionar, contratar o motivar gente competente).

- Es importante estar consciente de que para que una empresa alcance niveles mas altos de madurez es necesario un compromiso a largo plazo. A las organizaciones les toma años construir el fundamento y la cultura necesarias para una mejora continua del proceso, así como un alto nivel de esfuerzo, por lo que en un medio tan cambiante y competitivo como lo es la industria del software, esto no siempre es posible debido a las demandas del mercado.

- Existen muchos ejemplos de desastres causados por problemas de software. A medida que nuestra sociedad siga dependiendo de las computadoras, los riesgos debidos a la baja calidad del código serán peligrosos. Mientras que pueden cuestionarse, los tamaños y complejidades de los sistemas actuales, estos seguirán siendo productos de seres humanos, con todas sus fallas y talentos creativos. A menos que mejoremos nuestros rangos de error, un mayor volumen de código significará un incrementado riesgo de falla.

Bibliografía

***[1]* The Capability Maturity Model (Guidelines for Improving the Software Process)**
Carnagie Mellon University
Software Engineering Institute
Addison Wesley Publishing, 1995

[2] Patterns of Software Systems Failure and Success
Capers Jones
Thomsons Computer Press, 1996

[3] Death March (The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects)
Edward Yourdon
Prentince Hall, 1997

[4] Decline and Fall of the American Programmer
Edward Yourdon
Prentince Hall, 1992

[5] Software Engineering (The Production of Quality Software)
Shari Lawrence Pfleeger
MacMillan Publishing Company, 1987

[6] Managing the Software Process
Watts S. Humphrey
SEI Series in Software Engineering, 1989

[7] Creating a Software Engineering Culture
Karl E. Wiegers
Dorset House Publishing, 1996

[8] Software Engineering Notes Volumen 15, No 6
Sigsoft
Acm Press, 1990

[9] The great transition (Using the seven disciplines of Entreprise Engineering to aling people, technology and strategy)
James Martin
Amacom, 1995

[10] Assessment & Control of Software Risks
Capers Jones
UN Miller Freeman Inc.,1996

[11] Disponible en Internet en
http://cispom.boisestate.edu/cis320emaxson/cmmht1.htm

[12] Disponible en Internet en http://seir.sei.cmu.edu/pml/

[13] Disponible en Internet en http://www.satisfice.com/articles/cmm.htm

[14] Disponible en Internet en
http://www.cis.njit.edu/axp9532/cmm/cmm_iso_compare.html

[15] Comparing Iso 9000, Malcom Baldrige And Sei CMM For Software
Michael o. Tingeny
Prentice Hall,1997

ANEXOS

I. Modelo de Inmadurez y Eficiencia

El siguiente artículo es el que inspiró el presente trabajo.

El CMM (Capability and Maturity Model o Modelo de eficiencia y madurez) proporciona a las organizaciones de software una guía de como controlar sus procesos de desarrollo y mantenimiento de software y como avanzar hacia una cultura de excelencia. De acuerdo al SEI (Software Engineering Institute), mas del 70% de las organizaciones de software están en el nivel 1. Esta información no es adecuada, ya que muchas de ella están por debajo de lo meramente caótico. Todas estas organizaciones son colocadas a nivel 1, ya que simplemente no existen niveles inferiores.

A continuación se describirán los niveles de madurez menores, propuestos por el Dr. Sami Zahran de Bull Information Systems con sus actitudes contra producentes (KPA o Kounter Productive Attitudes), no olvidemos que en el CMM KPA hace referencia a las Key Process Areas o áreas claves de proceso que dan la definición de cada nivel de madurez.

0	Negligente	Indiferencia	Falla en permitir el desarrollo del proceso. Todos los problemas son percibidos como técnicos
-1	Obstructivo	Contra productivo	Los procesos son definidos de manera rígida y el seguimiento de reglas es forzoso
-2	Desdeñoso	Arrogante	Carencia completa de un programa de entrenamiento
-3	Desalentador	Sabotaje	Se premia a la falla y a la baja funcionalidad

Nivel 0. Negligente
Las organizaciones que están en este nivel actúan de tal manera que previenen los esfuerzos heroicos de dar algún fruto. La apatía, indiferencia y desorganización son las KPA de este nivel. Cuando las especificaciones y documentación son entregadas, se guardarán y no volverán a ser usadas. Cuando un producto esté casi terminado, la organización cambiará o añadirá requerimientos para asegurar la falla del proyecto.

Mientras que actitudes negligentes existen en toda la gente en un nivel 0, esta es mas evidente en la administración. Todas aquellas organizaciones inmaduras (por debajo del nivel 1) no reconocen que la administración es importante y creen firmemente que los problemas técnicos causan la baja calidad del software y los retrasos en tiempo. Todas las decisiones técnicas que se toman son hechas para casos aislados sin metas a largo plazo.

Los gerentes en el nivel 0 se refugian en lugares comunes para salvarse de si mismos. Estos incluyen herramientas Case, arquitecturas Open Systems o Cliente Servidor, Métricas de reuso, Mejoras al proceso de negocio, CMM, SPICE o cualquier 'camino de mejora' que este disponible. Estos excelentes esfuerzos de mejora nunca ofrecen mucha diferencia en las organizaciones de nivel 0, ya que aunque cada solución es introducida con gran fanfarria es inevitablemente sobre pasada por la apatía y falta de compromiso de la gerencia.

Adicionalmente a esta apatía, las organizaciones negligentes rara vez tienen visión o metas organizacionales.

Nivel -1 Obstructivo
Las organizaciones en el nivel obstructivo, van mas allá de la simple negligencia a través de revertir las actividades de desarrollo. Las KPAs en el nivel -1 incluyen una actitud demasiado rígida y formal. Insisten en procesos complejos y el uso de lenguajes y equipos de cómputo arcaicos. Los procesos no tienen responsables y no existe un método para cambiarlos, de hecho los cambios son desalentados.

Las organizaciones de nivel -1 por lo general usan administración colectiva para supervisar el desarrollo de software. La administración colectiva es el proceso de dividir esfuerzos complejos de software en piezas y asignarles a cada una su propio manager, haciendo a un lado la administración global del proyecto, de tal manera que la administración global es hecha por los managers en las fases individuales. A medida que pasa el tiempo, estas fases tienden a crecer en tamaño y duración, requiriendo mas y mas sub administradores, mientras que al mismo tiempo se generan brechas en el proceso de desarrollo y se sobreponen por esfuerzos duplicados.

Ya que la responsabilidad del producto final es también colectiva, este tipo de mantenimiento es muy útil para esconder las raíces de los problemas distribuyéndolos a través de varios encargados y varias fases, lo que le permite a cada manager culpar a los demás.

Muchas actividades son desarrolladas y se entregan con una puntualidad rígida. Pero los propósitos de estas no están documentados y a veces la razón de dichas actividades es olvidada. Los controles de calidad en este nivel no checan la calidad de la actividad o el contenido de la documentación. Hay muchas actividades requeridas y tantos documentos que revisar que es difícil juzgar la calidad de las actividades y documentos cuyo propósito no está definido. En ellas generalmente se hace doble trabajo porque se ejecutan actividades pre definidas -ceremoniales- en paralelo con las actividades de desarrollo que producen el resultado final.

Nivel -2 Desdeñoso
Las organizaciones en el nivel -1 creen sinceramente que están desarrollando software siguiendo buenas prácticas a pesar de la clara evidencia de lo contrario.

En contraste, las organizaciones de nivel -2 son abiertamente desdeñosas de las prácticas de ingeniería de software. Las KPAs en el nivel -2 incluyen un completo descuido y rechazo de cualquier esfuerzo para mejorar la forma en que se desarrolla un proyecto. Muestran su desacuerdo a las actividades de mejora por su falta de un programa de entrenamiento. Estas organizaciones no dan entrenamiento porque a)no hay presupuesto, b)no hay tiempo y c)es una pérdida de tiempo de cualquier forma.

Toda la gente nueva se espera que conozca el trabajo, o sea entrenado por la persona que dejo el trabajo 2 meses antes de que ellos ingresaran en el trabajo. Si se contratan ingenieros de software, se les critica por su conocimiento teórico porque no tienen experiencia real de desarrollo; si nuevas contrataciones tienen experiencia en desarrollo, son criticados por tener experiencia en desarrollo y no en mantenimiento, y si tienen ambas se les dice que el sistema, la organización y el usuario son diferentes y que esas ideas de ingeniería de software no funcionan en ese ambiente.

Un manual de desarrollo de software inevitablemente existe en este tipo de organizaciones, pero solo es sacado de su lugar para mostrarse a un grupo de revisión para probarles que existe un método para desarrollar software dentro de la organización. Mucha gente, dentro de la organización sabe de su existencia y hace referencia a él, pero nunca lo han leído.

Los gerentes en este nivel insisten en que solo tienen tiempo para desarrollar software, no para mejorar el proceso. Los desarrolladores creen que mejorar como se hace esta labor no es su trabajo, ellos solo tienen tiempo para apoyar la misión. Los grupos de mejora no realizan su misión debido a que su consejo es solo académico y no participan en el desarrollo. Cualquier esfuerzo de unir a estos 2 grupos, es detenido antes de que pueda llevarse a cabo.

Nivel -3 Desalentador

No contentos con arruinar su propio proceso de software, una organización en nivel -3 busca de manera activa desacreditar y destrozar el trabajo de otras organizaciones. Donde el trabajo de otra organización no pueda ser desacreditado, estas organizaciones clamaran crédito del trabajo, tanto como les sea posible. Estas organizaciones ignoran sus propios procesos en favor de desarrollar publicidad positiva para si mismos, básicamente enfocándose a crear una agradable piel roja alrededor de una manzana que tal vez este podrida en su interior.

Las KPAs asociadas a este nivel incluyen el creer que lucir bien es mas importante que estar bien, que cualquier atención es mejor que ninguna y que mientras entre dinero en ellas, son exitosas. A una organización saboteadora no le interesa si producen software de baja calidad mientras este seguro el trabajo y garantice mas dinero para mantener el sistema a través de toda su vida.

Estas organizaciones tratan de asegurarse de que todas las otras organizaciones están peor que ellas. Ya que es mas fácil destruir que construir, estas organizaciones lo llevan a cabo a través de comprometer y dañar a sus competidores. Después de todo, la mejor defensa es una buena ofensa. Cualquier debilidad en el contrario es voluntaria y metódicamente explotada.

Las organizaciones en el nivel -3 glorifican la falla y la baja funcionalidad. Dados 2 esfuerzos idénticos de desarrollo de software, el que está a tiempo, no sobre pasa el presupuesto y complace al usuario final es ignorado, mientras que el que está retrasado, cargado de problemas y renuentemente aceptado por el usuario, es alabado por producir resultados después de todo, ya que a todos les gusta la historia de como el hombre promedio supera la adversidad. Esta actitud de incentivos reversos motiva la anarquía. El tiempo destinado a mejorar la forma en que las cosas son hechas y alcanzar resultados, es tratado con un valor mucho menor que promover la fachada de que el esfuerzo de desarrollo es exitoso.

II. PEOPLE CMM

Debido a que el SW-CMM no contempla ninguna acción para entrenar y desarrollar las habilidades de las personas que intervienen en el proceso de software, este artículo resulta interesante debido a que es un resumen del esfuerzo a atender dicha carencia.

Este modelo adapta el esquema de madurez del CMM para las personas, a fin de administrar y desarrollar la fuerza de trabajo de la organización. La motivación para el P-CMM es mejorar radicalmente la habilidad de las organizaciones de software para atraer, desarrollar, motivar, organizar y retener el talento necesario para mejorar continuamente la eficiencia del desarrollo. El P-CMM está diseñado para permitir a las organizaciones integrar mejoras en la fuerza de trabajo con los programas de mejoras de proceso siguiendo el CMM. El P-CMM puede ser también utilizado por cualquier tipo de organización como una guía para mejorar sus prácticas relativas a la fuerza de trabajo.

Basado en las mejores prácticas de campos tales como recursos humanos y desarrollo organizacional, el P-CMM brinda a las organizaciones guías de cómo obtener control sobre sus procesos para administrar y desarrollar su fuerza de trabajo. Ayuda a definir características de madurez en las prácticas relativas a la fuerza de trabajo, apoyar un programa de desarrollo continuo, establecer prioridades para acciones inmediatas y una cultura de excelencia en ingeniería de software.

Describe un camino de mejora evolutiva desde prácticas inconsistentes y a la medida, hacia un desarrollo disciplinado y maduro del conocimiento, habilidades y motivación de la fuerza de trabajo.

El P-CMM consiste de 5 niveles de madurez los cuales tienen fundamentos sucesivos para mejorar continuamente el talento, desarrollar equipos eficientes y administrar exitosamente al personal de la organización. Cada nivel de madurez es una plataforma bien definida que institucionaliza un nivel de eficiencia para el desarrollo de talento dentro de la organización.

Excepto por el nivel 1, cada nivel de madurez está compuesto por varias áreas claves de proceso, para indicar las áreas en las cuales la organización debe enfocarse para mejorar su eficiencia en la fuerza de trabajo. Cada área de proceso es descrita en prácticas claves que contribuyen a alcanzar las metas. Las prácticas claves, describen una infraestructura y actividades que contribuyen mas a la implantación e institucionalización efectiva de las áreas claves de proceso.

Los 5 niveles de madurez del P-CMM son:

1) Inicial.

2) Repetible. Las áreas clave de proceso en el nivel 2 se enfocan a inculcar disciplinas básicas dentro de las actividades de la fuerza de trabajo. Las cuales son:

- Ambiente de trabajo
- Comunicación
- Personal
- Administración de la funcionalidad
- Capacitación
- Compensación

3) Definido. Las áreas claves de proceso para el nivel 3, se enfoca en los tópicos relacionados con la identificación de las competencias básicas y alienta las actividades de administración dentro de ellas. Estas son:

- Análisis de conocimientos y habilidades
- Planeación de fuerza de trabajo
- Desarrollo de competencias
- Desarrollo de carrera
- Prácticas basadas en las competencias
- Cultura de participación

4) Administrado Las áreas claves de proceso en el nivel 4 se enfoca a la administración cuantitativa del crecimiento organizacional de las habilidades de la gente y establecer equipos basados en competencias. Estas son:

- Asesoramiento
- Construcción de equipos
- Prácticas basadas en equipos
- Administración de la competencia Organizacional
- Consenso organizacional del desempeño

5) Optimizado. Las áreas claves del nivel 5 cubren los tópicos relacionados con los métodos de mejora continua para el desarrollo de competencias, tanto en el nivel organizacional como individual. Estas son:

- Desarrollo de las competencias personales
- Aleccionamiento
- Innovación continua de la fuerza de trabajo

III Comparación del CMM e ISO 9000

A continuación se presenta una comparación entre el CMM y el modelo mas popular de ingeniería de software.

1. Iniciativas, objetivos y ámbito

En general, el CMM e ISO 9000 están dirigidos a objetivos similares y tienen un interés común en la calidad y la administración de procesos.

ISO

- ✓ Está enfocado principalmente a la relación cliente - proveedor a fin de reducir el riesgo de elegir un proveedor.

CMM

- ✓ Está enfocado a que el proveedor mejore su proceso de software.

2. Objetivo

ISO

- ✓ Fue escrito para un amplio ramo de industrias además de software.
- ✓ Sus documentos son más abstractos.
- ✓ ISO 9001 tiene solo 5 páginas e ISO 9000-3 tiene 11 páginas.
- ✓ Identifica los requerimientos mínimos para un sistema de calidad.

CMM

- ✓ Es exclusivo de la industria de software.
- ✓ Tiene gran detalle en su documentación.
- ✓ Tiene mas de 500 páginas.
- ✓ Describe el proceso de software de manera detallada.

3. Desarrollo de productos

Ambos modelos soportan:

1) Definición y formalización de procesos.
2) Evaluaciones estandarizadas y objetivas de terceros.
3) Auto medición.

ISO

- ✓ Tiene un ámbito amplio que abarca hardware, software, materiales y servicios.

CMM

- ✓ Es específico a la industria del desarrollo de software.

4. Concepto

ISO

- ✓ Es seguir un conjunto de estándares para hacer que un éxito sea repetible.

CMM

- ✓ Enfatiza el alcance de madurez y mejorar el proceso continuamente.

5. Estructura

ISO

- ✓ Significa que algunas prácticas básicas son llevadas a cabo y el reto es solo mantener la certificación.

CMM

- ✓ Enfatiza la mejora continua, incluso en el último nivel.

6. Mediciones, evaluaciones, auditorias y certificación

En esencia, las evaluaciones del CMM tienen el mismo objetivo que las auditorias de ISO.

Ambos han sido desarrollados para evaluar la eficiencia de una organización de software para producir este de una manera repetible y a tiempo.

ISO

- ✓ Durante una auditoria, en una organización se revisa que esta siga un cierto conjunto de estándares.
- ✓ El modelo requiere auditores, de tal forma que el valor de la certificación depende de la experiencia y conocimiento de los auditores

CMM

- ✓ En una evaluación la organización es catalogada de acuerdo a 5 niveles.
- ✓ Una organización llevó a cabo una automedición tipo CMM después de una auditoria ISO y encontró que los auditores erróneamente habían percibido que ciertas prácticas se llevaban a cabo.
- ✓ Medición Interna.
- ✓ Permite llevar a cabo automediciones.

7. Estado de la industria del software

ISO

- ✓ No se puede determinar tal información debido a que las compañías están o no certificadas.
- ✓ Una conclusión geográfica indica que Europa tiene el mayor número de compañías certificadas.

CMM

- ✓ Desde la información estadística del CMM, la industria del software aún requiere mucha mejora
- ✓ Muchas compañías están en el nivel 1 y muy pocas en los niveles 4 y 5.

8. Tiempo necesario

ISO

- ✓ Toma de 1 a 1.5 años obtener la certificación ISO.
- ✓ Su objetivo es una mejora general.

CMM

- ✓ Toma en promedio 2 años ir de un nivel a otro.
- ✓ Su objetivo es implantar una base fuerte para el desarrollo de software.

9. Beneficios

Estos beneficios muchas veces, están acompañados de grandes números. Sin embargo, debe notarse que las compañías se sientan mas cómodas reportando

éxitos que fracasos y estos números algunas veces son relativos debido a como hayan sido calculados.

Los beneficios comunes son:

- ✓ Un cambio cultural positivo.
- ✓ Incremento de la productividad.
- ✓ Mejor comunicación.
- ✓ Mejor satisfacción del cliente.

10. Contraste

Debe notarse que el contraste y la comparación de estos modelos tiende a ser subjetiva en algunos casos

ISO

- ✓ Tiene libertad de interpretación, algunos argumentan que si uno lee el ISO 9000, este cubre material del nivel 1 al nivel 3 del CMM.

CMM

- ✓ Las organizaciones de software al tener una certificación con la finalidad de ser certificadas, siguen en el nivel 1 del CMM.[14]

La calificación ISO 9000 es llevada a cabo a través de un proceso de registro dentro de uno de los 3 estándares (9001, 9002 o 9003). El proceso de certificación consiste de 4 pasos principales:

1. Pre auditoria - Opcional-
2. Auditoria -Aprobada o No aprobada-
3. Registro
4. Auditoria de vigilancia -Semi anual/ continua-

El proceso de certificación inicia con una preauditoría que prepara a la compañía para la auditoria. Esta es seguida por una auditoria detalla del sistema de administración de la calidad por parte de una auditor externo perteneciente a agencias independientes que están acreditadas por organizaciones internacionales para llevar a cabo tales auditorias, las cuales verifican el cumplimiento de los estándares y dependiendo del tamaño de la organización puede durar varias semanas. Si no se encuentran grandes fallas en las auditorias, se recomienda a la compañía para su registro. Si se encuentran fallas, la compañía debe dar un plan de acciones correctivas al auditor. El último paso del proceso es un monitoreo constante y la verificación de la administración de la calidad. Las auditorias de vigilancia son llevadas a cabo por auditores externos, usualmente 2 veces al año.[15]

Para la medición del CMM se usa el siguiente proceso:

1. Selección del equipo evaluador
2. Cuestionarios de madurez
3. Visitas
4. Hallazgos

Primero se selecciona un equipo que ha sido entrenado en los conceptos del CMM, así como se selecciona un método de medición. Después, representantes

de la organización a ser evaluada llenan un cuestionario de madurez. El equipo evaluador analiza las respuestas y de este análisis se identifican aquellas áreas del CMM que requieren mas investigación. Las visitas generalmente requieren un periodo de 5 días, y consisten de entrevistas y revisión de documentos para determinar si las áreas claves de proceso han sido satisfechas. Los resultados de estas visitas son documentados en hallazgos que identifican las fortalezas y debilidades del proceso de la organización.[15]

IV. Propuesta para la ocupación del modelo

A continuación se presenta una propuesta de la utilización del modelo y recomendaciones de cómo el presente trabajo puede apoyar a las labores académicas de la M.A.C.

Nivel 2

Administración de requerimientos.
Establecer por escrito cuales son los requerimientos que serán cubiertos por el proyecto, una vez hecho esto obtener las firmas de los involucrados.

Planeación del proyecto de software
Establecido el alcance del proyecto, solicitar los recursos necesarios para cubrirlos en el tiempo requerido, si los recursos son limitados, determinar los tiempos para cada entregable.

Seguimiento del proyecto.
Ya establecido el plan, de acuerdo a los tiempos establecidos de cada entregable, se deberá dar un estatus del avance del proyecto a todos los involucrados.

Administración de contratantes secundarios de software
Apegar a los contratantes al método definido para determinación de alcance y entregables.

Medición de la calidad del software
Enviar encuestas a los usuarios sobre los entregables, para construir la encuesta se usará como base lo definido dentro del alcance del proyecto.

Administración de la configuración del software
Determinar que productos de software serán usados dentro del proyecto y definir sus requerimientos de uso y configuración con el apoyo de personal experto en ellos.

Nivel 3

Enfoque organizacional del proceso.
Definir un grupo -o persona-, encargado de identificar las mejores prácticas para integrarlas al proceso de desarrollo.

Definición del proceso organizacional.
Una vez determinadas las mejores prácticas, se deberán establecer como un proceso que se debe seguir para el desarrollo, el cual debe ser documentado y encontrarse a disposición de todo el personal de la organización.

Programa de entrenamiento.

Establecer planes de capacitación para dar a conocer este proceso para aquellos que se verán directamente involucrados con él.
Administración de la integración del software
Para cada proyecto, sustituir o eliminar aquellas actividades del proceso, que no son aplicables al proyecto, por ejemplo, si no es necesario contratar personal externo, eliminar esta actividad.

Ingeniería de productos de software
Establecer una metodología de análisis y diseño de software, apoyada en el uso de herramientas para ello de acuerdo a cada uno de los proyectos.

Coordinación intergrupal.
Definir vías de comunicación, responsabilidades y responsables de actividades en cada uno de los grupos involucrados en el proceso de desarrollo y darlas a conocer a aquellos involucrados en proyectos que requieran apoyo de otros grupos dentro de la organización.

Revisiones a fondo.
Establecer criterios de revisión, para los productos de software que serán entregados al usuario final, estos pueden ser revisión de código por una persona que también este involucrada en el proyecto. Definición de casos de prueba con el usuario. Establecimiento de un grupo de control de calidad, etc.

Nivel 4
Administración cuantitativa del proceso.
Definir criterios de éxito en base a ahorro en tiempo y recursos, funcionalidad entregada y número de transacciones llevadas a cabo por el software para posteriormente comparar estas cifras con las de proyectos anteriores con características similares.

Administración de la calidad del software
Medir y documentar el número de fallas encontradas en las revisiones a fondo, así como registrar el número de fallas en producción y correcciones hechas al producto entregado para posteriormente comparar estas cifras con las de proyectos anteriores con características similares.

Nivel 5
Prevención de defectos.
Documentar las causas de defectos en los productos entregados y ponerlos a disposición de todos los involucrados en los procesos de desarrollo de software dentro de la organización.

Administración de los cambios tecnológicos.
Establecer un grupo para evaluar nuevas herramientas de apoyo en el desarrollo de software para incorporar nuevas tecnologías e irlas incorporando en las fases iniciales de los desarrollos,

Administración de cambios del proceso.
Evaluar e incorporar todas aquellas sugerencias para la mejora del proceso, a través de las lecciones aprendidas de los proyectos.

Actualmente la carrera de Matemáticas Aplicadas y Computación, cuenta en su plan de estudios con la materia de Ingeniería de software, el presente trabajo puede servir como material de apoyo en la última unidad que es el ANALISIS DE LAS DIFERENTES TECNOLOGIAS EN LA INGENIERIA DE SOFTWARE, en el apartado 4 que es Calidad del software, así como también puede apoyar en la materia de PLANEACION DE PROYECTOS, a través de contemplar algunos de los pasos sugeridos por el modelo, tales como control de los cambios tecnológicos, configuración del software, etc., a fin de que sus conocimientos brinden productos de mejor calidad.