Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

MODELO DE CAPACIDAD DE MADUREZ DEL *SOFTWARE* Y SU INFLUENCIA EN LAS MEJORAS DE CALIDAD DEL *SOFTWARE*

Whuendy Yasmina Chacón Tarot

Asesorado por Ing. José Ricardo Morales Prado

Guatemala, marzo de 2004

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

MODELO DE CAPACIDAD DE MADUREZ DEL *SOFTWARE* Y SU INFLUENCIA EN LAS MEJORAS DE CALIDAD DEL *SOFTWARE*

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA

POR

WHUENDY YASMINA CHACÓN TAROT

ASESORADO POR ING. JOSÉ RICARDO
MORALES PRADO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MARZO DE 2004

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Sydney Alexander Samuels Milson
VOCAL I	Ing. Murphy Olympo Paiz Recinos
VOCAL II	Lic. Amahán Sánchez Álvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Ing. Marlon Antonio Pérez Turk
EXAMINADOR	Ing. Elizabeth Domínguez
EXAMINADOR	Ing. César Fernández
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

MODELO DE CAPACIDAD DE MADUREZ DEL *SOFTWARE* Y SU INFLUENCIA EN LAS MEJORAS DE CALIDAD DEL *SOFTWARE*

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha julio de 2002.

Whuendy Yasmina Chacón Tarot

Guatemala, 13 de febrero de 2004

Ing. Carlos Alfredo Azurdia Morales.
Coordinador Comisión de Trabajos de Graduación
Dirección de la Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad San Carlos de Guatemala

Ing. Azurdia:

Por medio de la presente hago de su conocimiento que he tenido a bien revisar el trabajo de graduación de Whuendy Yasmina Chacón Tarot, titulado “Modelo De Capacidad De Madurez Del *Software* Y Su Influencia En Las Mejoras De Calidad Del *Software*”, por lo cual me permito recomendar dicho trabajo para la respectiva revisión por parte de la comisión de trabajos de graduación de la escuela de Ciencias y Sistemas.

Sin otro particular, me suscribo atentamente,

Ing. José Ricardo Morales Prado

Guatemala, 17 de marzo de 2004

Ingeniero
Sydney Alexander Samuels Milson
Decano
Facultad de Ingeniería

Estimado Sr. Decano:

Atentamente, me dirijo a usted para informarle que después de conocer el dictamen del Asesor del trabajo de graduación de la estudiante Whuendy Yasmina Chacón Tarot, titulado "Modelo De Capacidad De Madurez Del *Software* Y Su Influencia En Las Mejoras De Calidad Del *Software*", procedo a la autorización del mismo.

Sin otro particular me suscribo con las muestras de mi consideración y estima.

"ID Y ENSEÑAD A TODOS"

Ing. Luis Alberto Vettorazzi España
COORDINADOR
CARRERA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

AGRADECIMIENTOS

A Dios	Fuente de sabiduría, amor y misericordia que me ha permitido alcanzar esta meta.
A mis padres, Mario† y Rosita	Por su amor, apoyo y darme la oportunidad de realizarme como profesional.
A mis hermanos	Sergio y Ericka, Ivonne y Estuardo, y Cintia; porque sus palabras me alentaron a seguir adelante, su apoyo y oraciones nunca faltaron, gracias.
Mis sobrinos	Heike, Brandon, Kristaly, Alexia, Mario y Andrea, como ejemplo de que las metas trazadas sí pueden cumplirse.
Mi novio, Mario Rodas	Por tu amor incondicional e incentivarme a seguir adelante.
Mi demás familia	Por sus oraciones, consejos y buenos deseos en el logro de mis metas, gracias por su apoyo.
Mis compañeros y amigos	Javier Ralda y Carolina de Ralda, Álvaro Díaz y Claudia de Díaz, Sergio Rodas, Néstor Ordóñez, Arturo Vásquez, Yohana Ponce e Ingrid García; porque su ayuda y amistad no faltaron en ningún momento.
A la familia Rodas	Por contar con ustedes en momentos difíciles, por el apoyo y cariño que me han brindado.
Al Ing. Ricardo Morales	Por su ayuda en la revisión del presente trabajo. Gracias.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Actividad Cualquier paso o función que se realiza (mental o físicamente) para alcanzar algún objetivo. Incluyendo todo el trabajo realizado para realizar las tareas del proyecto y la organización.

Áreas Clave del Proceso (ACP) Grupo de actividades relacionadas que cuando se llevan a cabo en conjunto alcanzan un conjunto de metas (consideradas importantes para aumentar la capacidad del proceso). Las ACP son al proceso de desarrollo de *software* lo que los cimientos a una casa. Cada una de las dieciocho ACP pertenece a uno y solo uno de los cinco niveles de madurez. Como resultado se obtiene un despliegue de los procesos de *software* que son efectivos, usables y aplicados constantemente a lo largo de la organización.

Capacidad Habilidad que tiene algo o alguien para cumplir con las exigencias, objetivos o metas.

Capacidad de un proceso El rango de resultados que se pueden obtener tras seguir un proceso.

CMMI	Siglas del modelo de capacidad de madurez integrado, no lleva las siglas SW (que representa la palabra *software*) por ser un modelo que abarca a toda la organización.
DRA	Siglas de desarrollo rápido de aplicaciones; es una metodología de desarrollo de *software*.
ECP	Siglas de la estructura común del proceso, se aplica al *software*; indica cuales son las características a tomar en cuenta en un proceso.
IEEE	Organización de estándares aplicados a normas, políticas, terminología, planes, herramientas, documentos y medición de una entidad.
Intranet	Red de computadores conectados entre sí, que se ubican dentro de una organización.
IPD-CMM	Siglas del modelo de capacidad de madurez de la integración de productos desarrollados.
KPA	Siglas de áreas clave del proceso en idioma inglés. Se tratan en el presente trabajo como ACP.
Madurez de un proceso de desarrollo de *software*	Es el punto hasta el cual un determinado proceso es explícitamente definido, administrado, medido, controlado y efectivo. La madurez es el potencial de crecimiento de la capacidad del proceso, también nos indica tanto

la riqueza del proceso de desarrollo de *software* de una organización, como la consistencia con la cual es aplicado en los proyectos a lo largo de la misma. La madurez de un proceso implica que la capacidad del proceso de desarrollo de *software* ha crecido. Específicamente debe ser: definido, documentado, entrenado, practicado, soportado, mantenido, controlado, verificado, validado, medido, y debe ser capaz de mejorar.

Nivel de madurez

Plataforma bien definida desde la cual se puede obtener un proceso maduro de *software*. A medida que una organización de *software* adquiere madurez en su proceso de desarrollo de *software*, ésta lo institucionaliza a través de políticas, estándares y estructuras organizacionales. La institucionalización conlleva a la construcción de una infraestructura y a una cultura corporativa que soporte los métodos, prácticas y procedimientos. Estos niveles definen una escala para medir la madurez y evaluar la capacidad de los procesos de *software*. Los niveles ayudan a la empresa a dar prioridades en el esfuerzo de mejora. En CMM se tienen cinco niveles de madurez para medir los procesos de *software*.

Proceso de desarrollo de *software*	Conjunto de actividades, métodos, prácticas y transformaciones para desarrollar y mantener *software* y productos asociados.
SA-CMM	Siglas del modelo de capacidad de madurez de la adquisición de *software*.
SE-CMM	Siglas del modelo de capacidad de madurez de la ingeniería de sistemas, siglas en inglés.
SEI	Siglas del instituto de ingeniería de *software*. Desarrollaron el CMM.
SQA	Siglas de aseguramiento de calidad del *software*, por presentarse en inglés; promueve una cultura de calidad del software para que cumpla con los requerimientos establecidos en una organización.
SW-CMM	Siglas del modelo de capacidad de madurez del *software*, siglas en inglés.
TQM	Siglas de administración de calidad total, cuenta con varias etapas en las que se implementan prácticas de calidad.

RESUMEN

El modelo de capacidad de madurez del software SW-CMM o CMM se basa en la mejora continua de procesos de desarrollo de software de una forma ordenada, consciente, estructural, sistemática, consistente; fue elaborado por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon.

En este trabajo se describen los distintos modelos y estándares cuyas características comunes se centran en la calidad de los procesos de desarrollo de *software*. Además, se explican los factores internos y externos de una organización que contribuyen e inciden en la calidad de los procesos de desarrollo de *software* y del producto terminado. También se aborda el proceso de desarrollo de *software* en las diferentes etapas que sugieren algunas metodologías de desarrollo de *software*.

El trabajo contiene la estructura del SW-CMM, que cuenta con niveles de madurez y un conjunto de áreas clave del proceso por cada nivel. Cada área clave involucra prácticas comunes que se implementan de una forma desordenada sin la aplicación del modelo. Este mismo capítulo trata las versiones que ha sufrido el SW-CMM, así como las diferencias.

Se muestran las diferencias entre ISO 9000 y SW-CMM y la transición entre ellos. Trata del modelo de madurez de capacidad del *software* integrado y sus diferencias con SW-CMM; al final, se presenta un caso de estudio a nivel guatemalteco de la aplicación del modelo.

OBJETIVOS

General

Conocer el modelo de madurez de la capacidad del *software*, su estructura, su influencia en los procesos de calidad de desarrollo de *software*, las diferencias entre SW-CMM y otros modelos y estándares para *software*, y la manera en que una organización adopta el modelo.

Específicos

1. Conocer estándares y modelos de *software* que adoptan la calidad de los procesos y productos.

2. Describir los factores que inciden de manera directa e indirecta en el proceso de desarrollo de *software*.

3. Desarrollar los aspectos que involucra el proceso de desarrollo, los puntos de vista del mismo en cuanto a SW-CMM y las metodologías de desarrollo de *software*.

4. Analizar la estructura del modelo de madurez de la capacidad del *software* y las últimas versiones de SW-CMM.

5. Especificar los aspectos sobresalientes del CMMI, de las series ISO 9000, las diferencias entre la últimas versiones de SW-CMM y el caso de uso del modelo en una organización en particular.

INTRODUCCIÓN

El presente trabajo fue elaborado sobre el modelo de madurez de la capacidad del *software* SW-CMM o CMM que se basa en la mejora continua de procesos de desarrollo de *software* de una forma ordenada, consciente, estructural, sistemática, consistente. El modelo SW-CMM fue elaborado por el Instituto de Ingeniería de *Software* (SEI) de la Universidad Carnegie Mellon. Se obtuvo información del modelo de fuentes bibliográficas, del sitio de Internet del SEI y de casos prácticos.

El tema se desarrolla en cinco capítulos, el primer capítulo incluye modelos y estándares cuyas características comunes se centran en la calidad de los procesos de desarrollo de *software* y que se encuentran en constante competencia con SW-CMM. Entre los modelos está el estándar ISO que es muy conocido e implementado en Guatemala por industrias de renombre nacional e internacional.

El segundo capítulo trata de los factores internos y externos que contribuyen e inciden en la calidad de los procesos de desarrollo de *software* ya sea de manera positiva o negativa o que de alguna u otra manera son considerados propios de la calidad. Se mencionan factores como: tecnología, educación, recurso humano, costos, riesgos, etc. Los factores, unos más que otros, son importantes en la mejora continua de procesos que promueve SW-CMM.

El capítulo tres aborda el proceso de desarrollo de *software*, las etapas que involucra desde que inicia hasta que termina, implicando también otras

características básicas; el punto de vista del proceso de desarrollo de *software* para SW-CMM y cómo involucra en su estructura la calidad a nivel de procesos de desarrollo de *software* y no del producto terminado. Se presentan las diversas metodologías de desarrollo de *software*, importantes éstas en la planificación de los proyectos de *software* y antes de iniciar con su desarrollo.

El cuarto capítulo involucra toda la estructura del SW-CMM, que cuenta con niveles de madurez del proceso de desarrollo de *software*, cada uno en forma escalonada. También la estructura nos muestra un conjunto de áreas clave del proceso (ACP) o KPA's (áreas clave del proceso por sus siglas en ingles) por cada nivel; hay que cumplir las áreas clave del proceso de un nivel para pasar al siguiente. Cada área clave involucra prácticas comunes para poder cumplir con las exigencias del modelo, estas prácticas comunes se implementan en empresas que no conocen el modelo o lo aplican, pero, de una forma desordenada. Como el modelo SW-CMM tiene varias versiones, estas se dan a conocer en este mismo capítulo y cuales son las diferencias entre ellas.

El capítulo cinco deja entrever las diferencias entre el estándar ISO 9000 y SW-CMM, así como de las ventajas y desventajas que presentan y cuál sería la transición del estándar ISO al modelo SW-CMM. Contiene las características más importantes de la norma ISO 9000. Además, hay un nuevo modelo para calificar la madurez del *software* y es CMMI (Modelo de Madurez de Capacidad del *Software* Integrado) este capítulo aborda los aspectos mas sobresalientes del modelo y los puntos de vista tanto positivos como negativos. En la parte final del capítulo se da a conocer un caso de estudio a nivel guatemalteco.

1. MODELOS Y ESTÁNDARES DE CALIDAD DE *SOFTWARE*

1.1 Ingeniería de *software*

Los modelos y estándares de calidad de *software* forman parte de la ingeniería de *software*; por eso comenzaremos con algunas definiciones de lo que es la Ingeniería de *software*:

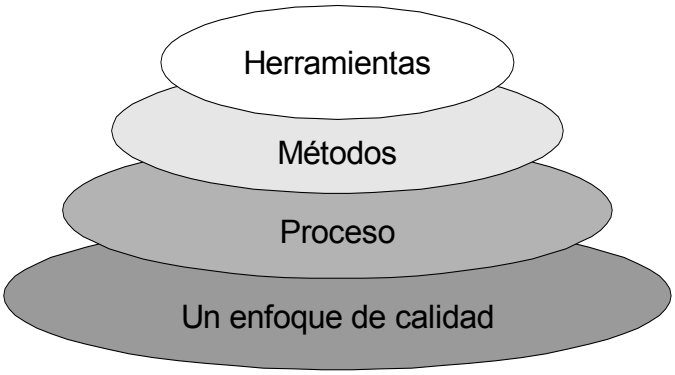
- Es la disciplina tecnológica y administrativa dedicada a la producción sistemática de productos de *software*, que son desarrollados y modificados a tiempo y dentro de un presupuesto definido con el objetivo de producir *software* de buena calidad de una manera sistemática y previsible (FARLEY, 1988).

- Es la disciplina cuyo fin es la producción de *software* libre de fallas, entregado a tiempo, dentro del presupuesto y que satisfaga las necesidades del cliente (SCHACH, 1998).

- La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del *software*, es decir la aplicación de ingeniería al *software*. Definición de la IEEE (Estándar IEEE, 610.12).

Si unificamos lo anterior concluimos que la ingeniería de *software* es una disciplina que integra proceso, métodos y herramientas para el desarrollo del *software* de computadora. La calidad es la base de todos ellos, como se puede observar en la figura 1.

Figura 1. Capas de ingeniería de *software*



Fuente: Roger Pressman, Capas de ingeniería de *software,* 20

El proceso es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la ingeniería del *software*. Los métodos indican cómo construir técnicamente el *software*. Las herramientas proporcionan un soporte automático o semi-automático para el proceso y para los métodos; y por supuesto la calidad que todo proceso debe llevar en conformidad con las herramientas y métodos que no inicia cuando todo está terminado sino antes de iniciar.

La ingeniería es el análisis, diseño, construcción, verificación y gestión de entidades técnicas. Para construir la ingeniería del *software* adecuadamente, se debe definir un proceso de desarrollo de *software*.

- La fase de definición se centra sobre el qué, es decir, durante la definición, el que desarrolla ha de identificar los requisitos clave del sistema y del *software*.

- La fase de definición de ingeniería del *software* de alguna manera tendrá lugar a tres tareas principales: a) ingeniería de sistemas o de informática, b) planificación del proyecto del *software* y c) análisis de los requisitos.

- La fase de desarrollo se centra en el cómo. Los métodos aplicados durante la fase de desarrollo varían, aunque las tres tareas específicas técnicas deberían ocurrir siempre: a) diseño del *software*, b) generación de código y c) prueba del *software*.

- La fase de mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del *software*, y a cambios debidos a las mejoras producidas por los pasos de las fases de definición y de desarrollo, pero en el contexto del *software* ya existente. Durante la fase de mantenimiento se encuentran cuatro tipos de cambios:

  1. Corrección
  2. Adaptación
  3. Mejora
  4. Prevención

Estas fases se complementan con un número de actividades protectoras, entre las cuales encontramos:

- Seguimiento y control del proyecto de *software*
- Revisiones técnicas formales
- Garantía de calidad del *software*
- Gestión de configuración del *software*
- Preparación y producción de documentos

- Gestión de reutilización
- Mediciones
- Gestión de riegos

1.2 Importancia de implementar modelos y estándares de calidad de *software*

Dado que la competencia cada día es más fuerte, es necesario que las empresas se preocupen en dar un mejor producto. Pero la calidad del producto no sólo se mide al terminarlo. La complejidad de los problemas que hoy en día buscan una solución en el *software* ha aumentado de manera considerable. Pero este crecimiento ha sobrepasado de sobremanera al aumento en la habilidad de desarrollar y mantener el *software* por parte de las organizaciones dedicadas a desarrollarlo o mantenerlo.

Enfrentamos una situación con dos caras. Por una parte las organizaciones quieren ser capaces de desarrollar y entregar *software* confiable, a tiempo y apegado al presupuesto acordado con el cliente. La segunda cara de la moneda nos muestra la perspectiva del cliente, el cuál quiere saber con certeza que todo lo anterior se cumplirá. Por esto las organizaciones deben buscar una norma, estándar o modelo que pueda ayudarlas a conseguir su meta de calidad (competitividad).

Sin embargo, la competitividad no es la única razón por la cuál se busque la calidad en el *software*. Debemos darle importancia a cada programa que se desarrolla. Debemos tomar conciencia y responsabilidad de las consecuencias que un defecto en nuestro producto podría ocasionar. Algunos defectos de *software* han ocasionado serios daños y hasta perjudicado

físicamente a personas. Gente ha muerto debido a *software* defectuoso (LEVESON, 1995).

El problema es que los sistemas cada vez son más rápidos, más complejos y automáticos. La posibilidad de una falla catastrófica aumenta a la par que el potencial del daño que podría ocasionar (PERROW, 1994) Así que debemos saber distinguir entre simple y fácil. Un error simple no necesariamente será fácil de encontrar, por tanto todos estamos involucrados en la calidad del producto, al ser responsables de la calidad de nuestro trabajo.

Otro aspecto negativo de los defectos es el económico. Cada defecto representa un costo adicional. Un error identificado en la misma fase donde se produjo es mucho más barato de resolver que el mismo defecto en una fase posterior, y aún más caro si éste sale a la luz después que el producto ya ha sido entregado.

Las siguientes son algunas razones importantes para implementar un sistema de calidad:

- Satisfacción del cliente
- Competencia
- Defectos

La buena implementación no sólo involucra el seguir los puntos o requerimientos que cada uno de los modelos o estándares señalan. El tener un proceso y prácticas documentadas de nada sirven si no se siguen. La norma por sí sola no dará un avance si no existe un compromiso por parte de la alta gerencia. O más aún, si las prácticas no se ejercen por cada uno de los integrantes de la organización.

La alta gerencia juega un papel muy importante dado que su visión del sistema de calidad es la que se manifiesta a todos los empleados. Si la gerencia observa a la norma como algo requerido por los clientes y no como algo beneficioso, lo mismo ocurrirá con el personal. La gerencia también es responsable de proporcionar los recursos necesarios para poder implementar el sistema de calidad. Debe existir un compromiso por parte de la gerencia en darle seguimiento y avance al sistema de calidad (MONTERO, 2000).

Para asegurar la buena implementación de cualquier norma o modelo se deben tomar en cuenta tres componentes:

- Las prácticas
- Las herramientas
- La gente

Las buenas prácticas deben institucionalizarse. La gente debe de ser capaz y responsable de seguir cada una de las prácticas que están definidas para toda la organización. Para poder ayudar a la gente a dar seguimiento a las prácticas correspondientes se puede hacer uso de herramientas especializadas. Las herramientas harán que las personas no vean al proceso como algo hostil y fastidioso. Es necesario definir que es lo que se va a hacer, por quien y cuando. Otro aspecto importante es el ciclo de vida de los procesos. El hecho de haber definido, documentado, medido e institucionalizado los procesos no significan que sean los mejores. Todo proceso está sujeto a cambios. Tener un mal proceso que no evoluciona representa más un obstáculo que una ayuda.

Un último punto sería el enfoque con que se ve el proceso. Los procesos deben ayudarnos a lograr un objetivo de la organización más no son ellos mismos el objetivo. La burocratización es el resultado de ver al proceso como objetivo (HUMPHREY, 2000).

1.3 Estándares más conocidos

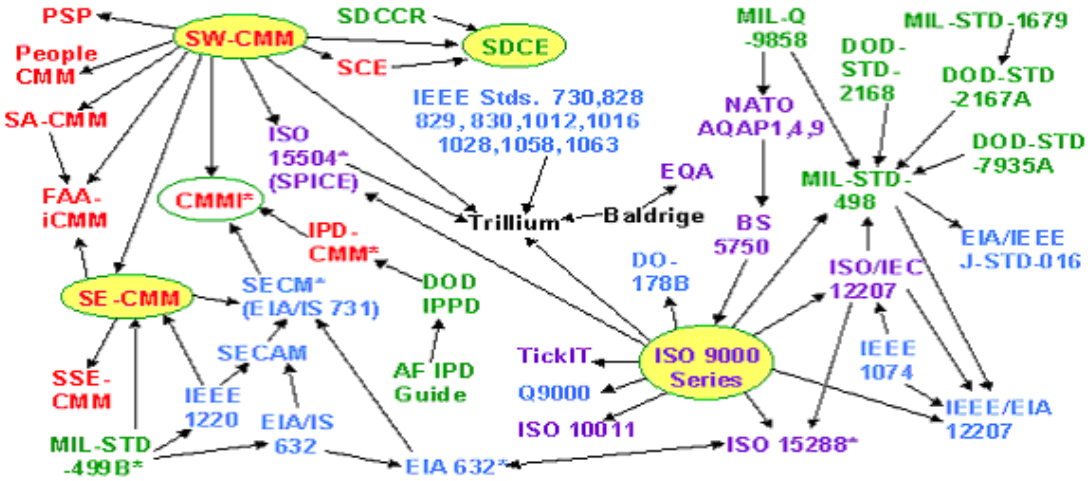
Hasta la fecha se han creado y aplicado muchos estándares y modelos de procesos y no se diga para la construcción y mantenimiento del *software*. Se han centrado en esto varias empresas interesadas en tener y mantener productos de alta calidad basados en estándares, pues esto proporciona no solo estabilidad sino comodidad en cuanto a la oferta y la demanda, puesto que un producto, si además de tener calidad esta respaldado por estándares conocidos, difícilmente pueda perder auge en el mercado de *software*.

Entre los estándares más conocidos podemos mencionar los siguientes:

1. Todas las series ISO 9000
2. SW-CMM
3. SE-CMM
4. IEEE *software*

Los modelos de trabajo se pueden clasificar como lo muestra la figura 2; mientras que la tabla 1 muestra una clasificación del modelo o estándar al que se aplican.

Figura 2. Marcos de trabajo



Fuente: SPC, Marcos de trabajo, 45

Tabla I. Área de aplicación de los estándares y modelos

Estándares o modelos que se aplican a un modelo de madurez.	SW-CMM, CMMI, SE-CMM, SSE-CMM, FAA-ICMM, PSP, People CMM, SA-CMM, SCE, IPD-CMM
Estándares o modelos que se aplican a un documento de una entidad.	SECM* (EIA/IS 731), SECAM, IEEE 1220, EIA/IS 632, EIA 632, Q9000, IEEE Stds. 730 828 829 830 1012 1016 1028 1058 1063, DO-178B, IEEE 1074, IEEE/EIA 12207, EIA/IEEE J-STD-016
Estándares o modelos que son estándares internacionales.	ISO 9000 Series, ISO 15504*(SPICE), NATO AQAP1 4 9, EQA, BS5750, ISO/IEC 12207, TickIT, ISO 10011, ISO 15288*
Estándares o modelos que se aplican a un documento del gobierno de EEUU.	SDCE, SDCCR, MIL-STD-499B*, DOD IPPD, AF IPD Guide, MIL-Q-9858, DOD-STD-2168, MIL-STD-1679, DOD-STD-2167A, DOD-STD-7935A, MIL-STD-498

Estándares o modelos que se aplican a otros diferentes a los mencionados (ejemplo: premios de calidad).	Trillium Baldridge

Debido a que los estándares están orientados a diferentes características se crearon categorías de marcos de trabajo, se catalogan tal como se observa en la siguiente lista:

1. Pautas y estándares
2. Modelos de mejoramiento de procesos y métodos de evaluación interna
3. Vehículos de selección de contratistas
4. Premios de calidad
5. Modelos de ciclo de vida de ingeniería de *software*

Entonces se clasificaron dentro de estas categorías todos aquellos marcos de trabajo de acuerdo a su alcance, propósito, enfoque y otros con la finalidad de poder orientar a los interesados en la creación de marcos de trabajo orientados a sus objetivos y metas; se mencionan los más importantes:

- El estándar ISO 9000 esta enfocado a empresas que fabrican productos centrándose en su calidad, su área de aplicación es a nivel comercial

- El modelo CMM para *software* se aplica a organizaciones de desarrollo de *software* y su propósito es la mejora de los procesos

- El modelo SE-CMM se aplica a organizaciones que desarrollan sistemas de información y también se orienta a la mejora de procesos

- El estándar IEEE *software* se aplica a todas las empresas que desarrollan, administran o comercian *software*, se orienta a las normas, políticas, terminología, herramientas, planes, documentos y medición aplicables al *software*.

1.4 Otros estándares

Así como fueron considerados los estándares principales que involucran desarrollo de *software*, también hay otros estándares que aunque no centrados en el desarrollo de *software* si pueden ser aplicados a ello; aunque son menos conocidos si han podido certificar a empresas de desarrollo de *software* a nivel de mejora de procesos y otros que implementan un conjunto de los estándares más conocidos.

El premio Baldridge se orienta a la calidad de los productos, asignando a cada proceso un porcentaje de calificación que al final determina si el producto lleva por lo menos un promedio óptimo de calidad.

El Trillium es un estándar aplicado a empresas en la mejora de procesos, basado en los mejores estándares y modelos más conocidos; como lo son ISO 9000, CMM y otros.

Los estándares de ciclos de vida del *software* que se orientan única y exclusivamente a empresas desarrolladoras de *software*, que si bien definen una estructurada metodología de desarrollo esta no es o no está por mucho tiempo debido a que la mayoría de las veces no se ajusta al proceso de desarrollo de la diversidad de organizaciones de desarrollo.

2. FACTORES QUE CONTRIBUYEN EN LA CALIDAD DE LOS PROCESOS

Antes de iniciar de lleno con los factores que afectan la calidad de los procesos cabe mencionar a que nos referimos con calidad y otros aspectos importantes relacionados con ello. Calidad es una característica o atributo de algo, como atributo de un artículo; la calidad se refiere a las características mesurables; cosas que se pueden comparar con estándares conocidos como longitud, color, propiedades eléctricas, maleabilidad.

La calidad de diseño se refiere a las características que especifican los ingenieros de *software* para un artículo y la calidad de concordancia es el grado de cumplimiento e las especificaciones de diseño durante su realización.

2.1 Control de calidad

El control de calidad es una serie de inspecciones, revisiones, y pruebas utilizados a lo largo del ciclo de desarrollo para asegurar que cada producto cumple con los requisitos que le han sido asignados. Un concepto clave del control de calidad es que se hayan definido todos los productos y las especificaciones mensurables en las que se puedan comparar los resultados de cada proceso. La garantía de calidad o aseguramiento de la calidad consiste en la auditoría y las funciones de información de la gestión.

2.2 Coste de calidad

El coste de calidad incluye todos los costes acarreados en la búsqueda de la calidad o en las actividades relacionadas en la obtención de la calidad. Los costes principales en el conseguimiento de la calidad son los siguientes:

Los costes de prevención:

- Planificación de la calidad
- Revisiones técnicas formales
- Equipo de pruebas
- Formación

Los costes de evaluación:

- Inspección en el proceso y entre procesos
- Calibrado y mantenimiento del equipo
- Pruebas

Los costes de corregir fallos:

- Revisión
- Reparación
- Análisis de las modalidades de fallos

Los costes de corregir fallos externos:

- Resolución de quejas
- Devolución y sustitución de productos
- Soporte de línea de ayuda
- Trabajo de garantía

2.3 Garantía de calidad del *software* (SQA)

La calidad de *software* se define como: concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo *software* desarrollado profesionalmente. La garantía de calidad del *software* es un diseño de acciones planificado y sistemático que se requiere para asegurar la calidad del *software*. La garantía de calidad del *software* comprende una gran variedad de tareas, asociadas con dos constitutivos diferentes, los ingenieros de *software* que realizan trabajo técnico y un grupo de SQA que tiene la responsabilidad de la planificación de garantía de calidad, supervisión, mantenimiento de registros, análisis e informes.

- Establecimiento de un plan de SQA para un proyecto. El plan identifica:
    - Evaluaciones a realizar
    - Auditorias y revisiones a realizar
    - Estándares que se pueden aplicar el proyecto
    - Procedimientos para información y seguimiento de errores
    - Documentos producidos por el grupo SQA
    - Realimentación de información proporcionada al equipo de proyecto del *software*
- Participación en el desarrollo de la descripción del proceso de *software* del proyecto
- Revisión de las actividades de ingeniería del *software* para verificar su ajuste al proceso de *software* definido

- Auditoría de los productos de *software* designados para verificar el ajuste con los definidos como parte del proceso de *software*

- Asegurar que las desviaciones del trabajo y los productos del *software* se documenten y se manejen de acuerdo con un procedimiento establecido

- Registrar lo que no se ajuste a los requisitos e informar a sus superiores

- Participar en el proceso de desarrollo

2.4 El tamaño sí importa

Actualmente, las organizaciones inmersas en el desarrollo y mantenimiento de *software* difieren en tanto y en casi todo, y si se habla de un tamaño, existe una enorme diferencia de tamaños organizacionales.

Específicamente el tamaño si importa, pero no es, precisamente, el factor más importante a nivel de calidad, pero en el modelo CMM el tamaño es mucho más importante de lo que se pensó puesto que fue creado para cubrir las necesidades de organizaciones con proyectos de gran tamaño, lo cual no significa que esta norma se aplicará al 100% en una organización grande. Tal vez para facilidad de conceptos sería bueno unificar los rangos de tamaño. Pero esto no sería totalmente justo, ya que al estandarizar esos rangos en el ámbito mundial las organizaciones pequeñas podrían correr el riesgo de pasar inadvertidas en el ámbito global. Entonces, se identifica que el tamaño responde a las propias necesidades de una determinada organización.

Ahora, el tamaño de una organización va de la mano con el número de personas que trabajan en ella. Si nuestra empresa es muy pequeña, lo más seguro es que se quiera ocupar al mayor número de personas para desarrollar *software*. Pero, si se cuenta con poco personal, probablemente la organización no va a utilizar a un grupo dedicado exclusivamente a mejorar o asegurar la calidad del proceso o productos de *software*, puesto que los empleados se mantienen ocupados en tareas mas especificas como el cumplir las metas en el tiempo previsto, sobre lo cual quizá pongan un poco de empeño para conseguir un trabajo de y con calidad, pero, esa no es la principal meta. La cadena de valor de la organización es distinta.

Pero el tamaño de las organizaciones no es lo único que varía, también se pueden descubrir otros tamaños importantes como:

- El tamaño del proyecto
- El tamaño del programa (en código)
- El tamaño del equipo de trabajo, y el ruido en la comunicación del mismo
- El tamaño de los errores inadvertidos
- El tamaño del intervalo de tiempo antes de ver los resultados de implementar un modelo de mejoramiento del proceso

El último punto antes citado podría pensarse sin importancia; sin embargo existen varias organizaciones que no están acostumbradas a esperar. Se quieren ver resultados rápidos, y si no hay resultados inmediatos la acción y la metodología es descartada. Por esto, el tamaño del intervalo de tiempo de respuesta si tiene peso.

2.5 Educación

Se habla de educación en un sentido figurado, porque el personal participante en una organización si está educado, es decir, está preparado para enfrentar los retos que su profesión requiere, que está calificado para poder realizar el trabajo asignado. Si queremos calidad debemos pensar en ingeniería y administración de *software*, pero, no se le esta dando el nivel de importancia adecuado a la enseñanza de estas dos áreas al personal participante en la organización. Hay que hacer una evaluación acerca de qué tan actualizada es la información en cuanto a técnicas, herramientas y modelos que se distribuye a los futuros profesionales y cuales por consiguiente son los medios para realizar su trabajo.

Según estudios realizados los problemas de calidad experimentados por la industria de *software* se deben a la ignorancia de técnicas, (estándares relativos) en otras industrias que, sin embargo, y pese a los constantes malos resultados continúan siendo ignoradas por los ingenieros de *software*.

La enseñanza de las técnicas, herramientas, programas se basa más en el producto que en el proceso que se debe llevar a cabo para llegar a ese producto. En otras palabras, se identifica la meta, más no las montañas ni abismos que hay entre ella y el personal; todos ellos acostumbrados por rutina, la mayoría de las veces, a realizar su trabajo aunque conozcan poco del resto de trabajo que sus compañeros realizan para llegar a esa meta, no se centran en la sinergia del sistema sino solo en lo que les toca realizar; si hay técnicas y herramientas por aplicar el empirismo las reemplaza.

Existe la mala costumbre de desarrollar sin planificar. Estamos seguros que la idea que llevamos en la cabeza es la correcta y nos saltamos fases de análisis y planeación. Creemos que todos los casos posibles ya los hemos identificado y omitimos la fase de diseño. Pensamos tener memoria eterna y omitimos la documentación explicativa. ¿Y si algo sale mal? ¡Para eso contamos con la improvisación! La documentación se ve más como un pesado costal de papas que como un mapa que nos ayudará a alcanzar la meta. Por lo tanto, ésta será llenada al finalizar el producto aunado con nuestra apatía y pensamiento de trabajo y tiempo perdido.

Pero, ¿cómo podremos mejorar si no podemos recordar? ¿Cómo podremos enseñar si no contamos con un método definido? Un mapa no sirve de nada si no se sabe dónde se está. Hay que olvidarse de las malas costumbres, la organización debe estar inmersa en una cultura de constante planeación, orden y seguimiento; aplicar reglas que lleven a los empleados a querer aprender y educarse por muy profesionales y capaces que sean; hay que ocuparse de la educación y promover una cultura de *software* con calidad.

2.6 Alta tecnología

Para cubrir las exigencias que una norma o modelo plantea, es necesario contar con determinadas herramientas, usualmente muy poderosas. El último muy puede interpretarse como dinero. Nos enfrentamos pues a una cruel realidad: la calidad cuesta. Es probable que la mayoría de las organizaciones desarrolladoras de *software* no cuenten con los recursos suficientes para adquirir este tipo de tecnología. Por cierto, al costo de la herramienta debemos sumar el costo que involucra entrenamiento, personal, tiempo invertido.

La mayoría de las organizaciones que desarrollan *software* no cuenta con la mejor tecnología y unas cuantas cuentan con tecnología obsoleta, esto se debe a factores económicos, como culturales y educativos; la organización necesita conocer y mantener una cultura tecnológica que se mantenga al día y actualizada, además de preparar al personal para su correcto uso y manejo.

Pero quizá este sea un punto a favor y no en contra ¿Por qué? Aquí cabe citar a Michael Hammer (el padre de la revolución de la re-ingeniería) quien nos dice que la automatización de malos procesos sólo agrava más la ineficiencia. Por lo tanto las empresas primero deberán enfocarse en solo identificar los procesos su inicio y su fin, ni siquiera querer mejorarlos, bastará con tan solo terminar los procesos. Ya después se podrá enfocar en mejorar uno o varios de esos procesos y por último automatizarlos.

Después de haber leído todo lo anterior podría pensarse que alcanzar una certificación o cumplir con ciertos estatutos de normas es una meta demasiado elevada para empresas u organizaciones pequeñas. Esto es falso, si se tiene la posibilidad de competir con empresas de economías de primer mundo. Se quiere mejorar, se busca contar con certificaciones de reconocimiento mundial. Se tiene la capacidad, solo basta comprometerse.

2.7 Otros

Hay organizaciones que toman en cuenta otros factores dentro del conseguimiento de la calidad para sus productos y sus procesos, que quizá, suenan aun más importantes estos que los anteriormente mencionados; pero esto depende de la organización, de su cultura, su educación, su área especifica, sus metas y propósitos, sus recursos, etc.

Podemos tomar en cuenta, como uno de los factores que no se puede obviar al considerar la calidad de los procesos y productos, al tiempo, este es importante puesto que si se planifica el tiempo para la elaboración de un producto y se sabe aprovechar, la organización va a tener más a favor que en contra. El tiempo es uno de los recursos que contribuyen con la calidad del proceso puesto que se debe dar cierta elasticidad al emplearlo de tal modo que demasiado tiempo o poco tiempo no dañen el conseguimiento de la calidad del producto final. Realizar las tareas y actividades que se han planificado en el tiempo que se ha planificado, si se puede optimizar el tiempo sin dañar la calidad y otros atributos del producto hay que hacerlo; pero, si el personal es esclavo del tiempo muchas veces la calidad se deteriora.

A medida que crece el tamaño del proyecto más personas se ven envueltas. Aunque la comunicación es absolutamente esencial para el éxito del desarrollo del *software*, cada nueva vía de comunicación requiere un esfuerzo adicional y, por tanto, un tiempo adicional. Se puede obtener beneficios usando menos gente durante un periodo de tiempo algo mayor para realizar el mismo trabajo.

También se debe considerar el esfuerzo, una distribución recomendada de esfuerzo en las fases de definición y desarrollo se la conoce normalmente con la regla 40-20-40. El cuarenta por ciento de todo el esfuerzo o más se asigna a las tareas de análisis y diseño. Un porcentaje similar se aplica a las pruebas, se puede deducir correctamente que no se insiste mucho en la creación de código.

El recurso económico es uno de los factores a considerar, hay empresas con economías que pueden sufragar los costos y gastos que conlleva la realización de todos sus procesos y todos aquellos detalles que incurren en

costos. Si la organización desea ingresar al mercado con un producto de calidad en grande debe invertir en grande, de lo contrario, puede conformarse con hacer uso óptimo de los recursos con los que cuenta aunque para ello deba haber más esfuerzo y tiempo.

Centrándose en el desarrollo de *software* específicamente, no solo en los procesos como se consideró antes, se pueden mencionar los siguientes factores que contribuyen a la calidad del *software*:

- Correcciones que se le puedan o deban realizar
- Fiabilidad
- Eficiencia
- Integridad
- Usabilidad
- Facilidad de mantenimiento
- Facilidad de prueba
- Reusabilidad
- Interoperatividad

Estos factores deben ser tomados en cuenta en la planificación y no cuando se está en la etapa de implementación, puesto que la calidad es considerada como un estado y los factores sus transiciones, en los cuales se puede llegar a la mejor calidad implementando cada factor; si se desarrolla tomando en cuenta estos factores además de los ya mencionados el producto al final puede ser bueno y la organización puede superarse a sí misma.

3. EL PROCESO DE DESARROLLO DE *SOFTWARE*

Los procesos son los medios por los cuales las personas, procedimientos, métodos, equipo y herramientas se integran para producir un resultado final deseado.

Se establece un marco común del proceso en el que se define un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos del *software*, con independencia de su tamaño o complejidad. Se definen un conjunto de tareas que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de *software* y a los requisitos del equipo del proyecto.

No todas las actividades forman parte del marco de trabajo de un proceso, un ejemplo: las actividades de protección, tales como garantía de calidad del *software*, gestión de configuración del *software* y medición; puesto que estas actividades son independientes de cualquier actividad del marco de trabajo.

3.1. Premisas de administración del proceso

- La calidad de un sistema de *software* es gobernado grandemente por la calidad de los procesos utilizados en su desarrollo y mantenimiento

- Tener un enfoque del proceso también como producto

- El valor de esta premisa es visible a nivel mundial en los movimientos de administración de calidad total en las industrias de manufacturación y servicio

3.2. Madurez del proceso

Se establecen cinco niveles de madurez del proceso, que se definen de la siguiente forma.

Nivel 1:	Inicial	El proceso del *software* se caracteriza según el caso, y ocasionalmente incluso de forma caótica.
Nivel 2:	Repetible	Se establecen los procesos de gestión de proyecto para hacer seguimiento del coste, de la planificación y de la funcionalidad.
Nivel 3:	Definición	El proceso del *software* de las actividades de gestión y de ingeniería se documentan, se estandarizan y se integran dentro de un proceso de *software* de toda una organización.
Nivel 4:	Gestionado	Se recopilan medidas detalladas del proceso del *software* y de la calidad del producto.
Nivel 5:	Optimizado	Mediante un resultado cuantitativo del proceso y de las ideas y tecnología innovadoras se posibilita una mejora del proceso.

3.3. Áreas clave del proceso

Las áreas clave del proceso ACP (o KPA por sus siglas en inglés) se refieren al conjunto de requerimientos necesarios que se deben cumplir en cada proceso, estas involucran:

- Objetivos propuestos que se deben alcanzar
- Compromisos que se deben cumplir para lograr los objetivos propuestos
- Capacidades que permiten que la organización cumpla los objetivos
- Actividades requeridas para lograr la función de ACP (o áreas clave del proceso)
- Métodos utilizados para supervisar la implementación
- Métodos utilizados para verificar la implementación

El gestor del proyecto debe decidir que modelo de proceso es el más adecuado para el proyecto, después debe definir un plan preliminar basado en un conjunto de actividades estructurales válidas para cualquier proyecto. Una vez establecido el plan preliminar, empieza la descomposición del proceso.

La planificación de un proyecto empieza con la maduración del problema y del proceso, entendiendo y documentando los requerimientos mínimos que cada proceso debe comprender. La estructura común de proceso (ECP) se refiere a las consideraciones que deben de llevar antes, durante y después de cada proceso. Se debe considerar en la estructura común de un proceso, independientemente de su fin, lo siguiente:

- Comunicación con el cliente
- Planificación

- Análisis de riesgo
- Ingeniería
- Construcción
- Entrega y evaluación del cliente.

La descomposición del proceso comienza cuando el gestor del proyecto pregunta: ¿Cómo vamos a realizar esta actividad de ECP?

3.4. Planes y procedimientos

La complejidad de los trabajos de *software* al igual que su tamaño cada día van en aumento. Es por esto que es importante planear (diseñar). El planear nos permite ver al proyecto desde una perspectiva global. Podemos identificar el conjunto de tareas que deberemos llevar a cabo, de una manera mucho más eficiente. Es importante que durante la planeación se incluyan actividades de medición. Si podemos medir nuestro proyecto podremos identificar malas o buenas prácticas, y darle un seguimiento mucho más exacto.

Un plan es flexible, se adecua al proyecto. Un procedimiento, al contrario de un plan, es estricto. El procedimiento se debe seguir completamente sin salto de pasos. Si el proyecto no se adecua al procedimiento, el proyecto es el que debe cambiarse. En ocasiones, seguir un procedimiento es necesario, y más cuando el no seguirlo al pie de la letra pueda contraer riesgos peligrosos o muy costosos.

Ya que hablamos de planes y procedimientos, si vamos desde lo más flexible hasta lo más estricto e incambiable, es que podemos ubicar a los procesos. Un proceso aplica para un conjunto de trabajos, a diferencia del plan

que sólo aplica para uno. Si se define un plan el cual es usado o será usado en múltiples ocasiones, será más conveniente definir un proceso. Ahora bien, no importa que se cuente con un proceso; cada proyecto deberá definir su propio plan. Este plan será más sencillo de crear ya que contamos con un proceso que nos indica aquellos aspectos importantes y necesarios que se deben tomar en cuenta en el proyecto.

Un proceso es una guía a seguir, más no un procedimiento inflexible. Algunos de los elementos que conforman al proceso deberán permanecer sin cambio alguno, mientras que algunos otros podrán variar en su contenido. Ya que un proceso puede ajustarse a un proyecto hasta cierta medida, es necesario que el ingeniero aplique sus conocimientos y buen juicio para definir el proceso a seguir.

Se debe tener cuidado de no convertir a un proceso en un procedimiento burocrático. Un proceso debe estar abierto a un mejoramiento continuo, sobretodo si se adquiere una disciplina de calidad a todo nivel y en todo momento del producto final. Si rechazamos la idea de que el proceso puede ser erróneo nos mantendríamos en el error, y peor aún ese error se diseminaría a lo largo de toda la organización causando grandes pérdidas.

Si el mapa y el terreno no son iguales, será mejor apegarse al terreno. Haciendo una analogía, el plan es nuestro mapa y el terreno el trabajo real. Hay que permanecer abiertos a posibles cambios.

3.5. Metodologías de desarrollo y su relación con los procesos y procedimientos

Cuando se habla del proceso de desarrollo de *software* no podemos obviar la metodología a aplicar, pues ésta es vital en el desarrollo del proceso; puesto que si se evalúa y si se implementa tal cual o cual metodología puede ser que al final del camino la que se haya seleccionado sea la que menos se ajuste, la que provoque incertidumbre, la que no sea implementada y en el peor de los casos la que colaboró con el fracaso total del proyecto.

Hay variadas metodologías y éstas son ajustables a cualquier tipo de proyecto, no se pude inferir en apoyar e implementar una u otra, porque cada una posee las características necesarias que deben ser tomadas y evaluadas en el análisis.

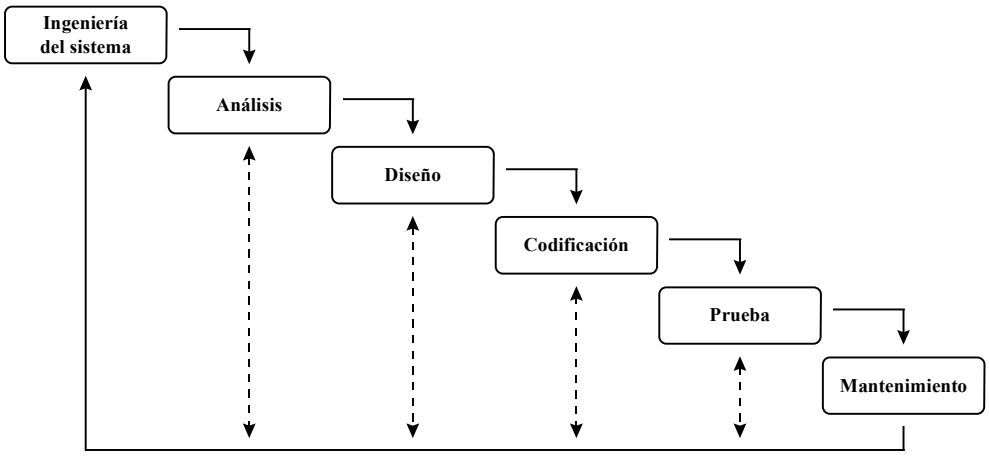
Se selecciona una metodología de desarrollo para la ingeniería del *software* según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizar y los controles y entregas que se requieren, etc. Todo el desarrollo del *software* se puede caracterizar como un bucle de resolución de problemas en el que se encuentran cuatro etapas de solución:

- Estado actual (representa el estado actual del problema)
- Definición de problema
- Desarrollo técnico
- Integración de solución

Ahora se definirán algunas metodologías de desarrollo que tienen un papel principal cuando se planifica, de ellas depende en gran parte la falta o no del éxito. Se hace referencia a las metodologías unas ya bien conocidas, otras que fueron implementadas en el pasado, etc.

3.5.1. El modelo lineal secuencial

Figura 3. Modelo lineal secuencial



Fuente: Kenneth Kendall, Modelo lineal secuencial, 70

El modelo lineal secuencial es llamado algunas veces “ciclo de vida básico” o “Modelo en cascada”, se muestra en la figura 3, sugiere un enfoque sistemático-secuencial del desarrollo del *software* que comienza en un nivel de sistemas y progresa con las etapas de análisis, diseño, codificación, pruebas y mantenimiento. El modelo lineal secuencial acompaña a las actividades siguientes:

- Ingeniería y modelado de sistemas/información: esta visión del sistema es esencial cuando el *software* se debe interconectar con otros elementos como hardware, personas y bases de datos
- Análisis de los requisitos del *software*
- Diseño
- Generación de código
- Pruebas
- Mantenimiento

El modelo lineal secuencial es el paradigma más antiguo y más extensamente utilizado en la ingeniería del *software*. Entre los problemas que se encuentran algunas veces en él se incluyen:

- Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo

- A menudo es difícil que el cliente exponga explícitamente todos los requerimientos

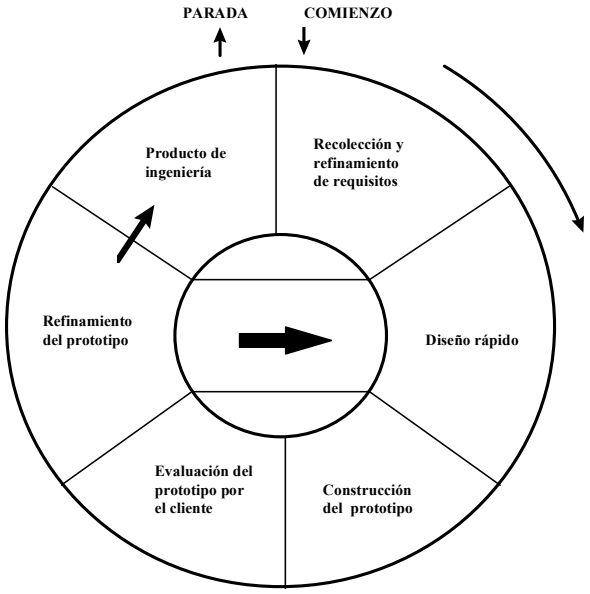
- Hay ocasiones en las que el cliente debe tener paciencia en su espera por el producto terminado

- Los responsables del desarrollo del *software* la mayoría de las veces se retrasan innecesariamente

3.5.2. El modelo de construcción de prototipos

El paradigma de construcción de prototipos es representado en la figura 4, el cual comienza con la recolección de requerimientos. El desarrollador y el cliente encuentran y definen los objetivos globales para el producto de *software*, identificando los requisitos conocidos, y las áreas del esquema en donde es obligatoria más definición.

Figura 4. Modelo de prototipos



Fuente: Kenneth Kendall, Prototipos, 72

El prototipo lo evalúan cliente/usuario y lo utilizan para refinar los requisitos del *software* a desarrollar. La interacción ocurre cuando el prototipo satisface las necesidades del cliente, a la vez que permite que el desarrollador comprenda mejor lo que se necesita hacer.

Sucede a menudo que se desarrollan varios prototipos hasta llegar al que realmente si cumple un porcentaje elevado de aceptación por sobre los demás, pero en los prototipos que no han sido aprobados se han perdido recursos muy valiosos.

La construcción de prototipos puede ser problemático por las razones siguientes:

1. El cliente ve lo que parece ser una versión de trabajo del *software*, sin saber que con la prisa de hacer que funcione no se ha tenido en cuenta la calidad del *software* global o la facilidad de mantenimiento largo.

2. El desarrollador a menudo hace compromisos de implementación para hacer que el prototipo funcione rápidamente.

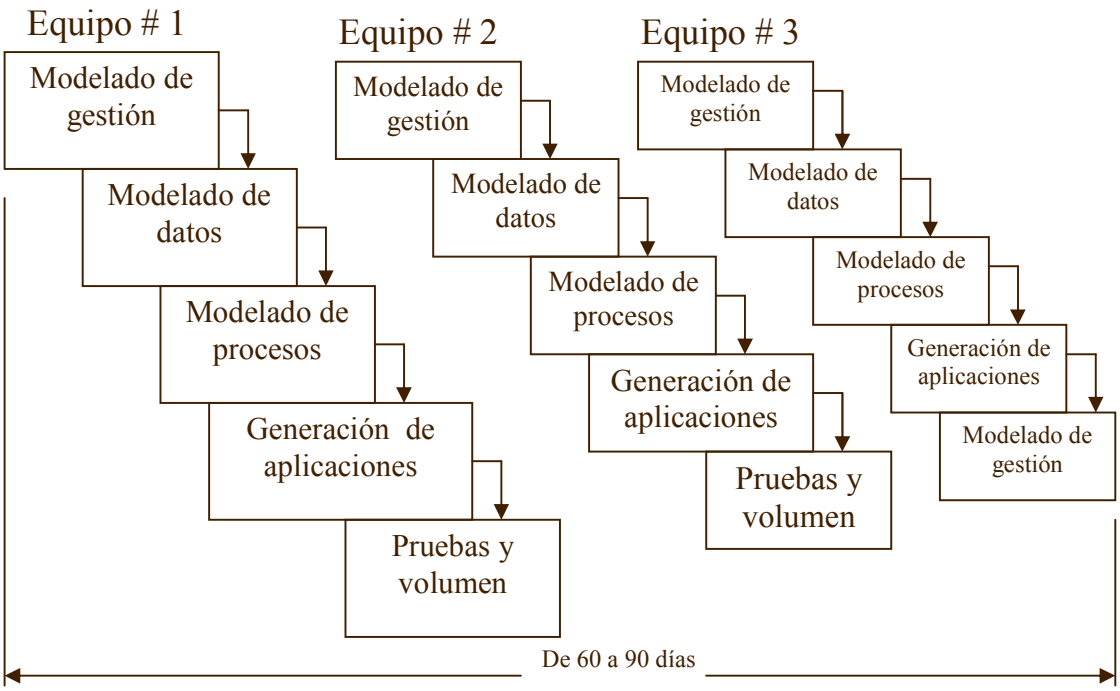
Aunque pueden surgir problemas, la construcción de prototipos puede ser un paradigma efectivo para la ingeniería del *software*. La clave es definir las reglas del juego al comienzo, es decir, el cliente y el desarrollador se deben poner de acuerdo en que el prototipo se construya para servir como un mecanismo de definición de requisitos.

3.5.3. El modelo DRA

El Desarrollo Rápido de Aplicaciones (DRA) (*Rapid Application Development*, RAD) se muestra en la figura 5. El modelo DRA es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Es una combinación de diversas técnicas estructurales con

técnicas de prototipos y de desarrollo conjunto de aplicaciones cuyo fin es acelerar el desarrollo de sistemas.

Figura 5. Modelo DRA



Fuente: Kenneth Kendall, Modelo DRA, 75

El enfoque de desarrollo rápido de aplicaciones comprende las siguientes fases:

- Modelado de gestión
- Modelado de datos
- Modelado de proceso
- Generación de aplicaciones
- Prueba y entrega

Al igual que todos los modelos, el enfoque DRA tiene inconvenientes:

- Para proyectos grandes aunque por escala, requiere recursos humanos suficientes como para crear el número correcto de equipos DRA

- Requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado

- No todos los tipos de aplicaciones son apropiados para DRA, no es apropiado cuando riesgos técnicos son altos, enfatiza el desarrollo de componentes de programas reutilizables.

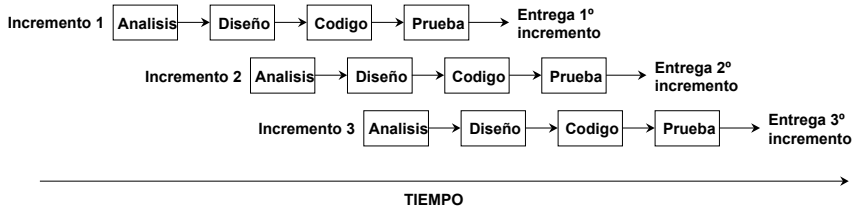
3.5.4. Modelos de procesos evolutivos de *software*

Se reconoce que el *software* al igual que todos los sistemas complejos evolucionan con el tiempo, los requisitos de gestión y de producto a menudo cambian conforme a que el desarrollo procede haciendo que el camino que lleva al producto final no sea real. Los modelos evolutivos son iterativos. Se caracterizan por la forma en que permiten a los ingenieros en *software* desarrollar versiones cada vez más completas del *software*. Se presentan a continuación algunos de los modelos que se clasifican en esta categoría.

3.5.4.1. El modelo incremental

Combina elementos del modelo lineal secuencial con la filosofía iterativa de construcción de prototipos. Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial, es decir, se afrontan requisitos básicos pero muchas funciones suplementarias sin extraer, esto puede darse en la siguiente iteración. La figura 6 muestra la estructura propuesta por el modelo incremental:

Figura 6. Modelo incremental



Fuente: Kenneth Kendall, Modelo incremental, 78

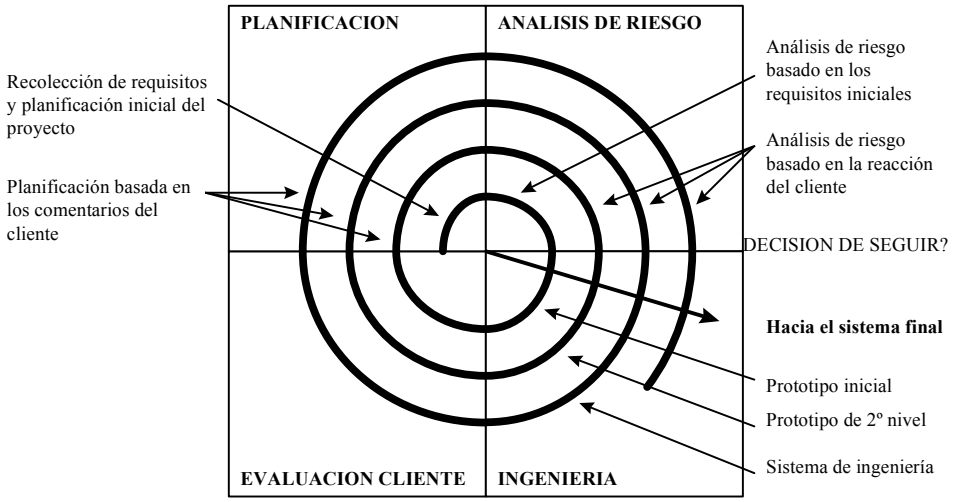
En este modelo el cliente utiliza el producto central. Como un resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento siguiente. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabora el producto completo, el modelo incremental se centra en la entrega de un producto operacional con cada incremento.

El desarrollo incremental es particularmente la dotación de personal que está disponible para una implementación completa en cuanto a la fecha límite de gestión que se haya establecido para el proyecto.

3.5.4.2. El modelo en espiral

Es un modelo que acompaña la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo inicial secuencial. En este, el *software* se desarrolla en una serie de versiones increméntales. Primeras iteraciones: un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez mas completas de ingeniería de sistemas. La figura 7 contempla el modelo espiral.

Figura 7. Modelo espiral



Fuente: Kenneth Kendall, Modelo Espiral, 80

El modelo en espiral se divide en un número de actividades estructurales, también llamadas regiones de tareas, un modelo en espiral puede contener las seis regiones de tareas:

- Comunicación con el cliente
- Planificación
- Análisis de riesgo

- Ingeniería
- Construcción y adaptación
- Evaluación del cliente

El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del *software* de computadora. Este modelo es un enfoque realista del desarrollo de sistemas y de *software* a gran escala. El modelo en espiral utiliza la construcción de prototipos como mecanismos de reducción de riesgo, pero lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto.

Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico. Puede resultar difícil convencer a grandes clientes de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la evolución del riesgo, el modelo en sí mismo es relativamente nuevo y no se ha utilizado tanto como los paradigmas lineales secuenciales o de construcción de prototipos.

3.5.4.3. El modelo de ensamblaje de componentes

Incorpora muchas de las características del modelo en espiral, sin embargo, el modelo ensamblador de componentes configura aplicaciones desde componentes preparados de *software*. La actividad de la ingeniería comienza con la identificación de clases candidatas.

Las clases creadas en los proyectos de ingeniería del *software* anteriores se almacenan en una biblioteca de clases o depósito. Una vez identificadas las clases candidatas, la biblioteca de clases se examina para determinar si estas

clases ya existen. Si una clase candidata no reside en la biblioteca, se aplican los métodos orientados a objetos o a construirse.

El flujo de proceso vuelve a la espiral y volverá introducir por ultimo la iteración ensambladora de componentes a través de la actividad de ingeniería. Estudios de reutilización informan que el ensamblaje de componentes lleva a una reducción de 70% de tiempo de ciclo de desarrollo, un 84% del coste del proyecto y un índice de productividad del 26.2% comparado con la norma de industria de 16.9%.

3.5.4.4. El modelo de desarrollo concurrente

Llamado algunas veces Ingeniería Concurrente, el modelo de proceso concurrente se puede representar en forma de esquema como una serie de actividades técnicas importantes, tareas y estados asociados a ellas. Todas las actividades existen concurrentemente, pero residen en estados diferentes.

El modelo de proceso concurrente define una serie de acontecimientos que dispararán transiciones de estado a estado para cada una de las actividades de la ingeniería de *software*. Este método se utiliza a menudo como el paradigma de desarrollo cliente/servidor. Cuando se aplica cliente/servidor, el modelo de proceso concurrente define actividades en dos dimensiones: de sistemas y de componentes. Los aspectos del nivel de sistemas se afrontan mediante tres actividades: diseño, ensamblaje y uso. La dimensión de componentes se afronta con las actividades de diseño y realización.

En realidad, el modelo de proceso concurrente es aplicable a todo tipo de desarrollo de *software* y proporciona una imagen exacta del estado actual de un proyecto.

3.5.4.5. El modelo de métodos formales

Los métodos formales permiten que un ingeniero del *software* especifique, desarrolle y verifique un sistema basado en computadora aplicando una notación rigurosa y matemática. Cuando se utiliza este método durante el desarrollo, proporcionan un mecanismo para eliminar muchos de los problemas que son difíciles de superar con paradigmas de la ingeniería de *software*. La ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen más fácilmente, no mediante una revisión a propósito para el caso, sino mediante la aplicación del análisis matemático.

Preocupaciones sobre su aplicabilidad:

- El desarrollo de modelos formales es bastante caro y lleva mucho tiempo

- Se requiere de un estudio caro porque los responsables del desarrollo de *software* tienen los antecedentes necesarios para aplicar métodos formales

- Es difícil utilizar los modelos como un mecanismo de comunicación con clientes que no tienen muchos conocimientos técnicos

No obstante, métodos formales tiene más partidarios entre los desarrolladores del *software* que deben construir *software* de mucha seguridad y entre los desarrolladores que pasan por grandes dificultades económicas al aparecer errores de *software*.

3.5.4.6. El modelo orientado a objetos

Hablar del modelo orientado a objetos lleva consigo la necesidad de hablar de objetos, de componentes, de patrones, de lenguajes, de arquitecturas, de persistencia, etc. Por otra parte, la tecnología de objetos hoy pasa por la integración de cada una de las técnicas y herramientas de desarrollo, tanto de análisis y diseño, como de programación, persistencia o distribución.

La programación orientada a objetos (llamada comúnmente POO) involucra clases, atributos y asociaciones entre clases, no contrario al paradigma relacional su utilización posee grandes ventajas, entre ellas:

- Reutilización de código desarrollado
- Aplicación de herencia
- Aplicación de polimorfismo

Por lo general, se puede decir que una metodología de desarrollo de *software* orientado a objetos consta de los siguientes elementos:

- Conceptos y diagramas
- Etapas y definición de entregas en cada una de ellas
- Actividades y recomendaciones

Etapas y actividades en un desarrollo orientado a objetos:

- Análisis de requerimientos
  - Identificar casos de uso del sistema
  - Dar detalle a los casos de uso descritos
  - Definir una interfaz inicial del sistema (si es aplicable)
  - Desarrollar el modelo
  - Validar el modelo

- Diseño del sistema
  - Identificar la arquitectura del sistema

- Diseño detallado
  - Agregar detalles de implementación al modelo
  - Desarrollar el modelo de interfaz
  - Desarrollar los modelos de control, persistencia y comunicaciones

- Implementación y pruebas
  - Definir estándares de programación
  - Codificación y pruebas unitarias
  - Pruebas de módulos y de sistema

4. MODELO DE CAPACIDAD DE MADUREZ DEL *SOFTWARE*

Las organizaciones que desarrollan y administran *software* presentan problemas de inmadurez debido a su falta de habilidad para administrar sus procesos de desarrollo de *software*, de hecho este es el problema fundamental. El CMM para *software* (CMM-SW) se convierte en una guía que nos ayudará a obtener el control sobre estos procesos y así desarrollar y mantener un mejor *software*. La meta a alcanzar será la evolución hacia una cultura de excelencia tanto en la Ingeniería de *software* como en la Administración de *software*.

El CMM incluye prácticas de planeación, ingeniería y administración de desarrollo y mantenimiento de *software*. Si se siguen estas prácticas aumentará la habilidad con la cual una organización podrá alcanzar metas como costo, programas basados en tiempos, funcionalidad y un producto de alta calidad.

El propósito de CMM es el guiar a las organizaciones en la selección de estrategias de mejora determinando la madurez del proceso actual e identificando los puntos importantes que se deben atacar para así mejorar tanto el proceso como la calidad del *software*.

Ahora, ¿por qué confiar en CMM? El modelo de capacidad de madurez está basado en prácticas reales, refleja las mejores prácticas en el área, también refleja la necesidad de los individuos de llevar a cabo una mejora en el proceso de desarrollo de *software*, al igual que la valoración del proceso de desarrollo de *software* el CMM está documentado y es público.

4.1. Historia

Para entender el origen del CMM es necesario regresar a 1986 cuando el *Software Engineering Institute* (SEI) junto con MITRE *Corporation* buscaron mejorar el proceso de desarrollo de *software* y comenzaron a desarrollar un marco de trabajo que llamaron Proceso de Madurez. Éste esta basado en el concepto de la Administración de la Calidad Total (TQM), el cual cuenta con cinco etapas evolutivas hacia una implementación de prácticas de calidad. CMM es una aplicación de TQM para *software*.

En 1989, W. Humphrey, el entonces Director del SEI, publica el libro Administrando el Proceso de *Software* (*Managing the Software Process*). En esta obra ya encontramos un marco de trabajo definido por cinco niveles de madurez. Este marco presentaba dos métodos (valoración del proceso de *software* y la evaluación de la capacidad del *software*) y un cuestionario de madurez para valuar la madurez del proceso de desarrollo de *software*.

Se involucran diferentes empresas y académicos relacionados al área de ingeniería de *software* y en 1991 ese marco de trabajo evoluciona para convertirse en el Modelo de Capacidad de Madurez (CMM). Este modelo presenta un conjunto de prácticas divididas en dieciocho Áreas Clave de Proceso que han demostrado acrecentar la capacidad de los procesos de *software*. El CMM siguiendo su propia filosofía continúa y continuará evolucionando siempre hacia un mejoramiento continuo.

De acuerdo con este modelo, el control de procesos de desarrollo de *software* y la excelencia en ingeniería y la administración del *software* podrán ser alcanzados, si se puede mejorar un conjunto finito de actividades que conformen la base central de un área clave. Al enfocarnos en este pequeño

conjunto de actividades y trabajando agresivamente para cubrirlas en su totalidad, la organización podrá mejorar de una manera estable y gradual en lo que respecta a sus procesos de *software*.

Es necesario entonces poder determinar el grado de madurez de los procesos de *software* actuales e identificar los puntos clave en donde se deberá enfocar la atención de mejora para así lograr por una parte calidad en el *software* y por la otra el mejoramiento de los procesos.

CMM es...

- Una estrategia de mejora
- Una señalización de deficiencias dentro de una organización
- Una guía para poder avanzar hacia una cultura de calidad

CMM no es...

- Una solución rápida, sino gradual
- Una lista de verificación que puede ser utilizada en todos los ambientes, aunque las prácticas detalladas en el CMM sirven como guía para tomar decisiones

Para hablar de organizaciones maduras en lo que a CMM se refiere, la tabla II ilustra las diferencias principales más sobresalientes entre una organización madura y una inmadura.

Tabla II. Organización madura contra una organización inmadura

Organización inmadura	Organización madura
Improvisa o no sigue rigurosamente los procesos de *software*.	Tiene definido e implantado el método de desarrollo y mantenimiento de *software*.
Improvisa o no emplea la gerencia de proyectos.	Los procesos técnicos y gerenciales están establecidos, son comunicados a toda la organización y se exige su aplicación.
Actúa en respuesta a las crisis que surjan.	Los roles y responsabilidades de los grupos y sus miembros están claramente definidos.
No se hacen estimaciones de costos y tiempo reales.	Las estimaciones de costos y tiempos se basan en experiencias anteriores, reales y cuantificadas.
La calidad del producto no es definida sobre una base objetiva.	Existen objetivos cuantificables para medir la calidad del producto.
No se puede predecir la calidad del producto.	Se controla la calidad del producto y se garantiza la satisfacción del cliente.

Ahora se puede hablar del proceso maduro y el proceso eficaz de *software*, se puede pensar que se orientan de la misma manera pero si los dos tipos de procesos son evaluados dentro de su propio contexto la diferencia radica en que la madurez del proceso de desarrollo de *software* se puede evaluar comparándolo a un modelo (ejemplo: CMM) y la efectividad de un proceso de desarrollo de *software* se puede determinar solo con respecto a los

objetivos del negocio de la organización. Cabe resaltar que la calidad de un producto de *software* está fuertemente determinada por la calidad del proceso implementado para su desarrollo y mantenimiento.

Un proceso de desarrollo de *software* puede considerarse maduro si cumple con los siguientes criterios, ver tabla III.

Tabla III. Características de un proceso de desarrollo de s*oftware* maduro

Está definido	El proceso de desarrollo de *software* es claro, sistemático y suficientemente detallado. Además, existe acuerdo entre el personal, la gerencia y los proyectos respecto al proceso de desarrollo que se va a utilizar.
Está documentado	Esta escrito en un procedimiento publicado, aprobado y fácilmente accesible. Una de las mejores maneras es a través de una intranet para apoyar los proyectos de desarrollo de *software*.
El personal ha sido entrenado en el proceso	Los ingenieros de *software* y la gerencia han recibido cursos y entrenamiento en cada proceso que aplica a su trabajo.
Es practicado	El proceso de desarrollo de *software* definido debe ser utilizado en las tareas habituales llevadas a cabo por los proyectos. El entrenamiento y la adaptación del proceso de desarrollo de *software* a la realidad de la empresa debieran garantizar su aplicación en la vida real.

Es apoyado	La gerencia no sólo debe firmar y promover los procesos de desarrollo definidos, sino que además debe asignar responsabilidad al personal y a los jefes de proyecto por su cumplimiento.
Es mantenido	El proceso de desarrollo de *software* es revisado regularmente, para asegurarse que está adaptado para satisfacer las necesidades reales de los proyectos.
Está controlado	Los cambios y puestas al día del proceso de desarrollo de *software* son revisados, aprobados y comunicados oportunamente a todos los usuarios.
Se verifica	La gerencia mantiene mecanismos para asegurarse que todos los proyectos siguen el proceso de desarrollo vigente.
Se valida	Se asegura que el proceso mantiene concordancia con los requerimientos y estándares aplicables.
Se mide	La utilización, los beneficios y el rendimiento resultante del proceso de desarrollo de *software* se miden regularmente.
Puede mejorarse	Existen mecanismos y apoyo de la gerencia para revisar e introducir cambios en el proceso de desarrollo de *software*, de manera de mejorar su eficacia e incorporar nuevas metodologías.

4.2. Estructura y niveles

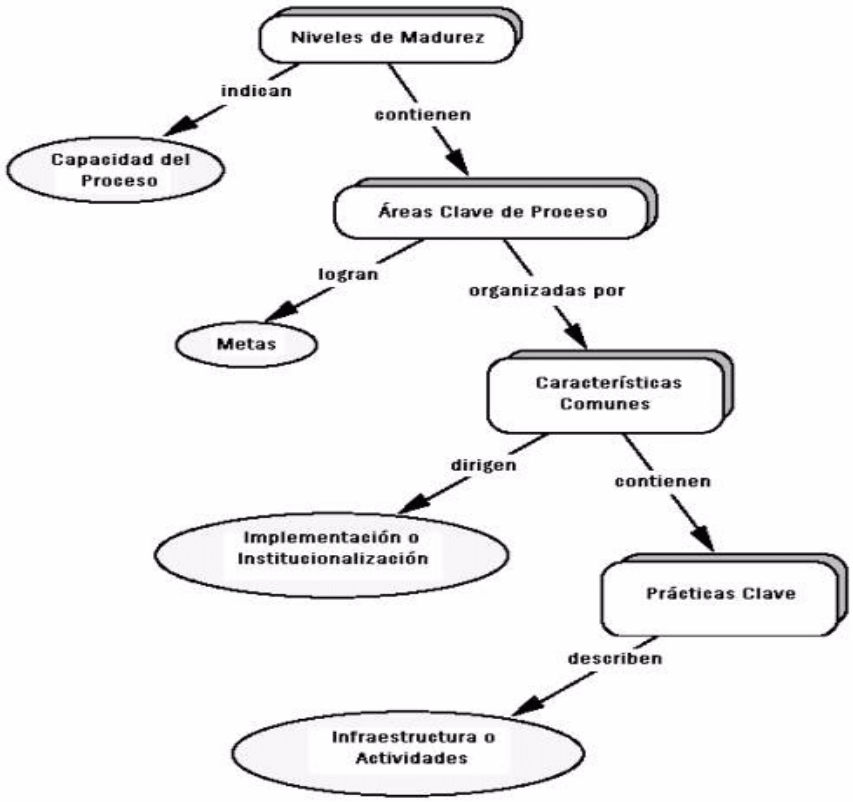
Se piensa en CMM como un modelo basado en el conocimiento, en donde si no se aprende a caminar jamás se podrá llegar a correr. CMM cuenta con cinco niveles de madurez, niveles que son progresivos y no autónomos. Estos niveles están organizados conforme a su importancia (prioridad) en donde uno es base para alcanzar otro y poder así escalar los niveles.

Una organización está formada por todo un equipo de personas. Y sólo involucrando e informando a cada una de ellas es que podremos transmitir, mejorar y aumentar el conocimiento ya adquirido.

Para tener claro como es la estructura del CMM se realizará una comparación. Imaginemos un edificio, cada piso de él representa una plataforma bien definida (esta es el nivel de madurez). Al tener ambos pies en el piso podremos intentar alcanzar el siguiente. Es decir uno a la vez, sin saltos. Ahora, los pisos de este edificio pueden ser muy altos por lo tanto necesitaremos de unas plataformas intermedias para poder ayudarnos a alcanzar el siguiente piso (en este caso nuestra próxima meta) A estas plataformas intermedias las llamaremos Áreas Clave del Proceso o ACP (estas serán nuestras metas intermedias). Para subir entre plataformas intermedias se necesitan escalones (prácticas clave). El número de escalones entre cada plataforma no es fijo. Esas prácticas clave podrían ser muchas así que por conveniencia las dividiremos en cinco grupos, de acuerdo con su objetivo. Estos grupos se llaman características comunes. Los últimos dos elementos nos permitirán transmitir el logro de nuestro avance en todo el personal involucrado.

Ahora en la estructura del CMM nos encontramos con los niveles de madurez. Estos nos indican la capacidad del proceso. Los niveles contienen áreas clave de proceso (ACP) que nos permitirán alcanzar ciertas metas consideradas importantes para la mejora del proceso. Las ACP se encuentran organizadas en cinco distintas características comunes, las cuales buscan la implementación o la institucionalización de las ACP. Estas características contienen a su vez prácticas clave que describen la infraestructura o las actividades que se deben realizar para satisfacer a determinada ACP (ver figura 8).

Figura 8. Estructura del CMM

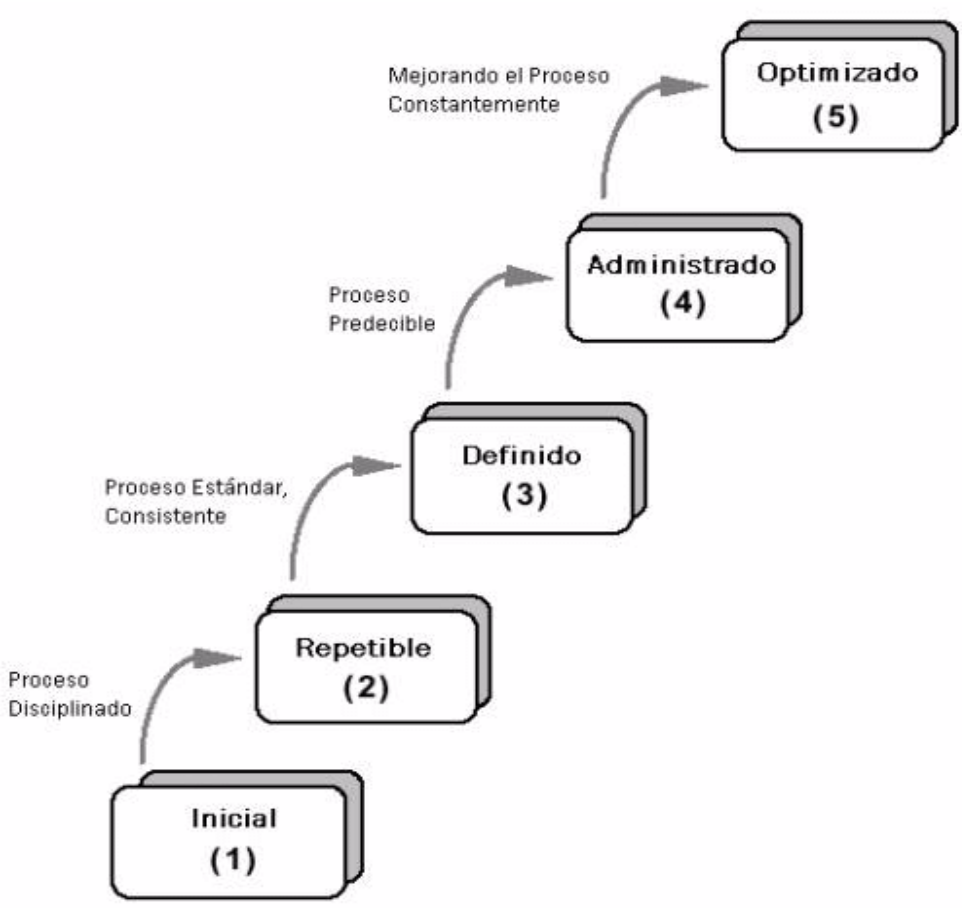


Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

4.2.1. Niveles de madurez

La estructura de más alto nivel del CMM son los cinco niveles de madurez. Recordemos la definición de nivel de madurez: plataforma bien definida desde la cual podremos obtener un proceso definido, administrado, medido, controlado y efectivo de *software*. Cada uno de estos niveles nos indicará que tan capaz es un proceso. Gracias a esto podremos determinar el resultado del próximo proyecto que la empresa decida realizar, en la figura 9 se muestran los niveles de madurez de un proceso de *software*.

Figura 9. Niveles de madurez de los procesos de *software*



Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

4.2.1.1. Nivel 1. Inicial

El primer nivel es el Inicial, que en realidad es el punto base sin valor. Una empresa estará ubicada en el nivel inicial si su proceso es caótico. Esto quiere decir que realmente no existe un ambiente estable en el cual se pueda desarrollar o mantener *software*. En este nivel tendremos un número de entradas, seguidas por cierto proceso que realmente no estaba documentado, pocas veces planificado y para nada medido. Esto último está representado en la figura 10.

Figura 10. Visibilidad y madurez del proceso de desarrollo de *software* inicial

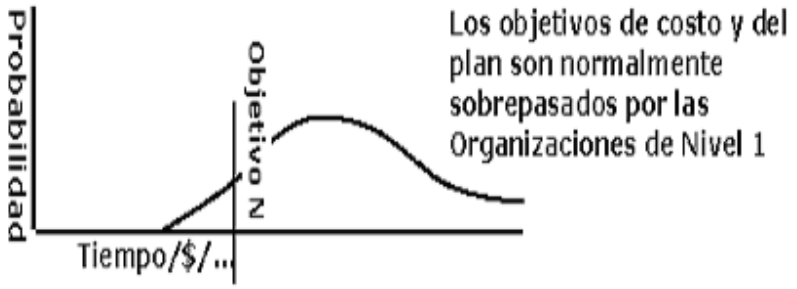
Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

En este nivel es frecuente no alcanzar las metas definidas ni en tiempo, ni costo, ni recursos planeados. Y si esto llegara a ocurrir, es porque dentro del equipo hay personas que planifican y tienen disciplina para cumplir con los objetivos. Suponiendo que con este esfuerzo se logra tener éxito en el proyecto, lo más probable será que éste éxito no se pueda repetir a menos que las mismas personas se encuentren involucradas nuevamente.

Supongamos que dentro de nuestra estimación en cuanto a tiempo y costo definimos un objetivo n. Dentro del nivel 1 la probabilidad de que cumplamos ese objetivo es casi nula. La figura 11 nos muestra como la mayoría

de las veces el objetivo n es sobrepasado. También podemos notar que el área estimada es muy dispersa, es decir no muy certera.

Figura 11. Capacidad del proceso en el nivel 1



Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

La figura 12 nos muestra otra forma de cómo se maneja el proceso dentro de las organizaciones de nivel 1. Se tiene como entrada las necesidades y requerimientos del cliente. Estas pasan por una transformación no definida para dar como resultado un producto. Ahora este producto puede o no satisfacer las necesidades del cliente y puede o no haber cumplido con los estimados propuestos por la organización que otorga el servicio. Si las cumple puede que no se le dé el seguimiento necesario porque los requerimientos son cambiantes, entonces, va a llegar un momento en que estas necesidades no van a ser cubiertas y entonces se va degradando el producto hasta que pierde su valor.

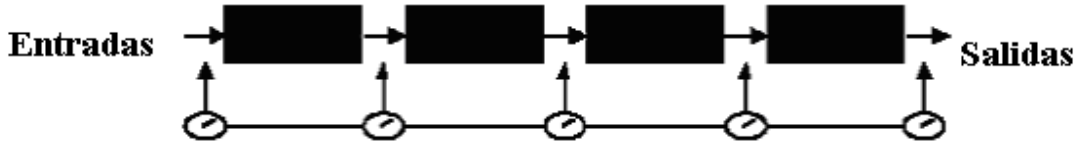
Figura 12. Nivel 1

Conclusión: la capacidad es una cualidad de las personas más no de la organización. Se alcanza el propósito del proceso de manera inconsistente. No es planeado ni lleva un seguimiento.

4.2.1.2. Nivel 2. Repetible

Para dejar atrás el nivel 1 la organización debe empezar a documentar su proceso, empezamos a guardar información. Así pues, si una empresa cuenta con políticas que le permitan administrar un proyecto de *software* y a su vez cuenta con procedimientos para verificar que esas políticas son implementadas, se ubicaría en el nivel dos. Lo que nos muestra la figura 13 es una entrada, que aunque pasa por ciertas cajas negras (ignoramos lo que pasa dentro de ellas), al menos ya son cajas definidas. Al término de cada caja tenemos una revisión donde se podrá detectar si el proceso está funcionando. Se puede actuar ante los problemas en cuanto se sabe de ellos. El cliente puede saber acerca del estatus del proyecto al finalizar cada una de las cajas.

Figura 13. Visibilidad y madurez del proceso de desarrollo de *software* nivel 2

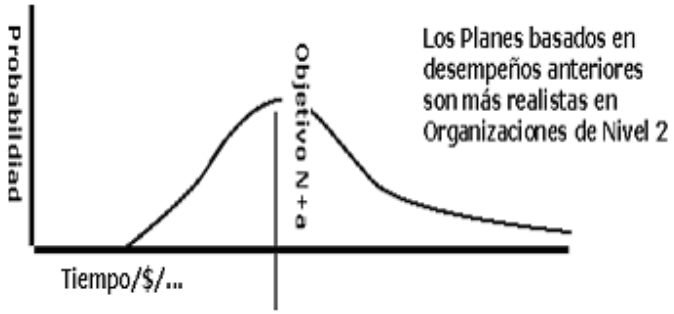


Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

El nombre de este nivel nos da una idea de la meta principal para poder ubicarnos en este nivel. El ser repetible, se refiere a poder repetir éxitos anteriores. Lo cual implica que debemos tener ya cierta experiencia y una gama de proyectos previos donde el éxito ha sido obtenido. Los proyectos entonces implementan procesos efectivos, que están bien definidos, documentados, practicados, entrenados, medidos, se llevan a cabo pero que pueden ser mejorados.

En este nivel los compromisos definidos en cuanto tiempo, costo y recursos son más factibles, reales. Para lograr esto es necesario llevar un recuento en cada proyecto acerca de costos, tiempo y funcionalidad. Si se presentara el caso, los problemas para cumplir estos compromisos son detectados en su momento. En la figura 14 podemos notar que el objetivo n ha sido recorrido (en a unidades) hacia la derecha a diferencia del nivel 1, lo que significa que nos tardaremos más tiempo, nos costará más. Esto podría causar una mala impresión, parecería que no hubo mejora sino al contrario que hubo pérdida. Pero la realidad es que lo que logramos en el nivel 2 fue ser más precisos en cuanto a nuestra estimación. En esta ocasión la estimación es más factible, y como resultado es más probable que la alcancemos.

Figura 14. Capacidad del proceso nivel 2

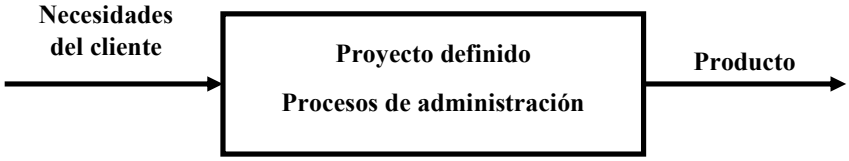


Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

Los requerimientos de *software* y sus productos desarrollados están bien definidos en este nivel. Es necesario tener políticas a nivel organización que permitan llevar a cabo los proyectos con los mejores procesos de administración. Pero, aún los procesos entre proyecto y proyecto podrán variar.

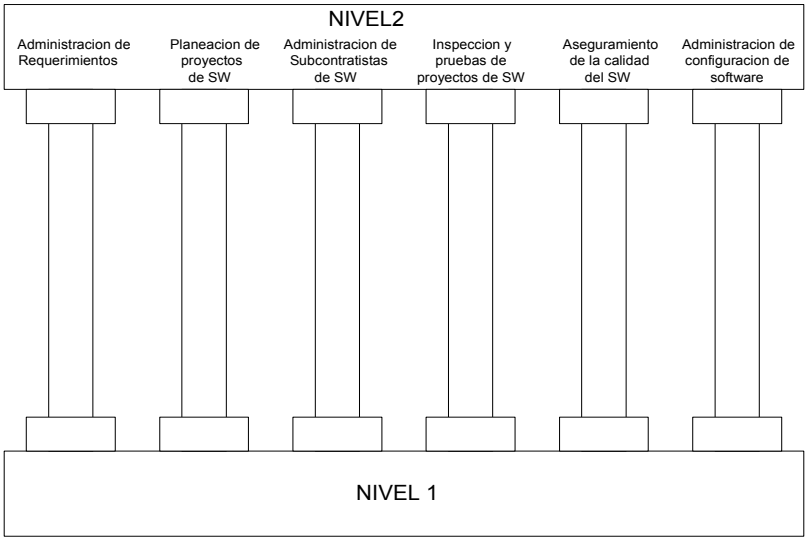
La figura 15 señala ya un proyecto definido junto con procesos de administración. Como salida obtenemos un producto bien definido.

Figura 15. Nivel 2



Conclusión: la capacidad del proceso de desarrollo de *software* de una organización nivel 2 tiene disciplina, ya que el proyecto de *software* involucra planeación y seguimiento; es un proceso documentado. El proceso es estable y los éxitos anteriores pueden repetirse. Pero aún no contamos con métricas para servicios, solamente para productos.

Figura 16. Pilares del nivel 2



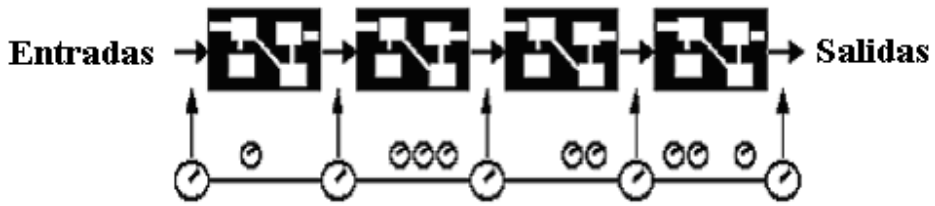
Si bien es cierto que hay empresas que pueden obtener los beneficios que conlleva encontrarse en el nivel 2, solo es por la práctica y experiencia de unos cuantos que saben cumplir las metas organizacionales y al final no saber ni tener documentado ese logro. Alcanzar el nivel 2 (figura 16) requiere de los pilares de la planeación de recursos y proyectos de *software*, del control de calidad estricto y constante, de la gestión y seguimiento de los requerimientos.

No obstante, no basta con llegar a la meta, cada pilar sostiene el siguiente nivel, si uno de ellos se deteriora todo el proyecto llega al caos, en el cual se pierden los recursos más valiosos. Porque no entonces, repetir éxitos pasados para obtener los presentes y, porque no decirlo, futuros.

4.2.1.3. Nivel 3. Definido

En este nivel nos olvidamos de las cajas negras, se cuenta con un proceso de desarrollo de *software* estándar de la organización para desarrollar o mantener el *software*. Éste está documentado y es implementado a lo largo de toda la organización en distintos proyectos. Este proceso es la unión de prácticas de ingeniería de *software* y de administración de procesos. La figura 17 nos muestra el interior de estas cajas. Como consecuencia se incrementa el número de accesos para conocer la situación del proyecto tanto por la organización como por el cliente.

Figura 17. Visibilidad y madurez del proceso de desarrollo de *software* nivel 3



Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

Algunos puntos clave del nivel 3 son:

1. Los procesos se implementan y actualizan para ayudar a los administradores de proyectos y equipo técnico a desempeñarse más efectivamente

2. Para estandarizar se utilizan prácticas de ingeniería de *software*

3. Un grupo dentro de la organización está encargado de las actividades del proceso de desarrollo de *software*.

4. La organización cuenta con un programa de capacitación para que todos los miembros de la organización cuenten con el conocimiento y las habilidades requeridas para desempeñar completamente sus roles.

5. El proceso de desarrollo de *software* estándar de la organización se amolda a cada proyecto para así determinar el proceso de desarrollo de *software* definido del proyecto (su propio proceso de *software*) Este contendrá información acerca de cómo saber que ya esta listo el producto, entradas, estándares y procedimientos para desarrollar el trabajo, mecanismos de verificación, salidas y un criterio de terminación.

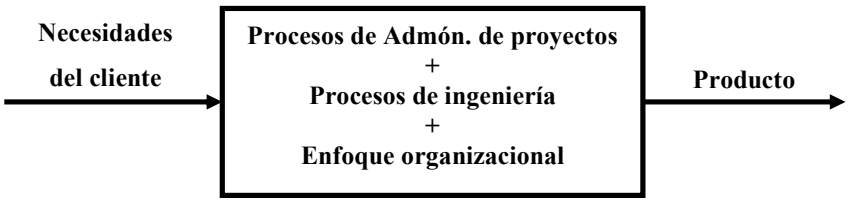
En el nivel 3 se recorre nuestro objetivo un poco a la izquierda lo cual indica reducción en tiempo, costo y otras variantes. La probabilidad de acertar en nuestro cálculo es muy alta (figura 18).

Figura 18. Capacidad de proceso nivel 3

Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

En el nivel 3 agregamos a nuestro proceso un enfoque organizacional y procesos de ingeniería, como lo podemos visualizar en la figura 19.

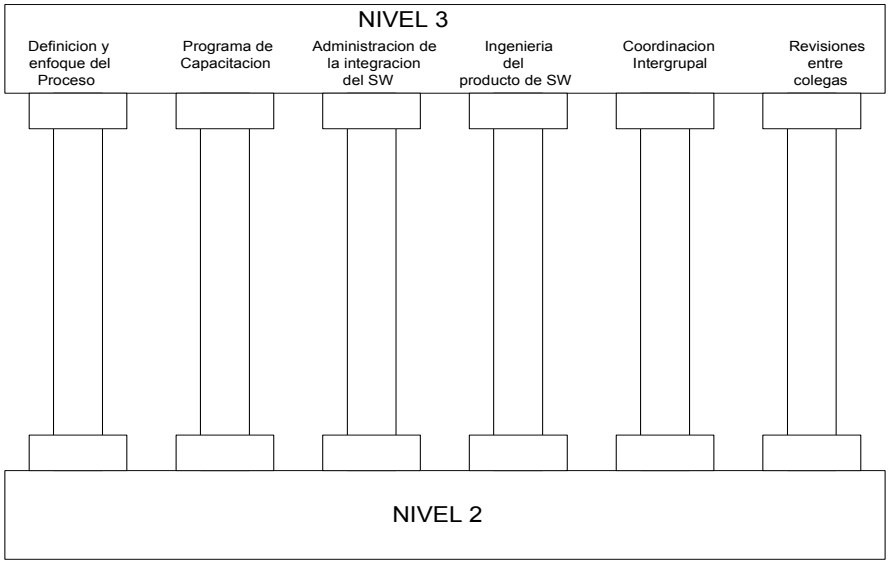
Figura 19. Nivel 3



Conclusión: el nivel 3 es estándar y consistente, ya que gracias a las prácticas de ingeniería de *software* y a las de administración de proyectos el proceso es estable y repetible. La capacidad se logra basándose en el entendimiento de las actividades, roles y responsabilidades en un proceso de desarrollo de *software* bien definido. La administración ahora puede prepararse con anterioridad para afrontar riesgos posibles. En este nivel se cuenta con planes y programas de mejora aunque no necesariamente se les da un seguimiento. La medición no es sólo de productos sino también de servicios.

Los pilares en el nivel 3 (figura 20) sostienen un proceso definido para desarrollar *software* de calidad y que cumpla con los requerimientos, pero si se trata de mantenerlo debe ser innovador y que actúe en respuesta a los cambios radicales y no tan radicales. Planear la capacitación de los usuarios con diferentes roles que interactúan directa o indirectamente en el proceso de desarrollo y producción. Si la base son los pilares del nivel anterior hay que mantenerlos y sobre ellos construir los del nivel 3.

Figura 20. Pilares del nivel 3



4.2.1.4. Nivel 4. Administrado

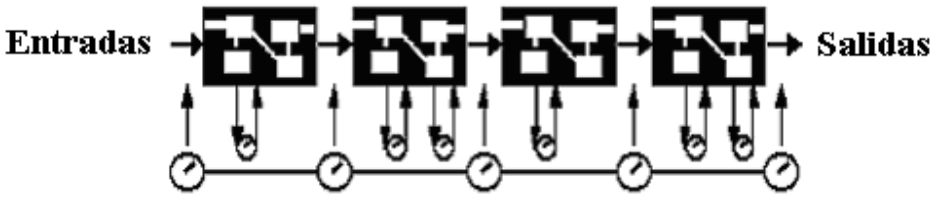
En el nivel 4 hacemos uso de todos los datos que hemos recolectado. Convertimos datos en información relevante para la organización para así identificar qué estaba mal.

Este nivel podría llamarse cuantitativo ya que en él cualquier decisión es respaldada por una base cuantitativa. Medimos el progreso y los problemas. Y mientras aumentamos la probabilidad de ser más precisos en nuestros estimados, reducimos la variabilidad (incertidumbre) de nuestro proceso. El cliente tendrá un entendimiento medible tanto de la capacidad del proceso como del riesgo que éste implica, incluso antes de que el proyecto inicie.

En este nivel la organización fija metas de calidad tanto del proceso como del producto. Existe un programa de medición dentro de la organización que es aplicado a lo largo de todos los proyectos, midiendo así la productividad y la calidad.

Al mismo tiempo, la organización cuenta con un repositorio de información donde almacena información relevante de proyectos anteriores que podría reutilizarse en proyectos futuros. En la figura 21 se logra apreciar que no sólo se toma información del proyecto sino que también se ingresa información; existe una retroalimentación.

Figura 21. Visibilidad y madurez del proceso de desarrollo de *software* nivel 4



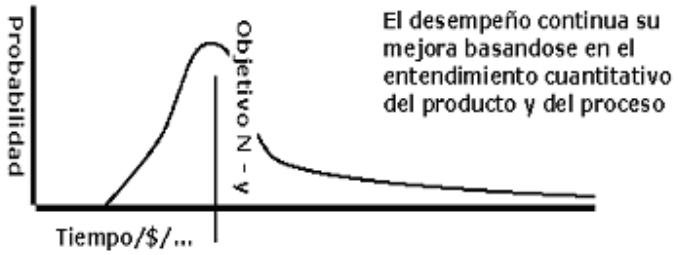
Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

Todos los procesos de *software* son medidos. En este nivel formamos la parte cuantitativa de la organización para poder evaluar los proyectos, los procesos y los productos. Esto no quiere decir que en niveles anteriores no se obtuvo información cuantitativa. Puede ser que sí se guardo este tipo de información, pero en el nivel 4 se le da un significado a esta información valiosa.

Todas esas mediciones nos marcan distintos límites (inferiores y superiores) de como debería llevarse a cabo nuestro proyecto. Si nuestros números están fuera de ese rango de valores posibles (dado por proyectos anteriores) podremos identificar ya sea mejores prácticas o, en su caso, malas.

En la figura 22 observamos que seguimos reduciendo nuestro objetivo y la probabilidad de ser certeros es aun mayor.

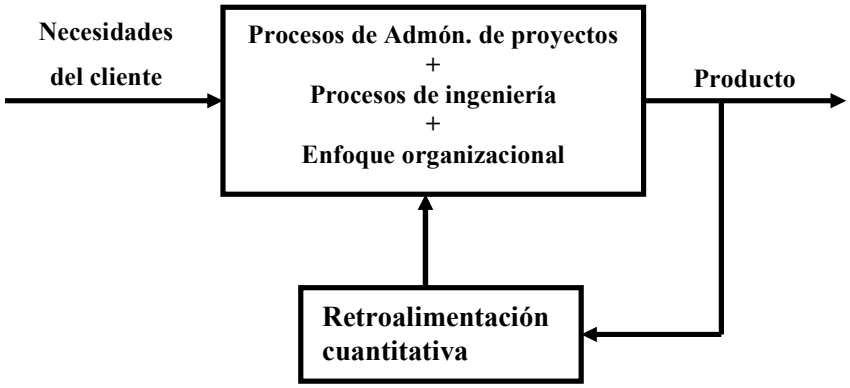
Figura 22. Capacidad del proceso nivel 4



Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

En el nivel 4 agregamos una retroalimentación cuantitativa a nuestro proceso. Esto lo podemos observar en la figura 23.

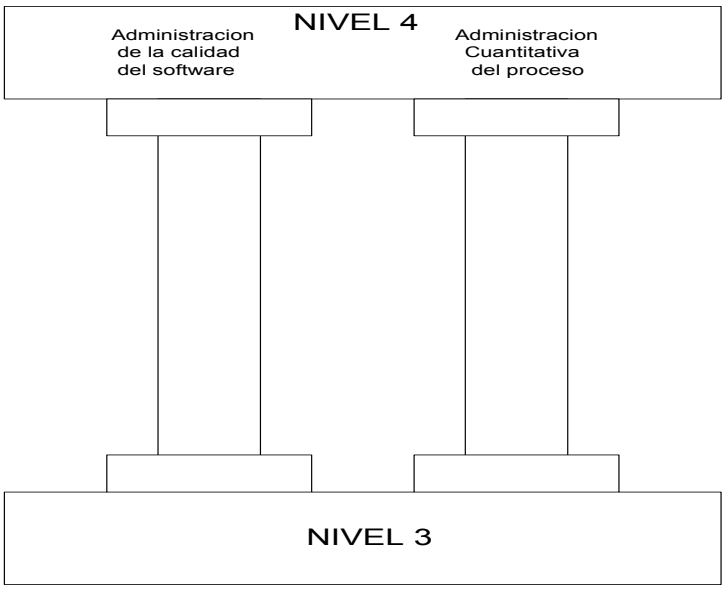
Figura 23. Nivel 4



Conclusión: el nivel 4 es cuantificable y predecible. Todo gracias a que el proceso, a la par que el producto y servicios, es medido y opera dentro de un límite cuantificable. Se cumple con planes y programas de mejora. Se aplica administración de calidad. Se hace una distinción entre los procesos principales

y los de apoyo. Se pueden observar cadenas cliente-servidor a lo largo del proceso. Como resultado obtenemos un producto de alta calidad, con seguimiento y mejoras.

Figura 24. Pilares del nivel 4

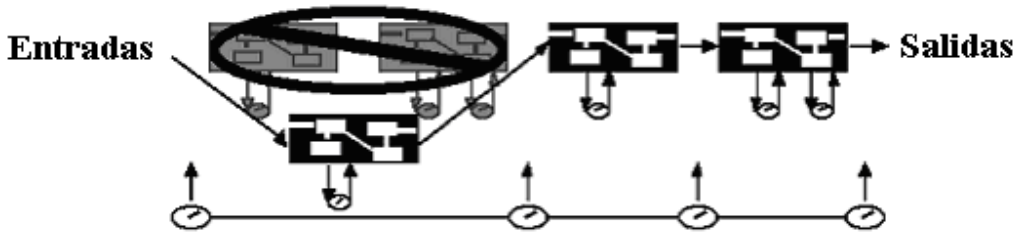


Pocas son las organizaciones que llevan un control estricto de calidad y que tienen un grupo especializado para ello, regularmente el personal de desarrollo es el que lleva este control. Este nivel (figura 24) tiene un pilar muy importante que es la administración de la calidad del *software*; por lo tanto, las organizaciones que deseen estar en este nivel deben invertir en un grupo especializado en control de calidad, que a la larga es más beneficioso, puesto que se invierte en calidad para ofrecer calidad; se implementa y crece la administración cuantitativa del proceso de desarrollo de *software*, que se produce en cantidad y calidad.

4.2.1.5. Nivel 5. Optimizado

En el nivel 5 la organización se dedica a mejorar continuamente su proceso probando nuevas maneras de construcción de *software*, pero siempre de una manera controlada. Para lograr esto es necesario localizar tanto debilidades como fortalezas. Al analizarlas podremos mejorar nuestro proceso y así prevenir defectos. Si un defecto es encontrado, entonces es revisado o sustituido, como lo podemos observar en la figura 25. Los defectos son analizados con el fin de determinar sus causas. El objetivo de esta actividad es el prevenir y evitar la recurrencia de estos defectos. Una vez encontradas las causas es importante difundirlas para que así todos los integrantes de la organización aprendan la lección.

Figura 25. Visibilidad y madurez del proceso de desarrollo de *software* nivel 5



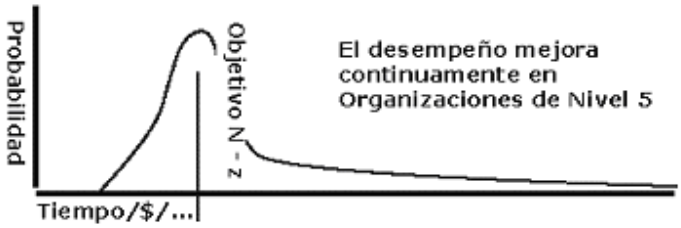
Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

En el nivel 5 los directivos son capaces de estimar y después dar un seguimiento cuantitativo al impacto y a la efectividad del cambio. La relación con el cliente es más fuerte.

A partir de la eficiencia de nuestro proceso es posible generar reportes de costo/beneficio de nuevas tecnologías o proponer cambios al proceso estándar de la organización.

El tiempo de re-trabajo se reduce gracias a que se pueden identificar casos problemáticos, donde los proyectos se salen de un rango de valores válidos; determinado por la medición. Es probable que el equipo ni siquiera sepa que está en problemas pero la experiencia pasada se los hará saber. Así podrán identificar el origen de su falla y corregirla antes de que sea demasiado costoso para el proyecto. Una vez más, nuestro objetivo de reducir costos y producir *software* en tiempo es mejorado, ver figura 26.

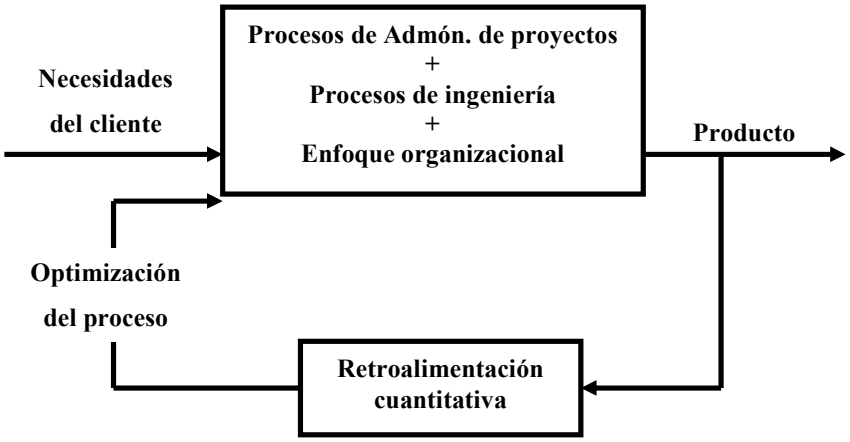
Figura 26. Capacidad de proceso nivel 5



Fuente: SEI, www.sei.cmu.edu/cmm/cmm.html

Es en este nivel 5 que agregamos una fase más, la optimización del proceso. Así se cierra el círculo de nuestro proceso. Gráficamente se muestra en la figura 27.

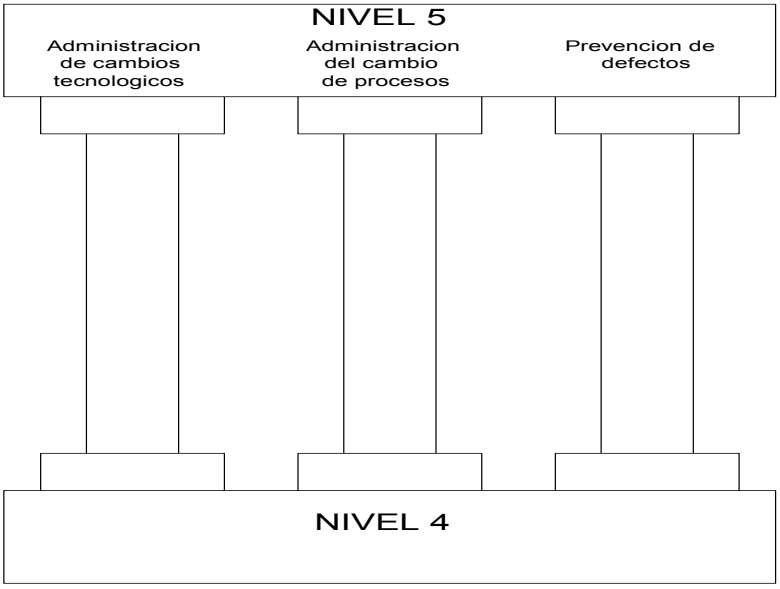
Figura 27. Nivel 5



Conclusión: el nivel optimizado se dedica al mejoramiento continuo de su proceso (capacidad) a la par de su madurez. Este mejoramiento se da gracias al uso o implementación de nuevas tecnologías o métodos. Obtenemos un cumplimiento total de objetivos de calidad. Los ciclos de mejora continua son identificables. Los indicadores de desempeño competitivos son comparados contra los mejores en su clase. Existe una relación causal entre la mejora de calidad y el desempeño financiero de la compañía.

Prevenir los defectos es un pilar muy frágil (figura 28) puesto que nada llega a ser perfecto, pero esta debe ser una meta constante y consciente, porque un defecto se nota más cuando tiene pocas probabilidades de ocurrir y entonces este influye y hace más grande el problema; y que decir de la administración de los cambios, si vamos a innovar vamos a cambiar pero hay que hacerlo de una manera organizada, documentada, bien planificada y manteniendo los pilares que se encuentran sosteniendo la estructura final que es el CMM.

Figura 28. Pilares del nivel 5



Cuando una organización alcanza el nivel 5 no significa que ya no se puedan proponer metas más altas, que no se fijen nuevos horizontes; es más, si la organización no persiste en su mejoramiento continuo ésta podría bajar a un nivel inferior de la escala de CMM. Por lo tanto, si lo ganado no se cuida, ni se mantiene, se pierde.

4.3. Definición de Áreas Clave del Proceso (ACP)

El CMM fue creado pensando completamente en el desarrollo y manutención de *software*. Al armar este modelo se reunieron las prácticas de Ingeniería de *software* más exitosas. Se definieron un conjunto de áreas clave del proceso para poder lograr un mejoramiento en el proceso de desarrollo de *software*.

Un nivel de madurez será obtenido si y sólo si todas las ACP comprendidas por él han sido cubiertas satisfactoriamente. Ahora bien, para saber si una ACP ha sido cubierta basta con que las metas especificadas en ella se hayan alcanzado.

El único nivel de madurez que no tiene ACP a cubrir es el nivel Inicial. Es entonces a partir del nivel 2, el repetible, que se definen un conjunto de ACP por nivel. A continuación se listan todas las áreas clave del proceso que están distribuidas a lo largo de los niveles 2 al 5:

- Administración de requerimientos
- Planeación de proyectos de *software*
- Administración de subcontratistas de *software*
- Inspección y pruebas de proyectos de *software*
- Aseguramiento de la calidad del *software*
- Administración de la configuración de *software*
- Enfoque en el proceso de la organización
- Definición del proceso de la organización
- Programa de capacitación
- Administración de la integración del *software*
- Ingeniería del producto de *software*
- Coordinación intergrupal
- Revisiones entre colegas
- Administración cuantitativa del proceso
- Administración de la calidad de *software*
- Administración de cambios tecnológicos
- Administración del cambio de procesos
- Prevención de defectos

Más adelante, en este mismo capítulo, se ubicarán las áreas clave del proceso en los cinco niveles del CMM.

4.4. Roles y grupos

Para escalar los niveles de madurez es necesario aumentar la capacidad de nuestra organización. Es decir, el rango de resultados esperados que se pueden obtener tras seguir un proceso debe reducirse. Rangos en cuanto tiempo, costos y recursos.

Cada rol (una persona puede jugar varios roles, dependiendo del tamaño del proyecto u organización) deberá cumplir con sus responsabilidades, a continuación la definición de los roles y grupos esenciales en un proyecto de desarrollo de *software*:

Roles

- Debe existir un compromiso por parte de la gerencia general.
- Los gerentes deberán planear, organizar, dirigir, y controlar el trabajo dentro de su área.
- Los líderes de proyecto deberán dirigir, controlar, administrar, y regular un proyecto para construir sistemas de *software* o *hardware*/*software*. Es el responsable ante el cliente.
- El Líder de proyectos de *software* será el responsable y controlará todas las actividades y recursos de *software* de un proyecto.
- El líder de *software* de primera línea funge como el responsable directo de la administración del equipo de trabajo y actividades de una unidad organizacional (departamento, equipo de proyecto) de ingenieros de

software y equipo de trabajo relacionado. Esto incluye el proveer dirección técnica, administrar el personal y salarios.

- El líder de tareas de *software* cubre el rol de líder de un equipo técnico para una tarea específica, tiene la responsabilidad técnica y provee dirección técnica al equipo de trabajo dedicado a determinada tarea.
- El equipo de trabajo (no gerentes) es responsable de realizar una función asignada, tales como desarrollo de *software* o administración de configuración de *software*.
- El equipo de trabajo de ingeniería de *software* son las personas técnicas de *software* (no gerentes) como analistas, programadores e ingenieros. Ellos realizan el desarrollo de *software* y las actividades de mantenimiento del proyecto.

Grupos

Los grupos básicos que se deben cubrir:

- Grupo de ingeniería de *software*
  - Conformado por: gerentes y equipo técnico.
  - Responsables de: actividades de desarrollo y manutención de *software*.

- Grupo de ingeniería de procesos de *software*
  - Conformado por: especialistas.
  - Responsables de: facilitar la definición, mantenimiento y mejoramiento del proceso de desarrollo de *software* usado por la organización.

- Grupo de ingeniería de sistema
  - Conformado por: gerentes y equipo técnico.

- Responsables de: especificar los requerimientos del sistema, asignar y definir las interfaces de los recursos necesarios y monitorear el diseño y desarrollo de los componentes.

- Grupo de pruebas de sistema
  - Conformado por: gerentes y equipo técnico.
  - Responsables de: planear y realizar las pruebas del sistema del *software* para determinar si es que el producto de *software* satisface sus requerimientos.

- Grupo de aseguramiento de calidad de *software*
  - Conformado por: gerentes y equipo técnico.
  - Responsables de: planear e implementar las actividades de calidad de los proyectos para asegurar que los pasos del proceso de desarrollo de *software* y estándares sean seguidos.

- Grupo de administración de configuración de *software*
  - Conformado por: gerentes y equipo técnico.
  - Responsables de: planear, coordinar, e implementar las actividades de administración de la configuración para los proyectos de *software*.

- Grupo de entrenamiento
  - Conformado por: gerentes y equipo de trabajo.
  - Responsables de: coordinar y arreglar las actividades de entrenamiento de una organización. Preparan y conducen la mayoría de los cursos de entrenamiento y coordinan el uso de otras vías de entrenamiento.

Después de nombrar cada uno de los roles y grupos en los que se basa CMM y de especificar sus responsabilidades, es importante que estos / éstas se cubran.

También es muy cierto que no es necesario que una persona cumpla con sólo un rol, o que forme parte de un solo grupo. El tamaño de la organización determinará esto. Una persona puede cubrir varios roles o ser parte de distintos equipos. Lo importante es que exista alguien responsable de cubrir las actividades de cada uno de los roles o grupos.

4.5. Apreciaciones de procesos de desarrollo de *software*

4.5.1. Valoraciones y evaluaciones

Básicamente, el SEI divide en dos las apreciaciones para identificar la madurez de una organización en la ejecución de su proceso de desarrollo de *software*:

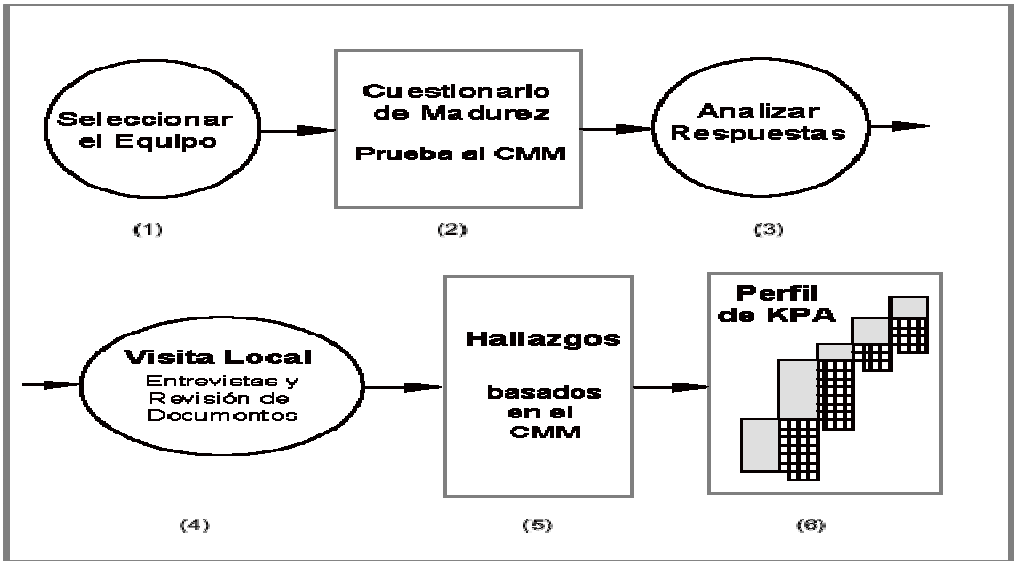
1. Valoración del proceso de desarrollo de *software* (a lo que comúnmente se le llama *Assessment*): las valoraciones del proceso de desarrollo del *software* se ocupan para:

   - Determinar el estado del proceso de desarrollo de *software* actual de la organización.
   - Para determinar los asuntos de mayor prioridad en cuanto al proceso que enfrenta una organización y
   - Para obtener el apoyo de la organización para la mejora del proceso de desarrollo de *software*.

2. Evaluación de la capacidad de *software*: las evaluaciones de la capacidad del *software* se utilizan para:
   - Identificar contratistas que sean calificados para realizar el trabajo de *software* o
   - Para monitorear el estado del proceso de desarrollo de *software* empleado en un esfuerzo existente de *software*

Los pasos comunes tanto para una evaluación como para una valoración se observan en la figura 29.

Figura 29. Pasos comunes para evaluaciones y valoraciones



Fuente: CMM, www.sei.cmu.edu/cmm/cmm.html

Pasos:

1. Seleccionar el equipo, éste deberá estar conformado por asesores reconocidos por el SEI como capaces de realizar evaluaciones o valoraciones según sea el caso. Este grupo de personas cuentan con un alto conocimiento en prácticas de ingeniería de *software*.

2. Localmente, se contesta el cuestionario de madurez, tomando a CMM como una guía para poder realizar esta tarea.

3. El equipo de asesores analiza las respuestas.

4. El equipo de asesores se traslada físicamente a la organización. Durante su estancia se llevan a cabo entrevistas al igual que se hace una revisión de documentos. Todo con el afán de comprender el proceso de desarrollo de *software* que sigue la organización. Si alguna práctica difiriere notablemente de aquellas definidas dentro del CMM, el equipo entrevistado debe explicarlo para sostener la práctica propia.

5. Una vez que toda la información ha sido recabada el equipo evalúa y determina aquellas fortalezas y puntos de mejora para la organización.

6. Por último, se realiza un estudio detallado de las ACP y de las metas que no fueron cubiertas. Es posible que aun cuando una ACP obtenga un puntaje satisfactorio se puedan identificar puntos de mejora.

4.5.2. Métodos de apreciación basados en el CMM

Los métodos de apreciación basados en el CMM son una traducción de *Appraisal Methods*, se les conoce más con este nombre. El SEI proporciona un juego de herramientas para diagnosticar, llamado apreciación, para aquellas organizaciones que desean saber el estado de su propio o el de otro proceso de desarrollo de *software* en relación al CMM para *software* versión 1.1. El término apreciación como es utilizado dentro del SEI incluye tanto las evaluaciones como las valoraciones. Ambas se enfocan en el proceso de desarrollo de desarrollo de *software* de la organización.

Para determinar el estado de los procesos propios, el método apropiado a utilizar es una valoración tal como lo es la Apreciación Basada en CMM para la Mejora Interna del Proceso (CBA IPI, por sus siglas en inglés)**.** Para determinar el estado del proceso de un externo, el método apropiado a implementar es una evaluación, tal como la Evaluación de Capacidad de *Software* (SCE, por sus siglas en inglés).

Para asegurarnos que ambos métodos sean rigurosos y sólidos y el programa del proceso sea desarrollado y mantenido se utiliza el marco de trabajo de la evaluación de CMM (CAF, por sus siglas en inglés). Éste define los estándares que un método de apreciación debe reunir.

El SEI entrena y autoriza a personas calificadas para dirigir las apreciaciones, en otras palabras, en la Apreciación Basada en CMM para la Mejora Interna del Proceso (CBA IPI), y para utilizar los materiales del SEI al conducir las apreciaciones. El SEI no valida ni certifica los resultados de la apreciación.

4.6. Niveles donde se ubican las ACP

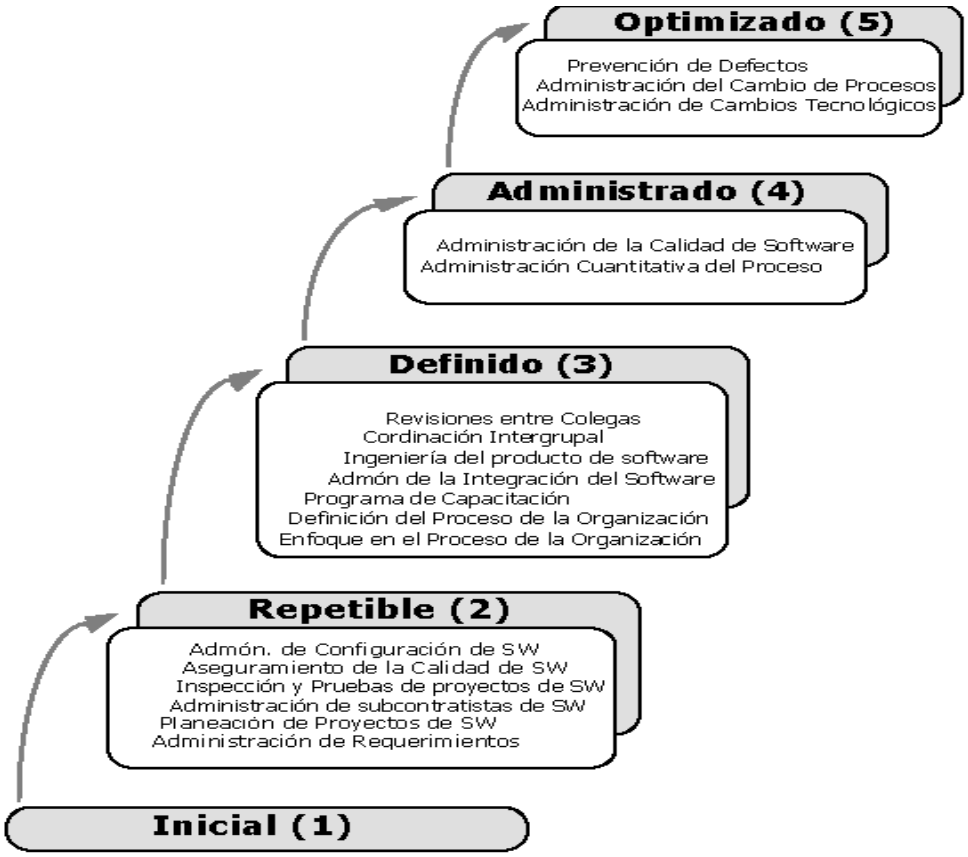
Las áreas clave del proceso están ubicadas a lo largo de los niveles del CMM, como se mencionó anteriormente, el único nivel que no contiene ACP es el inicial, puesto que por ser el nivel de inmadurez y poca administración de los procesos no se le pueden asignar o por lo menos no se pueden utilizar las prácticas de organizaciones cuyo nivel en el CMM es el inicial, pues estas pocas veces conllevan al éxito y este no es repetible como ya se vió anteriormente.

En total son dieciocho ACP que conforma al CMM. Estas ACP pueden ser distribuidas de acuerdo a tres categorías del proceso de desarrollo de *software*:

- Gerencial: se refiere a la planeación del proyecto de *software*, su administración, seguimiento, etc.
- Organizacional: abarca capacitación, infraestructura, cultura, etc.
- Ingeniería de *software*: se refiere al análisis de requerimientos, diseño, codificación, pruebas, implementación, documentación, etc.

En la figura 30 se ubican las áreas clave del proceso en los cuatro principales niveles del CMM, como se sabe, se deben llenar y cumplir todas las áreas establecidas en un nivel para poder avanzar al otro.

Figura 30. ACP por nivel de madurez



Fuente: PAULK, Áreas clave del proceso por nivel de madurez, www.sei.cmu.edu/cmm/cmm.html

De acuerdo con la distribución de las ACP dentro de las categorías del proceso de desarrollo de *software* y la ubicación de las ACP en sus respectivos niveles, a continuación, la tabla IV relaciona todos estos conceptos y los ubica de acuerdo a categorías contra niveles en donde las ACP los unen.

Tabla IV. ACP ubicadas en niveles y categorías organizacionales

Categoría / Nivel	Gerencia	Organizacional	Ingeniería de Software
5		Administración de cambios tecnológicos	
	Administración del cambio de procesos		Prevención de defectos
4	Administración cuantitativa del proceso		Administración de la calidad de software
3	-Administración de la integración del software -Coordinación intergrupal	-Enfoque en el proceso de la organización -Definición del proceso de la organización -Programa de capacitación	-Ingeniería del producto de software -Revisión entre colegas
2	-Admón. de Requerimientos -Planeación de proyectos de software -Seguimiento y supervisión de proyectos de software -Administración de subcontratistas de software -Aseguramiento de la calidad de software -Administración de la configuración de software		
1	Procesos Ad-hoc		

4.6.1. Características comunes en las ACP

Para alcanzar las metas definidas por las ACP, CMM nos brinda un conjunto de prácticas clave. Estas prácticas representan la infraestructura o las actividades que contribuyen en mayor medida en la implementación e institucionalización de las Áreas Clave de Proceso.

Por facilidad CMM agrupa a las prácticas clave en cinco grupos. De acuerdo a CMM versión 1.1, éstas son:

1. **Compromiso**. Describe las acciones que la organización debe realizar para asegurar que el proceso sea establecido y persistirá. Normalmente, involucra el establecimiento de políticas organizacionales y liderazgo.

2. **Habilidades necesarias**. Describe las condiciones previas que deben existir en el proyecto u organización para poder implementar el proceso competentemente. Normalmente, involucra recursos, estructuras organizacionales y capacitación.

3. **Actividades realizadas**. Describe las actividades, roles, y procedimientos necesarios para implementar una ACP. Normalmente involucra el establecer planes y procedimientos, realizar el trabajo, darle seguimiento, y tomar acciones correctivas según sea necesario.

4. **Medición y análisis**. Describe las prácticas básicas de medición que son necesarias para determinar el estado en relación al proceso. Estas mediciones son utilizadas para controlar y mejorar el proceso. Normalmente incluyen ejemplos de métricas que pueden tomarse.

5. **Verificación e implantación**. Describen los pasos para asegurarse que las actividades son llevadas a cabo de acuerdo con el proceso que se ha establecido. Normalmente abarca revisiones y auditorias por parte de la administración y aseguramiento de la calidad de *software*.

4.7. Beneficios

Recordemos que CMM es una herramienta que ayuda a las organizaciones de *software* para mejorar sus procesos. Por igual, ayuda a organizaciones que se dedican a la adquisición, a seleccionar o administrar a sus contratistas de *software*.

El propósito de CMM es describir buenas prácticas de administración y de ingeniería estructuradas por un marco maduro de trabajo.

Algunos de los beneficios de utilizar CMM como marco de trabajo para un mejoramiento continuo son:

1. CMM ayuda a formar una visión compartida lo que significa mejoramiento de procesos a nivel organizacional.
2. Establece un lenguaje común al hablar acerca del proceso de desarrollo de *software*.
3. Define un conjunto de prioridades para atacar los problemas de *software*.
4. Provee una estructura conceptual para mejorar la administración y desarrollo de productos de *software* en una manera disciplinada y consistente.
5. Aumenta la posibilidad de que una organización de *software* alcance sus metas de costos, calidad y productividad de una manera disciplinada y consistente.

CMM identifica las características de un proceso efectivo de *software* pero, la organización madura atiende los asuntos esenciales para un proyecto exitoso, incluyendo personal y tecnología, al igual que al proceso de desarrollo de *software*.

Para interpretar correctamente CMM en un ambiente específico, es necesario utilizar inteligencia, experiencia y conocimientos. Esa interpretación debe basarse en las necesidades de negocio, los objetivos de la organización y los proyectos.

Para ser más específicos, al aplicar CMM y a razón que la organización acrecienta su madurez es notorio el avance en la posibilidad de predecir, en el control, y en la eficiencia.

La primera mejora que se presenta es la posibilidad de predecir. Con esto nos referimos a que a lo largo que aumenta la madurez en una organización, la definición de resultados objetivo se vuelve más certera. Es decir, los resultados planeados se asemejan más a los resultados reales.

También se observa una mejora en el control. Aumenta el control que se tiene alrededor de los resultados objetivo. Es decir, si se presentara un proyecto similar a uno ya enfrentado es probable que se ocupe la experiencia ya adquirida y así reducir el rango (tiempo, costo, entre otros) de entrega.

Otra mejora es la eficiencia, ya que los resultados objetivo se perfeccionan si la madurez de la organización se incrementa. Esto se ve reflejado en distintos campos. El tiempo de desarrollo se acorta, los costos se reducen, la calidad y productividad aumentan, ya que al ocupar procesos más

efectivos, se reduce el número de errores generados. Al aprovechar tiempo en re-trabajo (arreglando esos errores) se reduce el tiempo de desarrollo.

4.8. Diferencias entre las versiones de CMM

Como se mencionó en el capítulo 1 el CMM ha sufrido varias transiciones, mientras que aparezcan desventajas va a seguir mejorando, si CMM es un producto que involucra calidad se debe aplicar este concepto a su estructura para que los críticos vean sus ventajas y desventajas.

Primero salió al mercado la versión 1.0, este modelo contenía 4 niveles de madurez, la siguiente versión de CMM agregó uno más para completarlo y redefinir las áreas clave del proceso. Esta versión, CMM 1.1 es la que se encuentra en el mercado. El SEI propone la mejora continua y la innovación tecnológica, este último punto no estaba involucrado explícitamente en la versión 1.1, se analiza al modelo para seguir con la cultura de calidad y mejoras. Entonces surgió la siguiente versión aun no liberada 2.0. Esta versión se encuentra en borrador a la mano de empresas certificadas, empresas no certificadas en CMM, de críticos y apoyadores hasta que la misma sea ya lo suficientemente fuerte para salir al mercado. En la tabla V vemos las diferencias de la versión en el mercado y la versión borrador.

Tabla V. Comparación entre versiones del SW-CMM

Nivel	SW-CMM v1.1	SW-CMM v2 borrador
2	Administración de subcontratistas de *software*	Administración de la adquisición de *software*
2	Inspección y pruebas de proyectos de *software*.	Control de proyectos de *software*
3	Programa de capacitación	Organización del programa de capacitación
3	Coordinación intergrupal	Coordinación de la interfase del proyecto
4	Administración cuantitativa del proceso	Administración estadística de los procesos
5	Administración de cambios tecnológicos	Organización del proceso e innovación tecnológica
5	Administración del cambio de procesos	Organización de la utilización de mejoras

Se movieron algunas áreas clave del proceso entre los cinco niveles del CMM, como:

1. Enfoque del proceso organizacional del nivel 3 al 2.
2. Programa de capacitación del nivel 3 al 2.
3. Revisiones entre colegas del nivel 3 al 2.
4. Administración del cambio de procesos del nivel 5 al 2.

Se agregaron nuevas áreas clave, prácticas comunes y cambios como:

- La administración de los riesgos agregada en las prácticas comunes, esta es una meta del área clave **ingeniería del producto de *software***
- Servicio al cliente en el área clave **ingeniería del producto de *software*** y en las prácticas

- Verificación y validación de pruebas, administración de las pruebas como prácticas comunes
- Reutilización de *software*, reingeniería, líneas de producto en el área clave **organización de los activos compartidos del *software*** dentro del nivel 4
- Ingeniería concurrente, integración del producto desarrollado, en el área clave que fue rescrita como **coordinación de la interfase del proyecto**
- Combinación de las áreas: **enfoque del proceso de la organización** y **definición del proceso organizacional**
- Reenfocar las áreas clave **programa de entrenamiento** y **construir habilidades**
- Cambio de área **revisiones entre colegas** a **inspecciones**
- Área clave en el nivel 4: **organización del rendimiento del proceso**
- Área clave en el nivel 4: **organización de los activos compartidos del *software***

5. OTROS MODELOS Y ESTÁNDARES

5.1. CMMI

Es un nuevo modelo del SEI que fue creado a solicitud del Departamento de Defensa de los Estados Unidos para las organizaciones con iniciativas de ingeniería de *software*, ingeniería de sistemas o industrias que requieran integración (*software* mas *hardware*). La intención de este nuevo modelo es consolidar y agrupar una familia de modelos que ya estaban siendo utilizados y reconciliar estos modelos con los estándares y metodologías (como ISO/IEC/TR 15504).

Como otros modelos de CMM, el modelo integrado de capacidad de madurez provee una guía acerca de cuando debe utilizarse el desarrollo del proceso. CMMI no es un proceso o la descripción de un proceso. El proceso actual, ya definido, es utilizado en una organización que depende de muchos factores, como, el dominio de aplicación, tamaño y estructura de la organización, etc.

La base del CMMI está constituida por los modelos SW-CMM (del que trata este trabajo), SA-CMM *(Software Acquisition Capability Maturity Model* ó madurez en la adquisición de *software*), IPD-CMM (*Integrated Product Development Capability Maturity Model* ó madurez integrada en el desarrollo del producto) y SE-CMM (*Systems Engineering Capability Maturity Model* ó madurez en los sistemas de ingeniería).

Este modelo se adapta para una unidad informática completa, no solo para el departamento de desarrollo.

5.1.1. Objetivos del CMMI

- Proporcionar una guía para desarrollar procesos
- Ayudar a evaluar la madurez de la organización o capacidad de un área de procesos de desarrollo

5.1.2. Categorías de áreas de procesos en CMMI:

1. Administración de procesos
2. Administración de proyectos
3. Ingeniería
4. Soporte

Existen dos versiones de este modelo, la escalonada (*staged*) y la continua (*continuos*). Aunque los expertos y algunos opositores a este modelo opinan que no existe una clara diferencia entre el modelo escalonado y el continuo, ya que el escalonado es más continuo de lo que aparenta.

La versión escalonada del modelo tiene 5 niveles, como el SW-CMM versión 1.1, que contienen 24 áreas clave de proceso (ACP), con 78 metas que se alcanzan al implantar las 618 prácticas comunes. Para la versión continua del modelo existen 6 niveles de madurez, que contienen 24 áreas clave de proceso (ACP), con 174 metas que se deben alcanzar y 455 prácticas comunes.

5.1.3. Niveles de CMMI modelo continuo:

0. Incompleto
1. Desempeñado
2. Administrado
3. Definido
4. Administrado cuantitativamente
5. Optimizado

Características:

- Permite seleccionar el orden de mejoras que se ajustan a los objetivos del negocio de la organización y ayudan a mitigar las áreas de riesgo de la organización.
- Habilita la comparación a través de y entre organizaciones en un área del proceso basado en las áreas clave del proceso o en sus niveles de madurez.
- Provee fácil migración de EIA/IS 731 a CMMI.
- Ofrece una fácil comparación de la mejora de procesos de la norma ISO 15504, porque las áreas clave del proceso de una organización se derivan de esta norma.

5.1.4. Niveles de CMMI modelo escalonado

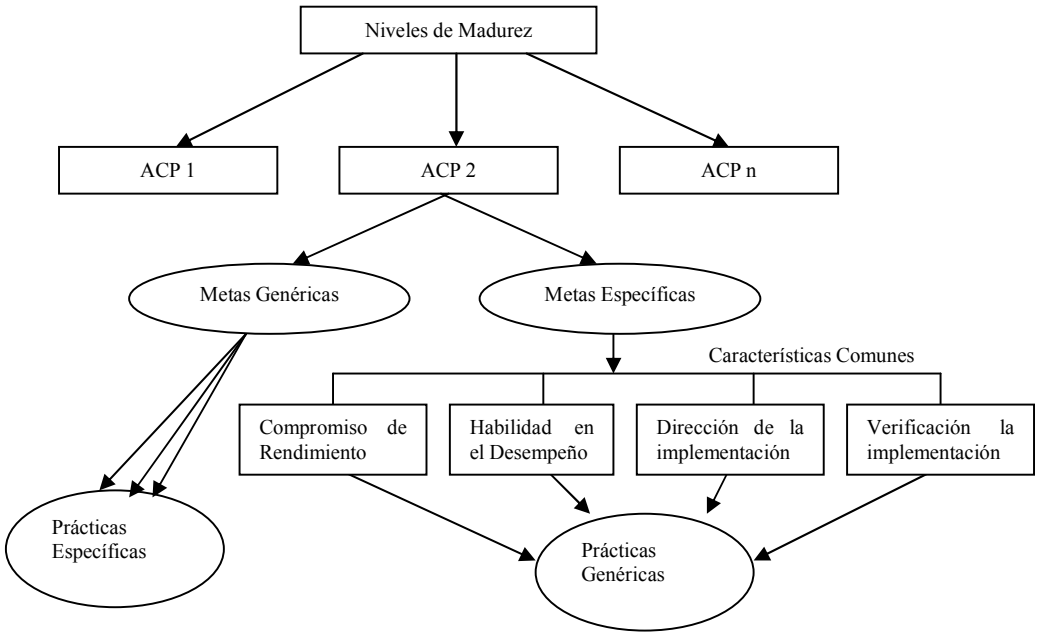
1. Inicial
2. Administrado
3. Definido
4. Administrado cuantitativamente
5. Optimizado

Características:

- Provee una acreditada secuencia de mejoras, iniciando con las prácticas básicas de administración y progresando con un predefinido y aprobado camino de niveles exitosos, donde cada uno sirve de base para llegar al siguiente.
- Provee fácil migración de SW-CMM a CMMI.
- Permite comparaciones con ISO/IEC 15504, pero, las áreas clave del proceso de la organización no corresponden a la organización que implementa ISO/IEC 15504.

A continuación se muestra en la figura 31 la estructura del CMMI (o CMM integrado) tomando en cuenta cualquiera de las dos versiones que se presentaron.

Figura 31. Estructura del CMMI



Fuente: CMM, www.sei.cmu.edu/cmm/cmm.html

El modelo CMMI esta diseñado para describir niveles discretos de mejora de procesos. Se parte de los niveles de madurez que organizan las áreas del proceso. Dentro de las áreas del proceso se encuentran las metas genéricas y las metas específicas y estas a su vez contienen prácticas genéricas y específicas. Las prácticas genéricas están organizadas por características comunes.

5.1.5. Componentes esperados, requeridos e informativos del CMMI

Todos los componentes del CMMI (o CMM integrado) están agrupados en tres categorías:

- **Componentes requeridos**: las metas genéricas y las metas específicas son componentes requeridos en el modelo, porque estas se logran por medio de la planeación de la organización y la implementación de los procesos. Los componentes requeridos son considerados esenciales para lograr la mejora de procesos proporcionados por las áreas del proceso. Éstos son utilizados en la estimación para determinar la satisfacción de las áreas clave del proceso y la madurez del proceso organizacional. Solamente las sentencias de las metas genéricas y las metas específicas son componentes requeridos del modelo.

- **Componentes esperados**: las prácticas específicas y las prácticas genéricas son componentes esperados del modelo. Los componentes esperados describen que prácticas de la organización van a ser típicamente implementadas para un conjunto de metas específicas y genéricas. Ellas son significativas para las guías individuales y la implementación de mejoras grupales y realizar evaluaciones. Cualquiera de las prácticas son descritas o

presentadas como una alternativa aceptable, dentro del proceso de implementación y planificación de la organización, si antes las metas se han considerado satisfechas. Únicamente las prácticas genéricas o específicas son componentes esperados en el modelo de componentes. Los títulos de una práctica genérica o específica y cualquier nota asociada a las prácticas son considerados componentes informativos del modelo. A través de las prácticas se alcanzan las metas, por eso ¿por qué no son requeridas? son esperadas porque las prácticas deben ser cumplidas a través del tiempo y lo que se espera es hacer que las prácticas se cumplan a cabalidad para cumplir con cada área clave del proceso.

- **Componentes informativos**: las subprácticas, los productos típicos de trabajo, las disciplinas, la elaboración de prácticas genéricas, los títulos de las metas y las prácticas, las notas de las metas y las prácticas, y las referencias son componentes informativos que pueden ayudar a los usuarios a entender las metas y las prácticas y como se pueden lograr. Los componentes informativos proveen detalles de ayuda al usuario para empezar a pensar en como llegar a las metas por medio de las prácticas.

5.1.6. Áreas clave de proceso del CMMI

Nivel 2

1. **Administración de requerimientos**: administrar los proyectos y los productos e identificar las inconsistencias entre los planes del proyecto, los productos de trabajo y los requerimientos.

2. **Planificación de proyectos**: establecer y mantener los planes definidos por las actividades del proyecto.

3. **Control y monitoreo de proyectos**: proveer entendimiento en cuanto al progreso del proyecto, si se han tomado las acciones correctivas apropiadas que puedan ser tomadas cuando el proyecto realiza una desviación significativa en el plan.

4. **Administración de subcontratistas**: administrar la adquisición de productos y servicios de los contratistas externos para proyectos donde exista un formal acuerdo.

5. **Mediciones y análisis**: desarrollar y sustentar métricas reales que sean utilizadas para soporte que la administración de la información necesita.

6. **Aseguramiento de la calidad del producto y del proceso**: proveer ayuda y administración para la comprensión de los objetivos de los procesos y los productos asociados.

7. **Administración de la configuración**: establecer y mantener la integridad de los productos por medio de la identificación de la configuración, el control de la configuración, llevar el conteo del estado de la configuración y realizar las auditorias de las configuraciones.

Nivel 3

1. **Desarrollo de requerimientos**: requerimientos del cliente, de producto y de los componentes del producto; el análisis de requerimientos para su desarrollo y comprensión.

2. **Soluciones técnicas**: para los requerimientos de los desarrolladores, los diseñadores y la implementación de soluciones. Solucionar, diseñar e implementar abarcando productos, componentes del producto y procesos relacionados con el producto, cada uno por separado o una combinación más apropiada.

3. **Integración del producto**: integrar el producto con los componentes del producto, asegurar la integración del producto, que el producto funcione apropiadamente y su entrega.

4. **Verificaciones**: asegurar que el producto seleccionado cumpla con los requerimientos específicos.

5. **Validaciones**: demostrar que el producto o los componentes del producto se cumplen a cabalidad por medio del entendimiento de su uso cuando se encuentran en un entorno proyectado.

6. **Enfoque en el proceso organizacional**: establece y mantiene el conocimiento de los procesos de la organización y el valor de los procesos, identificando planes y procesos de implementación de la organización en actividades de mejora.

7. **Definición del proceso organizacional**: establece y mantiene un conjunto aprovechable de procesos organizacionales.

8. **Entrenamiento organizacional**: desarrollo de las habilidades y los conocimientos del personal para que puedan realizar roles efectivos y eficientes.

9. **Administración de la integración del proyecto**: establecer y administrar el proyecto y el involucramiento relevante de los interesados de acuerdo a procesos integrados y definidos que se ajustan al conjunto de procesos estándar de la organización.

10. **Administración de riesgos**: identificar los problemas potenciales antes de que ocurran, administrar las actividades de riesgo que puedan ser planificadas y activar una orden necesaria a través de un ciclo de vida que mitigue las adversidades que impactan en el logro de los objetivos.

11. **Análisis y resolución de decisiones**: construir decisiones usando un método estructurado que evalúe las alternativas identificadas contra los criterios establecidos.

Nivel 4

1. **Rendimiento de los procesos organizacionales**: establecer y mantener un conocimiento cuantitativo del conjunto de procesos estándar de la organización y proveer la administración cuantitativa del rendimiento de los datos del proceso, los lineamientos y modelos de los proyectos de la organización.

2. **Administración cuantitativa de los proyectos**: administrar cuantitativamente los procesos definidos para el logro del establecimiento de la calidad y los objetivos de rendimiento del proceso.

Nivel 5

1. **Innovación y destacamento organizacional**: seleccionar y destacar las mejoras incrementales e innovadoras que mejoran mesuradamente los procesos y tecnologías de la organización. Soportar las mejoras de calidad de la organización y el rendimiento de los objetivos del proceso que están derivados de los objetivos del negocio de la organización.

2. **Resolución y análisis causal**: identificar las causas de los defectos y otros problemas y tomar acciones para prevenirlos cuando ocurran en el futuro.

Metas específicas

Las metas específicas son aplicadas solamente a un área clave del proceso y a las características que describen cómo debe ser implementada para satisfacer el propósito de esta área clave. Las metas son requeridas en el modelo de componentes y son usadas para asegurar si un área clave del proceso lo satisface.

Prácticas específicas

Son actividades que se consideran importantes para el logro de las metas específicas. Las prácticas específicas describen las actividades esperadas que resultan en el logro de las metas específicas de un área clave del proceso.

Metas genéricas

Las metas genéricas se aplican a todas las áreas clave del proceso. El logro de cada una de las metas en cada área clave del proceso significa que tanto la implementación y la institucionalización de cada área clave del proceso es efectiva, repetible y duradera.

Características comunes

Cuatro características comunes organizan las prácticas genéricas de cada área clave del proceso. Las características comunes nominadas en el modelo de componentes son informativas. Estas son agrupadas para proveer la manera de presentar las prácticas genéricas. Cada una es presentada en el modelo de forma abreviada.

(CO) Compromiso de rendimiento
(AB) Habilidad en el desempeño
(DI) Dirección de la implementación
(VE) Verificación de la implementación

Prácticas genéricas

Son aplicadas a todas las áreas clave del proceso porque en principio, estas pueden mejorar el rendimiento y control de cualquier área. Las prácticas genéricas proveen la institucionalización de características que aseguran que las áreas clave del proceso sean efectivas, repetibles y duraderas. Las prácticas genéricas son componentes esperados en el modelo.

El modelo del CMMI tiene el propósito de proporcionar una guía unificada para la mejora de múltiples disciplinas tales como ingeniería de sistemas, ingeniería de *software*, etc. Sin embargo, mucho se ha escrito y discutido sobre el tema de que las empresas que en realidad necesitan una guía de este tipo, son las grandes organizaciones (proveedores del Departamento de Defensa, principalmente) cuyos proyectos involucran múltiples disciplinas; mientras que la gran mayoría de los usuarios actuales del modelo SW-CMM son pequeñas organizaciones y/o áreas de desarrollo de *software*, que no utilizan o no necesitan otras disciplinas mas que la de ingeniería de *software* y ésta es el foco principal del SW-CMM.

La situación actual del CMMI en el mercado norteamericano, es incierta ya que a pesar de tener la presión del Departamento de Defensa por adaptar este nuevo modelo, muchas empresas han comentado que el CMMI no se adapta a sus necesidades, porque hay muchos elementos que resultan de más para empresas pequeñas-medianas cuyo negocio principal es el desarrollo de *software*, y no la integración de tecnología o *hardware*, que es el enfoque principal del nuevo modelo. Además del esfuerzo requerido para la implantación de las nuevas prácticas del CMMI, es necesario considerar el esfuerzo necesario para la capacitación y para la evaluación formal. El método de evaluación para el nuevo modelo se denomina SCAMPI (*Standard* CMMI *Assessment Method for Process Improvement*) y para realizar una evaluación se requieren considerar varios meses debido a la visión global que refleja el modelo.

La percepción en la industria internacional, es que este modelo se adapta más a empresas grandes, que requieren de diversas disciplinas. La transición del SW-CMM al CMMI requiere una inversión fuerte, incluso para las organizaciones que son maduras en el modelo SW-CMM (niveles 3, 4), ya que

se requiere realizar un esfuerzo extra para cubrir las nuevas áreas de proceso (en niveles inferiores y superiores), lograr un nuevo cambio de cultura y capacitar a la organización en el nuevo modelo de referencia. Si bien es cierto que la estructura del CMMI indica que se provee fácil migración de SW-CMM a CMMI, esta perspectiva de las empresas que han tenido experiencias en esta transición pueden hacer llegar a modificar la estructura del CMMI de modo que no sea una fácil migración entre modelos sino que SW-CMM puede servir de base para implantar CMMI con la debida consideración de fuertes inversiones y demás características que a las empresas hace un tanto difícil esa transición.

Sobre el tema de si CMMI se aplica a organizaciones de diferentes tipos y tamaños se presentó una conferencia titulada "CMMI *not for the little guy*?", impartida por Margaret Kulpa de Agile Dim Inc. El tema de la conferencia se enfocó al cuestionamiento de si el modelo CMMI se adapta a las organizaciones pequeñas, dónde el término “pequeñas” se utilizaba en el contexto de 20 o menos empleados, con 2 ó 3 empleados por proyecto, donde el personal asignado al proyecto desempeña varios roles y los proyectos tienen una duración de 3 a 6 semanas. A continuación se resumen algunos puntos de esta presentación con la finalidad de formar al lector con ideas más concretas de las implicaciones de este modelo en empresas pequeñas, como es el caso de muchas empresas latinoamericanas.

- El modelo del CMMI no provee guías de ajuste para adaptar el modelo a las organizaciones pequeñas. Esta guía es necesaria, debido a que la estructura del modelo (diseñada para muchos recursos asignados a proyectos, con muchos roles, proyectos muy largos que pueden durar años y cuestan millones de dólares).

- CMMI se basa en prácticas de organizaciones grandes, y está enfocado para organizaciones del departamento de defensa o aeronáutica.

- CMMI es demasiado grande para que pueda ser manejado en organizaciones pequeñas.

- El CMM ha sido criticado por tener muchas áreas clave, pero CMMI tiene muchas más. Esto implica que la documentación de procesos que cubra las prácticas del modelo puede ser agobiante para las organizaciones pequeñas, y por lo tanto, el tiempo, costo y recursos para la documentación pueden crecer exponencialmente.

- El retorno de inversión (ROI, por sus siglas en inglés) del CMMI no ha sido validado (especialmente en organizaciones pequeñas). El retorno de inversión, suele ser uno de las principales justificaciones para invertir en programas de mejora. Este punto es especialmente importante para las organizaciones pequeñas donde, la mayoría de las veces no se cuenta con un gran presupuesto e indudablemente el pago de la nómina cada dos semanas es una preocupación mayor. Actualmente no se tienen estudios que ayuden a calcular el ROI para el CMMI.

Las organizaciones pequeñas se administran de manera diferente a las grandes y enfrentan retos diferentes. El principal móvil de negocio para las empresas pequeñas es el tiempo de salida al mercado (*time-to-market*) de sus productos, por lo que la entrega rápida de productos es muy importante para éstas, y para el CMMI ese *time-to-market* no parece ser una fuerza impulsora.

CMMI fue escrito para organizaciones maduras. El material introductorio de las primeras versiones del modelo escalonado (CMMI *staged*), decía que las

organizaciones evaluadas en niveles superiores del SW-CMM deberían utilizar el CMMI. La mayoría de este tipo de organizaciones son grandes, y por lo general ya han trabajado en programas de mejora o han alcanzado un buen entendimiento de lo que significa la mejora de procesos. El requerimiento de CMMI para el programa de métricas es complicado y sofisticado desde el nivel 2, y aunque la definición de métricas es importante para cualquier programa de mejora esto puede ser difícil de implantar en una organización principiante.

Se podría seguir listando una serie de elementos que han sido criticados en el nuevo modelo de CMMI y las inquietudes que existen, incluso en la industria estadounidense y en el propio SEI, en cuanto a la aceptación por el modelo. Con esto se llega a reflexionar sobre la dificultad que representa para las empresas latinoamericanas la adopción de este nuevo modelo. Sobre todo para aquellas que no tienen experiencia anterior con un programa de mejora basado en el SW-CMM.

Actualmente, no hay muchas organizaciones que hayan adoptado o estén haciendo la transición hacia el nuevo modelo. Incluso los grandes corporativos norteamericanos tienen que realizar una fuerte inversión para hacer la transición.

Lo que la comunidad internacional está pidiendo es que se mantengan los dos modelos, se libere la versión 2 del SW-CMM que actualmente está en versión borrador y permitir que el mercado decida cual de los dos modelos debe utilizar con base en sus necesidades y objetivos de negocio (SW-CMM versus CMMI).

Finalmente, es necesario comentar que el éxito del proyecto no está en la selección de un modelo en particular, sino en establecer un programa de

mejora que establezca nuevas prácticas y disciplinas de trabajo para el desarrollo de *software* utilizando un modelo como un marco de referencia que ayude a las organizaciones a lograr sus objetivos de negocio. Lo recomendable es que éste sea reconocido mundialmente con el objetivo de ser comparable con otros proveedores en mercados internacionales.

Para una organización que involucra departamentos de desarrollo, sistemas, soporte, etc. El modelo CMMI en su nivel 2 puede involucrarlas de la siguiente manera:

Sistemas:

- **Administración de proyectos**: como los servicios de recuperaciones y respaldos a la base de datos y sistema, administración y mantenimiento de la base de datos, mantenimiento a usuarios del sistema tanto internos como externos.

- **Planificación de requerimientos**: los de *hardware*, por ejemplo.

- **Control y monitoreo de proyectos**: de los servicios mencionados anteriormente, asignando tiempos y llevándolos a cabo.

- **Administración de subcontratistas**: personas o empresas que ofrecen productos de *hardware* o *software*, que dan asesorías y capacitaciones al departamento.

- **Mediciones y análisis**: del impacto en la base de datos cuando ocurre un problema, de los cambios al *hardware* y *software*, de los respaldos al sistema y base de datos, de las fallas.

- **Aseguramiento de la calidad del producto**: el servicio que prestan debe llevar calidad.

- **Administración de la configuración**: de la base de datos, del sistema, del *hardware* y *software* asociados a los servicios que presta el departamento, de los cambios a la base de datos de producción.

Soporte:

- **Administración de proyectos**: los técnicos en equipo tienen proyectos para llevar a cabo como lo son el soporte, mantenimiento correctivo, adaptativo y preventivo de *hardware* y *software*, etc.

- **Planificación de requerimientos**: los de *hardware* y *software*.

- **Control y monitoreo de proyectos**: control y seguimiento de tareas en base a los servicios que prestan.

- **Administración de subcontratistas**: a las personas encargadas de distribuir productos, suministros y que hasta cierto punto puedan dar capacitaciones sobre los mismos.

- **Mediciones y análisis**: en el estado de las tareas, en el tiempo involucrado para cada una.

- **Aseguramiento de la calidad del producto**: en los servicios que prestan.

- **Administración de la configuración**: en los servicios que presta el departamento debe llevarse un estricto control del *hardware* y *software*, control de las versiones que los mismos presentan.

5.2. ISO para *software*

ISO es la Organización Internacional de Estándares, no es una organización gubernamental. Es el Cuerpo Mundial de la Federación Nacional de Estándares, fue establecida en 1947; actualmente se encuentra en más de 100 países para promover el desarrollo de la estandarización mundial.

Series ISO 9000 para productos

Es una serie de estándares aprobados y respaldados por el estándar ISO y fue adoptado en 1987 como un estándar mundial. A éste le concierne la administración de calidad y este se convierte en un requisito para la competencia global. Para asegurar la satisfacción del cliente una organización se compromete a aplicar requerimientos que regulen el producto y a realizar mejoras continuas en su producto.

Es un conjunto de requerimientos con los que debe cumplir una organización de acuerdo con los procesos del negocio que van desde diseño y desarrollo, a la producción, instalación y servicio.

Una organización escoge un sistema para certificarlo en ISO 9001 en cuanto a su calidad se refiere, el sistema debe cubrir los procesos del negocio y la calidad necesaria para poder certificarse.

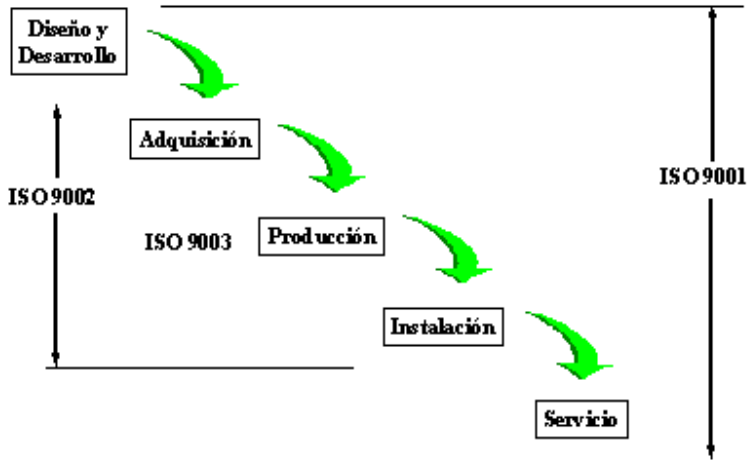
ISO 9000: definiciones, conceptos y términos relacionados con la calidad.
ISO 9001: estándar general que comprende el aseguramiento de la calidad en el diseño, desarrollo, manufactura, instalación y servicio de los productos.
ISO 9002: estándar que detalla el enfoque especialmente en manufactura e instalación de productos.
ISO 9003: estándar que detalla la cobertura de inspección final y las pruebas necesarias para completar el producto.
ISO 9004: provee los lineamientos para administrar un sistema de control de calidad para utilizar sistemas de auditoria de calidad.

Aunque no se especifique para que tipo de productos está orientada la norma ISO, puede ser aplicado a todo tipo de productos; el desarrollo y manutención de *software* es uno de ellos puesto que involucra diseño y desarrollo, manutención y puesta en marcha, incluyendo la administración de calidad total. El *software* es un producto final y como tal las empresas pueden certificarse en el estándar ISO y ofrecer con su producto calidad mundial.

Como se aplica ISO al *software*

La figura 32 muestra las normas ISO de administración de productos, se muestra que si se desea aplicar una de las normas ISO para desarrollar *software* se podría utilizar ISO 9001. La norma ISO 9001 está enfocado a todos los factores menos la tecnología; éste es importante porque ISO no es un modelo evolutivo y la tecnología es completamente evolutiva, esto produce un estancamiento del producto que hasta podría volverlo obsoleto por no estar a la altura de la tecnología. ISO provee requerimientos (que necesitan ser completados) y no especifica el uso de soluciones prescriptivas, o sea, como hacerlo.

Figura 32. Estándares ISO para productos



5.2.2. Criterios para la certificación ISO9000/9001

La norma de estandarización ISO es comúnmente llamada “Lista de Verificación” puesto que hasta que no se cumple el último requerimiento en el producto, y no necesariamente software, la empresa no obtiene su certificación. A continuación se enumeran los criterios básicos a seguir para obtener una certificación:

- Revisar las operaciones actuales y la facilidad de su estructura de negocios.
- Proporcionar una sesión introductoria en los requerimientos ISO 9000/9001 e instruyan en la preparación de las descripciones del trabajo y trabaje en base a estas instrucciones y documéntelas.
- Intervenir en las descripciones del trabajo completadas y en las instrucciones de trabajo, preparar la estructura para las políticas y procedimientos manuales.

- Escribir el primer borrador con las políticas y procedimientos manuales que sean apropiados a los estándares ISO 9000/9001, incorporando los documentos con las instrucciones de trabajo existentes.
- Proponer el primer borrador de manuales para revisión y aprobación. Prepare el borrador final de la documentación y manuales de auditoria de conformidad con los requerimientos ISO 9000/9001.
- Entrenar al personal en las políticas, procedimientos y manuales de instrucción de trabajo y reciba retroalimentación acerca de la veracidad de la documentación.
- Aplicar una tercera auditoría que simule la implementación del sistema de calidad utilizando auditores de sistemas calificados.
- Ajustar el sistema de calidad para prepararlo para la certificación de la tercera auditoría.

Puntos fuertes de ISO 9001

- Es un factor competitivo para las empresas.
- Ahorro de tiempo y dinero al evitar tener que demostrar la calidad una y otra vez.
- Adoptado en más de 90 países e implantado en todo tipo de organizaciones, industriales y de servicios, del sector privado y del público.
- Garantía de que las actividades se hacen bien.

Puntos débiles de ISO 9001

- Es estático, de escaso valor y caro (según la empresa Motorola)
- Es cuestión de tiempo que deje de ser un factor competitivo
- Adoptado en muchos casos por obligación y para cubrir el expediente
- Diferencias en cuanto a la interpretación de las cláusulas del estándar

5.3. CMM e ISO 9001

La mayoría de las empresas en Latinoamérica que cuentan con un nivel CMM están certificadas en ISO 9001. En Latinoamérica una de las normas más conocidas es ISO 9001, ya que aplica no sólo para empresas dedicadas al desarrollo de *software*. CMM todavía no cuenta con un reconocimiento en Guatemala como lo tiene en otros países tales como: Estados Unidos, México o la India. Requerirá de tiempo para que un cliente externo al desarrollo de *software* conozca y pida a su proveedor de *software* contar con un nivel de madurez.

En este momento la mayoría de las empresas latinoamericanas con un nivel de madurez inicial, trabajaron con CMM para satisfacer los requerimientos de sus clientes internacionales o para satisfacer la competencia internacional. Aunque no debemos olvidar que CMM no sólo ayuda a dar seguridad a los clientes sobre la capacidad de una organización para cumplir con sus metas sino también a ayuda a la organización de desarrollo o manutención de *software* a crecer internamente.

5.3.1. Comparación ISO 9001 versus CMM

La norma ISO 9000 a través del referencial ISO 9000 - 3 Gestión de la calidad y aseguramiento de la calidad, directrices para la aplicación de la norma ISO 9001 al desarrollo, suministro, instalación y mantenimiento del soporte lógico ofrece elementos de interpretación para adecuar un sistema que "produce" soporte lógico, aunque la certificación sólo se puede efectuar por una de las tres normas ya conocidas (normalmente se diseña soporte lógico - *software* y por lo tanto se aplica la ISO 9001).

5.3.2 Cuadro comparativo entre ISO 9000 y CMM

ISO es una norma o estándar en el cual las empresas pueden certificarse, mientras que CMM es un modelo con el cual, también, las empresas pueden obtener una certificación. La adopción de la norma o del modelo va a depender de las necesidades de cada organización, en la tabla VI se muestra un cuadro comparativo entre el modelo y el estándar, lo cual puede constituir una base para la adopción de cualquiera de ellos.

Tabla VI. Cuadro comparativo entre la norma ISO y el modelo CMM

	ISO 9000/9001	**CMM**
Definición	Es un conjunto de documentos que tratan con sistemas de calidad que pueden utilizarse para propósitos se aseguramiento de aseguramiento de calidad externa.	Describe los principios y prácticas que son la base de un proceso de madurez del *software* y propone ayudar a las organizaciones de *software* a mejorar la madurez de sus procesos de *software*.
Desarrollo	Desarrollado por ISO.	Desarrollado por SEI.
Escrito para	Amplia gama de la industria.	Industrias de *Software*.
Documentos	Abstractos.	Detallados.
Longitud de páginas	Solo 5 páginas.	Más de 500 páginas.
Conceptos	Seguir un conjunto de normas para repetir los éxitos.	Para dar énfasis en el logro de un proceso de mejora continua.
Documentación	En la relación cliente/proveedor para reducir el riesgo de que un cliente escoja un proveedor.	El proveedor para mejorar el proceso interno de *software*.
Administración de la responsabilidad	Las políticas de calidad están definidas, documentadas,	Las políticas de calidad

	entendidas, implementadas y mantenidas; las responsabilidades y autoridades del personal son específicas, los logros y monitoreos de calidad están definidos.	se dirigen principalmente al aseguramiento de la calidad del *software*.
Calidad del sistema	Requiere un sistema de calidad documentado incluyendo instrucciones y procedimientos establecidos.	Los procedimientos son dirigidos y aseguran la calidad del *software*. Las tareas de la ingeniería de *software* están definidas, integradas y se realizan de forma consistente
Revisión de contratos	Se revisan contratos para determinar si los requerimientos se definen adecuadamente, están de acuerdo con la oferta y pueden ser implementados.	Se documenta y revisa que los requerimientos faltantes sean claros.
Control de diseño	Requiere de procedimientos para controlar y verificar que el diseño sea establecido. Incluye actividades de planificación de diseño, identificación de entradas y salidas, verificación del diseño y control de cambios al diseño.	El ciclo de vida en las actividades de análisis de requerimientos, diseño, codificación y pruebas son descritos en la ingeniería de *software* del producto.
Control de la documentación	Requiere que la distribución y modificación de documentos sea controlada.	Descrito en la administración de la configuración de *software*.
Compra	Requiere la compra de productos conforme a los requerimientos específicos, esto incluye la valoración de subcontratistas	Se especifica en la administración de subcontratistas de *software*.

	potenciales y la comprobación de los productos comprados.	
Producto proporcionado	Requiere que el producto proporcionado sea verificado y mantenido.	Describe el uso del *software* adquirido, que hace que se pueda identificar la reusabilidad como parte de la planificación.
Identificación del producto y seguimiento	Requiere que el producto sea identificado y se le de seguimiento durante todos los estados de producción, entrega e instalación.	Los estados del producto de ingeniería de *software* especifican las necesidades de consistencia y seguimiento entre productos de *software*.
Control de procesos	Requiere que la producción de procesos este definida y planificada, esto incluye llevarlo a producción y mantenerlo bajo control según las instrucciones documentadas.	Los procedimientos que definen el proceso de producción de *software* están distribuidos a lo largo de las áreas clave del proceso en las variadas actividades realizadas en la práctica.
Servicio	Requiere que las actividades de servicio se realicen de forma más específica.	Pretende aplicarlo en los ambientes de desarrollo y mantenimiento de *software*.
Entrenamiento	Requiere que se identifiquen las necesidades de entrenamiento y este debe proporcionarse por personal especializado. El entrenamiento debe documentarse.	Se identifican las prácticas de entrenamiento y orientación en la característica común de Habilidad del desempeño.
Calidad de registros	Requiere que la calidad de los registros sea colectada, mantenida y a disposición.	Especialmente esta cláusula pertenece a las pruebas y prácticas de revisión de pares que se encuentran en el producto de Ingeniería de *Software*.

5.3.3. Puntos de vista del modelo SW-CMM e ISO 9001

5.3.3.1. Pretensiones, objetivos y alcance

La norma ISO 9001 se centra en la relación entre el cliente y el suministrador, con objeto de reducir el riesgo en la contratación de proveedores de *software*.

Por su parte CMM, aunque presenta el enfoque necesario para evaluar la capacidad del proceso *software* de otras empresas, se centra principalmente en la determinación de la capacidad del propio proceso de desarrollo de *software* de la empresa, así como en evaluar la capacidad de este proceso.

ISO 9001, cubre diversos aspectos como *hardware*, *software*, materiales, servicios, etc. indica varios puntos que no pertenecen al proceso de producción de *software* propiamente dicho, como por ejemplo el control de la documentación o la atención al cliente en el servicio posventa. Punto éste mal considerado por CMM. CMM tan solo implica al cliente de forma clara en la fase de especificación de requisitos, pero no proporciona aclaraciones sobre el control de los productos no conformes, el servicio posventa, o el almacenamiento y distribución del *software*. En este sentido, la norma ISO 9001, contempla un escenario más amplio de los elementos que intervienen en una organización productora de *software* y presenta una gran atención hacia el cliente, mientras que CMM se restringe al proceso de producción de forma más específica y también, por tanto más eficaz.

Por otro lado, y para nivelar la diferencia anterior, la norma ISO 9001, es poco concreta, o mejor dicho muy general, ya que su concepción fue pensada para abarcar cualquier tipo de empresa de cualquier sector, y aunque cuenta

con el apoyo y guía proporcionado por la norma ISO 9000-3, no es, y con mucha diferencia, tan minuciosa y elaborada como CMM; con ISO 9001 se pierden los detalles y no se enfoca al proceso de desarrollo de *software* sino a cualquier proceso en donde se necesite calidad, por lo tanto el *software* no toma un camino seguro y conocido sino uno lleno de inconvenientes. El modelo CMM, nació enfocado hacia el desarrollo concreto de productos *software* y proporciona mecanismos y ejemplos, a muy bajo nivel, tocando los más recónditos rincones del proceso de desarrollo y producción de *software*, lo que permite conocer y depurar éste de forma muy minuciosa obteniendo éxitos que en ISO 9000 no se pueden tener.

La norma internacional ISO 9001, tan solo indica las áreas a considerar y aspectos de estas que es necesario cubrir; CMM por el contrario, sin enfatizar en el hecho estructural de la organización, precisa como deben ocurrir las cosas para lograr mejorar y alcanzar un proceso de desarrollo maduro.

CMM y su espíritu incremental mueven el sistema de calidad en una dirección de mejora continua, motiva a la organización en el progreso y en la mejora. Mientras que ISO 9001, pese a que también proporciona mecanismos de mejora, lo hace más débilmente (y con diferencia); tanto es así que la realidad empresarial demuestra que el único reto es mantener la certificación.

5.3.3.2. Implantación en la organización.

Una empresa que ha adoptado la norma ISO9001, tan solo constata que ciertos hábitos básicos de calidad están establecidos en la organización, lo que asegura que esa empresa es capaz de satisfacer aquello que oferta, pues su sistema y procesos están tan bien encauzados, que sus desarrollos se repiten

exitosamente. Una empresa que ha adoptado el modelo CMM como sistema de calidad, indica que no solo implanta los cimientos básicos del aseguramiento de la calidad, si no que está continuamente construyendo sobre estos cimientos.

Si una empresa con certificado de calidad ISO 9001, abriera una nueva línea de producción, deberá, para certificar la calidad de este nuevo proceso, repetir los mismos pasos que realizó para implantar el sistema de calidad ISO 9001, en los procesos anteriores. Es más, una organización, con diversidad de producción, puede certificar algunos procesos y productos a través de la ISO 9001, sin certificar los restantes. Una empresa con un sistema de calidad basado en el modelo CMM, es mucho mas flexible, a la hora de integrar un nuevo proceso; eso si, siempre y cuando ese nuevo proceso, sea de desarrollo de *software*, si fuera por ejemplo, ensamblaje de elementos *hardware*, de nada serviría CMM, mientras que ISO 9001 si sería útil.

CMM permite a la empresa cierta independencia, a la hora de ser implantado; es cómodo y flexible, y está pensado para su uso dentro de la organización, es muy cómoda para la autoevaluación. La norma ISO 9001, por el contrario, precisa de consultores externos, conocedores de la norma, la cual es muy árida para la autogestión por neófitos; no es flexible (aunque si adaptable), es inamovible respecto a su definición. Para estar seguros de haber implantado correctamente la norma, necesitamos que una institución certificada a tal efecto, constate la adecuación del sistema, y nos registre como empresa certificada ISO 9001.

5.3.3.3. Estructura y aplicación

La norma ISO 9001 tiene carácter estático, mientras que el modelo CMM presenta una personalidad más dinámica. Este hecho genera dos factores muy importantes que diferencian ambas propuestas. La naturaleza estática de ISO 9001, puede entenderse como si se hiciera en un momento dado una fotografía del sistema de madurez de la empresa que se obtendría gracias a la suculenta documentación y registros generados por la norma. El pequeño matiz de dinamismo se logra, comparando resultados sucesivos con la intención de repetir procesos exitosos. Pero esta característica ya es contemplada por el modelo CMM en sus fases iniciales, y tras ello va mucho más allá. CMM debe su gran dinamismo, a su arquitectura de niveles y al flujo piramidal de mejora continua que inunda continuamente este modelo.

Pero este hecho dinámico, por el contrario, hace que la implantación de CMM, necesite de varios años; se ha constatado, que puede llevar del orden de dos a tres años el salto del nivel 1, o nivel inicial al nivel 2 o repetible, y de uno a dos años pasar a niveles sucesivos. Mientras que la naturaleza más estática de la norma ISO 9001, que permite ser plasmada casi de un vistazo (esto ha de entenderse metafóricamente, pero da una idea clara del concepto que se pretende exponer), hace que ésta se pueda implantar en unos meses; estadísticamente una empresa media tarda menos de un año en certificarse, sin dedicar una persona a tiempo completo para tal menester, lográndose, incluso, la certificación en menos de seis meses disponiendo una persona por completo dedicada a ello. Pudiera este último hecho resumirse diciendo que CMM proporciona un marco jerárquico que permite un progreso continuo e incremental. Mientras que ISO 9001 presenta una estructura plana donde todo está al mismo nivel.

Al igual que en la implantación, en la aplicación, CMM permite llevar a cabo autoevaluaciones, gestionadas por la propia empresa. Mientras que la norma ISO 9001, necesita de auditorías externas periódicas, para comprobar la continuidad y adecuación de la norma, mediante las cuales la empresa conserva la certificación.

Ambos modelos presentan un buen método de evaluación y determinación de la capacidad de los sistemas y procesos a terceros. La única diferencia es que CMM se basa en la evaluación de la capacidad y madurez del proceso, mientras que la ISO 9001, revisa la organización en base a la comparación con unos estándares predefinidos.

5.3.3.4. Posición en el mercado

El modelo CMM está muy extendido en los EE.UU. y Canadá, mientras que la certificación a través de la ISO 9001, es más frecuente en Europa y Asia; Europa es la región comercial con más empresas certificadas por las normas ISO 9001.

Ciertos estudios revelan que la mayoría de las empresas de desarrollo de *software* se encuentran en el nivel 1 o inicial; y prácticamente ninguna alcanza todas las metas propuestas en los niveles cuarto y quinto del modelo CMM. Esta información no se puede extraer del sector de empresas certificadas por la ISO 9001, pues éstas simplemente están certificadas o no lo están. Aunque debe decirse que toda organización que se propone certificarse finalmente lo logra, y son pocas las pierden la certificación rápidamente.

5.3.3.5. Beneficios obtenidos

Aunque la adopción de uno u otro estándar, afecta y moldea la cultura global o idiosincrasia de las organizaciones, imprimiendo a éstas el carácter, ya descrito de cada modelo, lo cierto es que ambos reportan grandes beneficios a las empresas. Incrementan la productividad, mejoran las comunicaciones horizontales y verticales y mejoran la satisfacción del cliente, se reducen costos y se genera entusiasmo.

Pese a que se han presentado diferencias, entre un modelo y otro, no olvidemos que ambos persiguen y consiguen implantar un sistema de calidad en las organizaciones de desarrollo de *software*.

La idea es aumentar la calidad del *software* aplicando controles de calidad al interior del proceso del desarrollo y no controlar la calidad del producto en sí. La diferencia está en que la segunda premisa centra el control de calidad cuando el producto ya se generó y la primera premisa distribuye este esfuerzo en las diferentes etapas y actividades que tiene un proceso de desarrollo. Esta diferencia conceptual motiva tomar el modelo y no la norma ISO 9000-3, dado que para asegurar un producto de calidad se debe especificar un proceso de calidad. Las normas ISO 9000-3 e IEEE se utilizan para especificar los estándares de documentación y algunos criterios de validación y verificación tanto de la documentación como del *software*.

ISO 9000 aplica control de calidad sobre productos terminados y esto implica que si el *software* tiene fallas se debe rehacer o se deben depurar los errores y esto lleva más tiempo que la primera tarea, en cambio el aplicar control de calidad en el proceso verifica la correcta aplicación de estándares y la ejecución de buenas prácticas de desarrollo minimizando el riesgo de fallas,

inconsistencias u omisiones en las funcionalidades. Sin embargo, toma años el aplicar CMM y por ende las mejoras del proceso se ven en forma paulatina, en cambio el aplicar ISO 9000 puede sólo tomar meses el implantarlo y algún tiempo el visualizar sus resultados.

La necesidad de este análisis es identificar la importancia estratégica para la organización que puede tener el generar *software* de calidad.

- El nivel 2 de CMM está estrechamente relacionado con ISO 9001.
- Toda área clave de CMM está al menos relacionada con ISO 9001.
- Dada una implementación razonable del proceso de desarrollo de *software*, una organización que obtiene, y retiene, la certificación ISO 9001 debe estar cerca del nivel 2 de CMM.
- Aún una organización en el nivel 2 de CMM debería tener poca dificultad en obtener la certificación ISO 9001.

La conclusión: para fines prácticos es conveniente basar la ruta de capacitación en CMM por su énfasis en la mejora o madurez del proceso, porque es más claro y específico para *software* y porque permite lograr también la certificación ISO.

5.4 Aplicación del modelo en empresas guatemaltecas que desarrollan y administran *software*

La información en Guatemala con respecto a CMM es muy escasa. En realidad, no existe un organismo nacional que tenga un control sobre aquellas organizaciones interesadas en este modelo o que por lo menos lo dé a conocer.

Se han llevado a cabo distintos talleres y conferencias sobre metodologías y normas de calidad aplicadas a *software*, no precisamente sólo sobre CMM. A estas reuniones han asistido distintas organizaciones dedicadas al desarrollo y manutención de *software* y académicos.

Al tratar de recabar información específica sobre CMM en Guatemala nos encontramos con varios obstáculos, el mayor de todos la falta de conocimiento acerca del modelo. El segundo obstáculo fue que en los registros del SEI no aparece ninguna empresa guatemalteca con un nivel de madurez mayor a 1, es más, ninguna empresa esta tratando de certificarse o por lo menos no se ha dado a conocer en el SEI.

Un punto a favor es que hay una empresa que está preparándose para certificarse en el nivel 2 del CMM y es la Superintendencia de Administración Tributaria (SAT) y esta es el caso de estudio del presente tema.

A continuación se muestra en la tabla VII un plan de implementación para SAT que involucra el tiempo estimado para la certificación de cada nivel del modelo CMM, también se elaboran ventajas y desventajas de su implementación para la transición entre niveles.

Tabla VII. Plan de implementación de CMM en SAT

Nivel	Tiempo	Ventajas	Desventajas
Inicial a repetible	3 años	▪ Se toman en cuenta detalles ▪ Mejor coordinación ▪ Mejoras en la comunicación ▪ Permite hacer correcciones ▪ Se realizan mas	• Se pierde el control sobre lo que se hace o planifica • Cambios en la planificación inicial • Tendencia a certificarse en otro modelo y abandonar el presente

		revisiones	
Repetible a definido	2 años y medio	▪ Personal experto ▪ Mejoras sustanciales en los procesos ▪ Aumento de la productividad ▪ Organización se enfoca en los procesos mas productivos ▪ *Software* cada vez mas integrado	• Dependencia del personal que capacita • Aumento de la complejidad de los sistemas • Pocos o escasos planes de contingencias
Definido a administrado	6 meses	▪ Procesos automatizados de control de calidad ▪ *Software* con mayor y mejor calidad ▪ Se estima de mejor manera la inversión en los proyectos de *software*	• La nueva tecnología hace los sistemas obsoletos • Los cambios no son administrados
Administrado a optimizado	1 año	▪ Se elaboran planes de contingencias ▪ Control en el crecimiento acelerado del *software* ▪ Repetición de éxitos de proyectos anteriores ▪ Se previenen defectos en el *software* a todo nivel	• Tendencia a invertir poco por el tiempo de implementación • Rotación del personal daña planes y administración de procesos • Retorno a niveles anteriores por no poder mantenerse en este nivel

La SAT para poder certificarse en el nivel dos de CMM se enfoca en los siguientes aspectos:

- Implantar una cultura de procesos, que para todo exista un procedimiento y también para que los éxitos obtenidos en la realización de un proceso (o un proyecto) puedan repetirse y además sea implementado en las distintas áreas y departamentos de la gerencia.

- Compromiso de la gerencia de informática y la institución.
- Adecuar y mejorar los procesos de desarrollo de *software* de manera evolutiva.
- Establecer bases de datos del capital intelectual de la empresa con las políticas, procesos y procedimientos de calidad, planes, estimados, métricas y mejores prácticas de los proyectos para el uso de todo el personal.
- Realizar un esfuerzo de aprendizaje y disciplina, contratando expertos y capacitando ampliamente a su personal en las técnicas y materias requeridas por el nivel 2 de CMM
- Involucrar al resto de áreas de la institución en el proceso de mejora.

Modelo del proceso

- El modelo de ciclo de vida que se ocupa en el desarrollo de *software* es el Iterativo incremental.
- Los hitos más importantes son el análisis de requerimientos, el diseño de requerimientos, la terminación del *software*, la elaboración del manual del usuario, la entrega del *software*, las pruebas del producto, el mantenimiento luego de ponerlo en marcha, la atención al usuario final en turnos de 24 horas.

5.4.1 Proceso administrativo

5.4.1.1 Objetivos y prioridades

La meta primordial es certificarse en el nivel 2 organizacionalmente. Para cumplir con esta meta se llevará a cabo lo siguiente:

- Comunicar y capacitar a todo el personal de la gerencia en la metodología, procedimientos y uso de herramientas
- Elaborar un reporte semanal, del avance y estatus de los proyectos de desarrollo.
- Contratar personal de consultoría para que de soporte en la administración de proyectos en herramientas Rational.
- Crear métricas de requerimientos y calidad.

5.4.1.2. Mecanismos de monitoreo y control

Para dar un seguimiento a los proyectos se llevan a cabo las siguientes tareas:

- Se elaboran reportes sobre el estado de los proyectos.
- Se hacen revisiones a los proyectos de *software*.
- Se evalúa a los analistas de sistemas, para constatar que están cumpliendo con las normas y políticas.
- Se realizan métricas por estados de desarrollo, por persona asignada y por estabilidad del proyecto.

5.4.1.3. Plan de equipo de trabajo

Para llevar a cabo la meta se necesita de una persona que trabaje en la búsqueda de las metas propuestas por unos meses, un subcontratista que conozca el modelo CMM y además maneje las herramientas de *software* y *hardware*.

5.4.1.4. Métodos, herramientas y técnicas

Requerimientos de *hardware*

- 1 computadora

Requerimientos de *software*

- Sistema operativo Windows en versiones nuevas.
- Herramientas
  - Herramientas de desarrollo Oracle, Portal Oracle, Java, Visual Basic, etc.
  - Herramientas Rational para el soporte y administración de productos.
    - Rational Rose para el modelado de casos de uso.
    - Rational Requisite Pro para el control de requerimientos de *software*, productos y casos de uso.
    - Rational Test Manager para llevar a cabo el control de calidad de los proyectos de desarrollo.
    - Rational Clear Quest como sistema de control de seguimiento.
    - Rational Clear Case para el control de versiones.

Se ha visto que la organización está involucrada no solo en las áreas clave que contiene el nivel 2 sino también áreas clave de otros niveles.

Se manejan plantillas y estándares, para ser aplicados y tomarlos como base; ver tabla VIII.

Tabla VIII. Plantillas y estándares

Nombre del archivo	Tipo
Administración de proyectos en RequisitePro.	Administración de requerimientos.
Plantilla otros requerimientos Complementarios.	Administración de requerimientos.
Plantilla casos de uso.	Administración de requerimientos.
Plantilla visión.	Administración de requerimientos.
CMM-AR-P-GI-005 - Atención a cambios y desarrollo nuevas aplicaciones.	Administración de requerimientos.
Organización de proyectos Test Manager.	Control de calidad.
CMM-AC-Pruebas sistemas v 1.2	Control de calidad.
Proceso de desarrollo SAT.	Evaluación del proceso.
Preguntas áreas clave CMM nivel 2.	Evaluación del proceso.
Plantilla términos de referencia v 1.0	Manejo de subcontratistas.
Política para la administración de subcontratistas.	Manejo de subcontratistas.
CMM-PP-Planificación de proyectos.	Planificación de proyectos.
CMM-PP-Plantilla cronograma.	Planificación de proyectos.
CMM-SP-Seguimiento de proyectos.	
Procedimiento Lista de verificación de pruebas para aplicaciones *web.*	Evaluación de sistemas de *software* para *web*.

Como se administran otras áreas clave del proceso del nivel 2:

- **Inspección y pruebas de proyectos**: listas de chequeo, validación de sistemas elaborados para *Web*, planificación de pruebas, diseño de pruebas, ejecución de pruebas, reportes de resultados, afinación de código.

- **Administración de configuración del *software***: control de versiones a la codificación sin importar la fase en el que este se encuentre ni la herramienta sobre la cual se elabora.

- **Planificación y seguimiento de proyectos**: creación de proyectos, asignación de actividades, asignación de tareas, asignación de servicios a las tareas, tareas con documentos adjuntos; control de tareas programadas y no programadas; asignación de pesos a proyectos; una tarea lleva seguimiento desde que inicia hasta que finaliza por medio de reportes de avances en las mismas; reasignación de tareas; realizar consultas personales; elaboración de reportes como:

  - Servicios por proyecto por usuario
  - Reporte de avances (horas trabajadas en un periodo de tiempo)
  - Tareas asignadas al usuario que inició la sesión
  - Tareas asignadas pendientes
  - Tareas asignadas terminadas
  - Tareas por responsable
  - Tareas por asignadas por memorándum
  - Tareas por número
  - Tareas en general
  - Proyectos por área

  - **Aseguramiento de la calidad del *software***: involucra las actividades realizadas en la inspección y pruebas de proyectos de *software*, se realizan revisiones semanales de proyectos en la herramienta Requisite Pro y se reportan avances en proyectos por analista y por proyecto, se recibe y proporciona capacitación sobre el correcto uso de herramientas, se llevan estadísticas de prueba para cada sistema.

CONCLUSIONES

1. Los modelos y estándares que se centran en la calidad del proceso y del producto de *software* son las series ISO 9000, SW-CMM, SE-CMM e I-EEE; unos se centran más en la calidad del producto terminado y otros, en los procesos de desarrollo de *software*.

2. Los factores que inciden sobre la calidad de los procesos de desarrollo de *software* son el tamaño (de la organización, del recurso humano), la educación, la tecnología, el tiempo, el esfuerzo, el recurso económico y otros que tienen menos importancia como la fiabilidad, la eficiencia, facilidad de mantenimiento y la usabilidad.

3. El proceso de desarrollo de *software* involucra un plan en tiempo y costos, análisis, diseño, construcción, entrega y evaluación del cliente. El punto de vista del SW-CMM es involucrar estas etapas en una metodología de desarrollo y a la vez ir integrando cada una de las áreas clave del proceso de los niveles de madurez.

4. La estructura del SW-CMM contiene cinco niveles de madurez, en cada nivel se encuentran un conjunto de áreas clave del proceso que para cumplirlas se deben realizar un conjunto prácticas comunes que son propias de cada área clave del proceso. Se debe cumplir con las áreas clave del un nivel para poder pasar al siguiente nivel. La mayoría de organizaciones que desarrollan *software* y no están certificadas se sitúan en el nivel 1. Es posible retroceder entre los niveles de madurez cuando no se da un seguimiento continuo por medio de las prácticas comunes.

5. El CMMI es un modelo que califica la madurez de los procesos de *software* no solo a nivel de desarrollo sino de todas las etapas que involucra, la administración de productos y procesos, la ingeniería y el soporte. Las series ISO 9000 están centradas en la calidad de los productos y no de los procesos, pero involucra implícitamente algunas áreas clave del proceso que presenta SW-CMM. El caso de estudio realizado en Guatemala es la única empresa, hasta el momento, que vela por certificarse en el nivel 2 del SW-CMM por medio del seguimiento de las áreas clave del proceso y prácticas comunes que abarca el nivel. Certificarse para cada uno de los niveles puede llevar de 1 a 2 años y medio dependiendo del nivel de certificación.

RECOMENDACIONES

1. Evaluar ventajas y desventajas de acuerdo al proceso de desarrollo de *software* y analizar si es conveniente certificarse en el modelo.

2. Identificar los factores que inciden en los procesos de desarrollo de *software* de acuerdo a que afectan, como afectan, cuando afectan y a quien afectan, para tomarlos en cuenta en la planificación.

3. Establecer una política de mejora continua dando lugar a la calidad del proceso de desarrollo de *software* y del producto final independientemente de la metodología de desarrollo que se adopta.

4. Si una organización desea certificarse en el modelo SW-CMM debe saber que se encuentra en el nivel inicial aunque se haya identificado en algún nivel del modelo, e iniciar a escalar en los distintos niveles.

5. La transición de ISO 9000 y sus series para desarrollo de *software* a SW-CMM sitúan a la organización en el nivel 2 del modelo; no se recomienda de manera contraria porque SW-CMM es dinámico e ISO 9000 es estático.

BIBLIOGRAFIA

1. M. Paulk y B. Curtis. **The capability maturity model: guidelines for improving the software process**. E.E.U.U.: Editorial Addison Wesley, 1995.

2. D. Kenneth. **Guide to the CMM: understanding the capability maturity model for software**. 4ª ed. E.E.U.U.: Editorial A Process Inc. 1996

3. Kendall, Kenneth E. y Julie E. Kendall. **Análisis y diseño de sistemas**. 2ª ed. México: Editorial Prentice Hall Hispanoamericana, S.A. 1997.

4. Pressman, Roger S. **Ingeniería del software, un enfoque práctico**. 4ª ed. México: Editorial McGraw-Hill. 1998.

5. Senn, James A. **Análisis y diseño de sistemas de información**. 2ª ed. México: Editorial McGraw-Hill. 1992.

6. www.iso.com Julio 2002

7. www.sei.cmu.edu/cmm/cmm.html Julio 2002

8. www.udlp.edu.mx Noviembre 2001