UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
A C A T L Á N

"PROPUESTA DE MEJORA EN EL PROCESO DE PRUEBA DE SOFTWARE PARA ALCANZAR EL NIVEL 2 DE CMMI A TRAVÉZ DE RUP"

T E S I N A

QUE PARA OBTENER EL TÍTULO DE LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

P R E S E N T A:

ALMA DELIA MUÑOZ LÓPEZ

A S E S O R:

M. EN C. SARA CAMACHO CANCINO

NAUCALPAN. EDO DE MÉXICO,2005.

CONTENIDO

CAPITULO III : Fundamentos de la metodología RUP (*Rational Unified Process*)

INTRODUCCIÓN

Las organizaciones se han dado cuenta a lo largo del tiempo y después de muchas promesas que la mejora en los procesos, la adopción de nuevas metodologías y tecnologías durante el desarrollo de software dan resultados satisfactorios en la productividad y calidad del producto final. Sin embargo hoy en día seguimos enfrentándonos a proyectos caóticos y a organizaciones indisciplinadas que sufren problemas de presupuesto y de adaptación a la fecha de finalización en los sistemas ya que no disponen de la infraestructura y soporte necesario.

Sin embargo, incluso en organizaciones indisciplinadas se pueden producir proyectos con buenos resultados; cuando esto ocurre, es generalmente por el gran esfuerzo de un dedicado equipo de trabajo pero, apoyarse sólo en la disponibilidad del mismo personal no proporciona las bases para conseguir a largo plazo mejor productividad y calidad en toda la organización. Por el contrario, éstas mejoras pueden llevarse a cabo si centramos nuestros esfuerzos en adoptar un *modelo* que nos permita mejorar los procesos de Ingeniería de software.

Tras éstas observaciones, se hace evidente la construcción de un marco de madurez en las organizaciones que permita contar con procesos bien definidos, dirigidos, medibles, controlables y efectivos. El CMMI (*Capability Maturity Model Integration*) es un modelo que se utilizará en el presente trabajo para juzgar la capacidad y la madurez de los procesos de software que se siguen dentro de una empresa dedicada al desarrollo de sistemas Administrativos siendo, el área de mayor interés y el motivo principal de este trabajo la *"Gerencia de Control de Calidad "*.

El objetivo del presente proyecto es el de proponer estrategias de mejora para el proceso de revisión y pruebas de software que sigue el departamento de control de calidad de una empresa dedicada al desarrollo de Sistemas Administrativos; mostrando cómo el uso de RUP (*Rational Unified Process*) proporciona las herramientas necesarias para alcanzar el nivel 2 de madurez del modelo CMMI (*Capability Maturity Model Integration*).

El presente trabajo está dirigido a ingenieros de sistemas, miembros del grupo de aseguramiento de calidad del software, personas dedicadas a la mejora de procesos y en general a quienes estén interesados en comprender el nuevo modelo CMMI y la metodología RUP .

A lo largo del *Capítulo I* se definirá al Proceso de desarrollo de software y al modelo ideal que debe tomarse como referencia para ésta actividad. Así mismo, se describirán las 6 mejores prácticas que contribuyen a la mejora continua de dicho proceso.

El *Capítulo II* hará hincapié en la <<madurez del proceso >>; mostrándose los principios, objetivos, estructura, niveles de madurez y su respectivo agrupamiento en las áreas de proceso (*PA – Process Areas*) del modelo CMMI (*Capability Maturity Model Integration*) y en general, al conjunto de funciones de ingeniería de software que deberán estar presentes conforme las organizaciones alcanzan diferentes niveles de madurez en sus procesos.

En el *Capítulo III* se mostrarán los fundamentos de RUP (*Rational Unified Proccess*) como una metodología de desarrollo de software que integra las mejores prácticas descritas en el primer capítulo y que además contempla todos los aspectos a considerarse durante el ciclo de vida del desarrollo de software.

En el *Capítulo IV* se dará la base de cómo utilizar RUP para alcanzar el nivel 2 de CMMI dentro de una organización.

Y finalmente en el *Capítulo V*, se expondrá la situación actual del proceso de prueba de software que se sigue dentro del Departamento de Control de calidad de una empresa dedicada al desarrollo de sistemas administrativos. Se hará una evaluación de sus procesos y posteriormente se propondrán estrategias de mejora que combatan las deficiencias y problemas encontrados, y de esta manera; alcanzar el nivel 2 de madurez del Modelo CMMI utilizando RUP como una metodología para lograr ésta meta.

AGRADECIMIENTOS

Le agradezco a mis padres Jesús y Gloria todo el apoyo, confianza y empeño que han puesto en mi formación y en general por todo el cariño , regaños y consejos que me han brindado y que me han hecho ser la persona que ahora soy . A mi hermano Víctor que de igual manera comienza a abrirse paso y con quién he compartido muchas experiencias, algunas buenas y otras malas pero que después de todo permanecemos unidos apoyándonos el uno al otro. A ellos tres, les dedico especialmente éste logro; muchas gracias y los amo.

A todos mis amigos les agradezco las vivencias, los consejos y amistad que hemos compartido y mantenido a través de los años, no se sientan pero sería muy largo mencionar porque son importantes; reciban mi agradecimiento: Marlen, Chelita, Miguel Angel, Janet, Alfredo, Marco, Ibsen, Alejandro, Benito, Silvia, Leonardo, Endira, Rosa y Oscar.

A mi asesora Sara Camacho le agradezco el tiempo que invirtió al presente trabajo y a la disponibilidad y flexibilidad en su asesoría.

A mis profesores les agradezco la enseñanza y las exigencias con las que me formaron y la experiencia profesional que algunos de ellos me compartieron.

Finalmente, agradezco a la Universidad Nacional Autónoma de México el haberme permitido ser parte de sus aulas y la institución donde me formé como profesional, donde viví muchas experiencias, donde hice amigos e invertí parte de mi vida.

CAPÍTULO I

INTRODUCCIÓN AL MEJORAMIENTO DEL PROCESO DE DESARROLLO DE SOFTWARE

En la última década la comunidad del software se ha hecho conciente que la calidad de los productos de software depende en gran medida de los procesos que se utilizan para su desarrollo y mantenimiento, por lo que se hace evidente la necesidad de irlos adaptando a los nuevos requerimientos de cada empresa e irlos mejorando con el paso del tiempo; es decir, se trata de implementar procesos disciplinados y vigentes que ayuden a liberar los productos en el tiempo y con el presupuesto estimado.

Este capítulo pretende explicar el proceso estratégico de un programa de mejora, de calidad y de implantación de una cultura de procesos. Se explicarán los elementos que deben de integrarse al plan: Gente, Tecnología y Modelos. Asimismo, se mostrarán algunos elementos importantes que la dirección y la gerencia de las organizaciones debe de tomar en cuenta para la concepción, puesta en marcha y culminación exitosa de un programa de mejora.

Por ésta razón, el foco de atención del presente capítulo se centra en el <<Proceso>> presentándose el modelo IDEAL que el SEI (*Software Engineering Institute*) ha creado para sentar las bases de lo que resulta ser un buen proceso así como la exposición de las 6 mejores prácticas que ayudan a mejorarlo de manera continua.

Por tanto el objetivo particular que persigue éste capítulo es :

" Describir los conceptos relacionados a la mejora de procesos de desarrollo de software"

1.1 ¿Qué es un proceso de desarrollo de software?

Antes de entrar a la definición del Proceso de software es importante identificar primero aquellos factores que intervienen en el desarrollo de un producto. Primeramente , se debe tomar en cuenta al factor humano, a la "tecnología" y finalmente al "proceso", elementos que en su conjunto deben trabajar de manera equilibrada a lo largo de todo el proceso de ingeniería de software tal y como se muestra en la figura 1.1 .

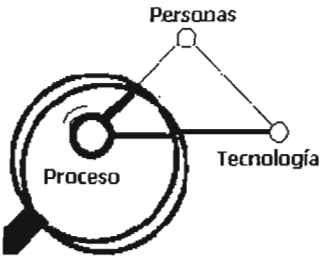


Figura 1.1: Factores que intervienen en el desarrollo de software

Para definir al proceso, a continuación se presentan algunas definiciones :

- El IEEE define un *proceso* como "una secuencia de pasos realizados para conseguir un propósito". [Watts ,1989]

Con lo anterior se puede definir al procesos de desarrollo de software como sigue:

- "El *Procesos de desarrollo de software* es un conjunto de actividades, métodos, prácticas y transformaciones que los individuos emplean para desarrollar y mantener el software, así como los productos asociados (e.j. Planes del proyecto, diseño de documentos, casos de uso, código, manuales de usuarios)". [Paulk,1985]

- "Un *proceso de Ingeniería de software* se refiere al conjunto de todas las tareas, herramientas, estándares, métodos y prácticas de apoyo que están involucradas en la producción y mantenimiento del software a través de todo su ciclo de vida" . [Brown,1996]

En general se puede decir que un proceso de desarrollo de software es aquel que define QUIÉN hace QUÉ cosa CUÁNDO y CÓMO para obtener productos de alta calidad, siendo éste la guía para todos los participantes en el desarrollo (usuarios, analistas, desarrolladores, ingenieros de prueba , etc.) que permitirá construir software más ordenado dentro de los tiempos y presupuestos planeados. Cabe destacar, que no existe un proceso universal que aplicar ya que éste depende de las necesidades específicas de cada organización en términos de su gente, tecnología y herramientas. Lo que si es evidente es que debe existir una mejora continua del proceso que conlleve al desarrollo de enfoques cada vez más robustos. Se debe estar consciente que algunos de los elementos que conforman al proceso deberán permanecer sin cambio alguno mientras que algunos otros podrán variar.

Se debe tener cuidado de no convertir a un proceso en un procedimiento burocrático ya que un proceso debe estar abierto a un mejoramiento continuo. Si se rechaza la idea de que el proceso puede ser erróneo nos mantendremos en el error y peor aún , ese error se diseminaría a lo largo de todo el proyecto causando grandes pérdidas.

A continuación, en la *figura 1.2* se describen algunos de los aspectos que involucran una mejora en los procesos.

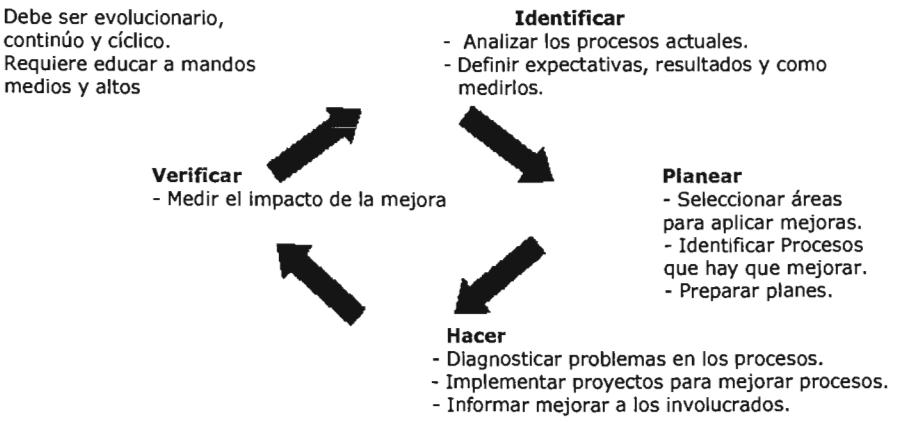


Figura 1.2: Factores que intervienen en el desarrollo de software

De igual forma, los procesos deben estar soportados por la planeación y el trabajo en equipo, ya que son la base para desarrollar productos de alta calidad .Definir el qué, cuándo, cómo y quién debe hacer las tareas diarias en la empresa, son los elementos básicos del proceso.

1.2 Modelo para el proceso de desarrollo de software

Un modelo de proceso software es una representación abstracta de la arquitectura, el diseño o la definición del proceso de desarrollo de software. Cada una de estas representaciones describen a distintos niveles de detalle, la estructura de los elementos del proceso y que pueden ser usados para :

a) proveer un marco de referencia a fin de comprender, experimentar y razonar sobre el proceso
b) facilitar la automatización del proceso y
c) proporcionar una base para su control

El objetivo de la modelización del proceso software es orientar, controlar y gestionar las actividades que involucra. Tradicionalmente, los modelos de proceso software han centrado sus representaciones en tres características elementales del proceso: la actividad, el artefacto y el agente (humano y computarizado), pero empíricamente está demostrada la gran influencia en el proceso de otras características tales como: los roles de los humanos y la organización.

1.3 El Modelo Ideal

Las organizaciones reconocen cada vez más la necesidad de ser específicamente guiadas en una implementación cuando adoptan nuevas herramientas de ingeniería de software, procesos y métodos. La mejora del proceso de software, la administración de riesgos continua y la introducción de un ambiente nuevo de desarrollo son una tarea tan compleja y sus efectos sumamente difíciles de alcanzar, que se requiere un enfoque especializado y sistemático para manejar el ciclo de vida de la adopción de la nueva tecnología.

Es por ésta razón que el SEI (*Software Engineering Institute*) ha desarrollado y ha refinado el modelo IDEAL (sigla formada con las primeras letras de las palabras inglesas que identifican las fases de Iniciar (*Initiating*), Diagnosticar (*Diagnosing*), Establecer (*Establishing*), Actuar (*Acting*) y Difundir (*Leveraging*) para ayudar a satisfacer esta necesidad. El modelo IDEAL proporciona un enfoque usable y entendible a la mejora continua del proceso resumiendo los pasos necesarios para establecer un programa exitoso. Seguir las fases, actividades y principios del modelo provee beneficios y disminuye los esfuerzos de la adopción de una nueva cultura de proceso.

Dicho modelo igualmente proporciona un enfoque ingenieril que se centra en manejar el programa de mejora a largo plazo. El modelo se compone de cinco fases las cuales se ven representadas en la *figura 1.3.*

1.3.1 Fases del Modelo

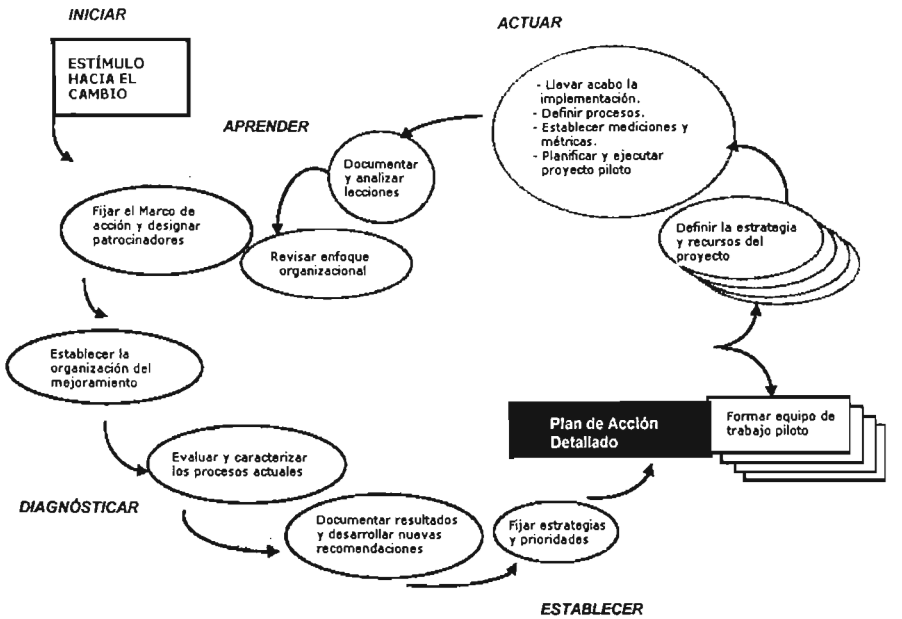


Figura 1.3:Fases del Modelo Ideal

El propósito y las actividades principales de cada etapa se resumen a continuación.

1.3.1.1 Iniciar (Initiating)

Su propósito es establecer los fundamentos básicos para garantizar que la iniciativa de mejoramiento del proceso se va a llevar a cabo. En esta fase, se aclara con la gerencia cuales son los objetivos de la empresa u organización que serán cubiertos; sustentando y justificando cada uno de ellos a fin de lograr el apoyo de la alta dirección y gerencia; pues ellos, serán los responsables de ofrecer y autorizar la disponibilidad de recursos e infraestructura para la puesta en marcha del programa de mejoramiento.

Actividades principales de la etapa
Estímulo para iniciar el mejoramiento:
Es el detonante de la iniciativa y puede provenir de la necesidad de hacerse más competitivo, de una decisión de la gerencia ó por exposición externa.
Es importante reconocer cuales son las razones que motivan a la empresa e identificar los

aspectos comerciales u organizacionales que se pretende asegurar (mejores costos, tiempo de desarrollo , mejor calidad, etc.)
Establecimiento del contexto: Consiste en definir las metas que el programa de mejoramiento busca alcanzar así como las áreas que participarán en el proyecto de mejora.
Establecer patrocinio de la gerencia: Lograr el patrocinio de la alta dirección asegurará que los mandos medios tendrán los incetivos necesarios para lograr los objetivos originalmente planteados; tomando en cuenta que el apoyo debe ser consecuente y efectivo.
Establecer infraestructura para el mejoramiento: Se debe contar con un mecanismo capaz de dirigir e implementar el proyecto de mejoramiento. Se debe capacitar a los distintos niveles de gerencia, de responsables y de personal de proyecto.

1.3.1.2 Diagnosticar (Diagnosing)

Su propósito es evaluar mediante un método formal la fortaleza y debilidades del proceso seguido por los proyectos. Los objetivos del programa se relacionan con las prácticas existentes y se determinan aquellas que no están suficientemente desarrolladas.

Actividades principales de la etapa
Evaluar y caracterizar el estado de prácticas: Es equivalente a identificar el punto de partida y el destino al que se quiere llegar. Para ello se puede tomar un modelo de referencia que ayude a determinar el estado actual de los procesos y que además proporcione información necesaria sobre lo que se debe hacer para alcanzar el nivel deseado. Un ejemplo de modelo es CMMI.
Desarrollar recomendaciones y documentar los resultados de la fase: Consiste en documentar las debilidades y fortalezas en los procesos que hayan sido encontradas durante la evaluación. Dicha información servirá como entrada al plan de acción para el mejoramiento. La salida es generalmente un informe de resultados.

1.3.1.3 Establecer (Establishing)

Esta etapa tiene como propósito diseñar un plan detallado que sirva de guía para la puesta en marcha del proyecto de mejoramiento Se deberán establecer las estrategias y las prioridades de la empresa en base a sus recursos y necesidades ya que solucionar todas las debilidades de manera simultánea conlleva un alto costo.

Actividades principales de la etapa
Establecer los equipos de acción de procesos: La mejor comprensión de las necesidades que se ha ido construyendo en los pasos previos permite establecer a estas alturas la estrategia y los recursos necesarios para completar el proyecto. Se identifica al equipo de trabajo, tecnología y demás recursos que participarán en la tarea de mejoramiento.
Elaboración del plan de acción: Las recomendaciones de la evaluación se transforman en un plan concreto que satisface las prioridades y necesidades de la empresa. Se convierte en la guía maestra del mejoramiento de procesos. Habitualmente considera acciones de corto, mediano y largo plazo. El plan incluye calendarios de proyecto, tareas, hitos, puntos de decisión, recursos, responsabilidades, métricas, mecanismos de seguimiento, riesgos con sus respectivas estrategias de mitigación así como otros elementos requeridos por la organización.

1.3.1.4 Actuar (Acting)

El propósito es simplemente implementar el mejoramiento de procesos llevando a cabo el plan de acción. Aquí se introducen o mejoran los procesos (e.g: modelamiento , introducción de nuevas metodologías, etc.), se entrena a los respectivos niveles de personal, se miden los avances / beneficios logrados, se realizan proyectos pilotos, se implantan los procesos mejorados en los proyectos nuevos o existentes y se hacen mini – evaluaciones para constatar la evolución del plan.

Actividades principales de la etapa
Planificar, ejecutar y seguir la instalación: Crear la "mejor solución" para resolver las necesidades de la organización implica la integración de las herramientas, conocimientos, información de procesos y habilidades tanto existentes como recién introducidas. Pueden provenir del interior de la empresa o de consultores expertos.
Planificar y ejecutar proyectos piloto: Una vez que las soluciones han sido diseñadas, se necesita probarlas en proyectos pilotos antes de decidir institucionalizarlas en el resto de los proyectos.
Refinar la solución: Cuando la solución propuesta ha sido aplicada en un proyecto piloto ésta se puede refinar para reflejar el conocimiento, la experiencia y las lecciones aprendidas en el ensayo. Se pueden

requerir varias iteraciones antes de alcanzar una solución satisfactoria.
Implementar la solución: Una vez que se ha decidido que se tiene una solución aceptable, se procede a aplicarla a lo largo de la organización. Se puede implantar usando una variedad de alternativas para su despliegue, dependiendo de la naturaleza de cada caso o condiciones existentes en los proyectos.

1.3.1.5 Difundir (Leveraging)

El propósito es aprender de la experiencia del ciclo recién realizado y aumentar la habilidad de la empresa u organización para mejorar los procesos de manera continua.

Se determinan los logros, el esfuerzo invertido, la manera en que las metas fueron satisfechas y la forma más adecuada de implementar cambios en el futuro. Se utiliza las mediciones y registros acumulados durante la aplicación de las etapas anteriores del ciclo.

Actividades principales de la etapa
Documentar y analizar las lecciones: Esta actividad identifica el grado en que el esfuerzo invertido logró los propósitos deseados, las cosas que trabajaron bien y como se podrían hacer mejor durante el ciclo de mejoramiento siguiente. Las lecciones se recolectan, se analizan, se resumen y se documentan. Se reexaminan las necesidades de la empresa identificadas en la fase de inicio para ver si fueron cubiertas.
Revisar el enfoque seguido y proponer acciones futuras: Se desarrollan y documentan las recomendaciones que resultan del análisis y validación del proceso. Se proponen pautas y acciones para el siguiente plan de mejoramiento. Generalmente el final del ciclo coincide con las primeras etapas del ciclo siguiente, así que se recomienda efectuar una nueva evaluación para determinar las nuevas necesidades y fortalezas que servirán de entrada al nuevo plan de acción.

1.4 Mejores prácticas para el proceso de desarrollo de software

A través del estudio y análisis de miles de empresas que desarrollan software durante los últimos 19 años, se ha identificado que las "mejores prácticas" para el desarrollo de software "son aquellas metodologías que tienen un mayor impacto en la mejora del proceso de desarrollo. Son aquellas que más contribuyen a desarrollar aplicaciones con alta calidad y dentro del tiempo y presupuesto planeados."[Perks,2003]

Las mejores prácticas de las que se hacen mención son las siguientes:

1. *Desarrollo Iterativo*
2. *Administración de Requerimientos*
3. *Arquitectura de componentes*
4. *Modelado visual con UML*
5. *Verificación continua de la calidad*
6. *Control de cambios y configuraciones*

1.4.1 Desarrollo iterativo

Reducir el riesgo y el tiempo

Debido al grado de complejidad actual de los sistemas de software, no es posible secuencialmente definir primero todo el problema, luego diseñar la solución completa, codificar el software y finalmente probar todo el producto ["desarrollo en cascada"]. Es necesario utilizar un acercamiento iterativo que permita un entendimiento incremental del problema a través de refinaciones sucesivas, creando de manera incremental una solución efectiva mediante múltiples iteraciones tal y como se muestra en la *figura 1.4*.

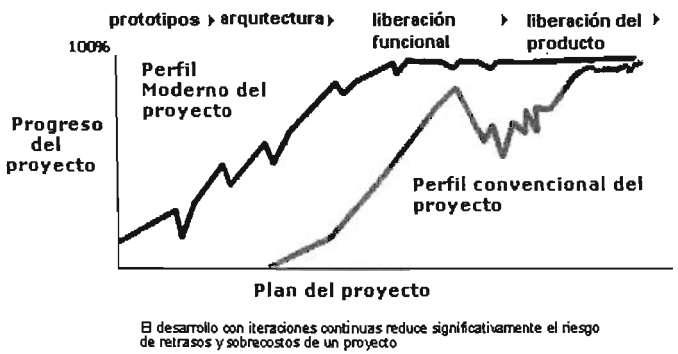


Figura 1.4. Desarrollo Iterativo.

Como cada iteración termina con una liberación ejecutable, el equipo de desarrollo permanece enfocado en producir resultados, así como las revisiones frecuentes del status aseguran que el proyecto se mantenga dentro del calendario preestablecido. El desarrollo iterativo también facilita la inclusión de cambios tácticos a los requerimientos, a las características o al calendario.

1.4.2 Administración de requerimientos

El primer paso del éxito del proyecto

Estudios del Standish Group de 1997 al 2000 revelan que una de las principales causas para el fracaso de un proyecto de software es la mala (o ausencia de) administración de requerimientos. Los principales problemas de un mal manejo de requerimientos son:

1. Incapacidad para manejar los cambios en los requerimientos durante el desarrollo.
2. Falta de especificación detallada de los requerimientos.
3. Mala organización y control de requerimientos y
4. Requerimientos mal entendidos.

La administración de requerimientos comprende las actividades relacionadas con la definición, clasificación, asignación, seguimiento y control de los requerimientos durante todo el ciclo de vida de desarrollo de software. Es una metodología indispensable para el aseguramiento de la calidad de los productos, así como para el control y seguimiento de los proyectos.

1.4.3 Arquitectura de componentes

La base para la reutilización de software

Uno de los peligros en sistemas grandes y complejos es que al hacer cambios a una parte de la aplicación se impacten otras partes con el "efecto dominó". Los equipos de desarrollo exitosos resuelven estas dependencias potenciales a través del uso de arquitecturas basadas en componentes. El ensamble de aplicaciones mediante módulos manejables crea arquitecturas resistentes y flexibles que evitan que los cambios al software conlleven esfuerzos masivos.

A medida que las organizaciones reutilizan componentes, reducen la cantidad de código a generar para cada aplicación. Así logran acelerar el tiempo de desarrollo y reducen la probabilidad de introducir errores al sistema.

1.4.4 Modelado visual con UML

Los planos del éxito

Para describir la complejidad de las operaciones actuales, los equipos de desarrollo necesitan una manera altamente descriptiva, intuitiva y precisa para modelar toda la funcionalidad. De lo contrario no serían capaces de comprender, especificar o documentar los sistemas que deben de construir. La mejor manera de modelar es utilizando UML (*Unified*

Modeling Language) que es el lenguaje de modelado creado por Rational y que se ha convertido en el estándar de la industria.

UML permite describir un sistema en diferentes niveles de abstracción, simplificando la complejidad sin perder información, para que tanto usuarios, líderes y desarrolladores puedan comprender claramente las características de la aplicación.

1.4.5 Verificación continua de la calidad

Construyendo con calidad de principio a fin

No es secreto que es mucho más costoso detectar y arreglar errores ya en producción que encontrarlos en etapas tempranas. Los principales factores de rechazo de una aplicación son desempeño deficiente y baja confiabilidad. Por lo tanto, la calidad debe ser comprobada siempre con respecto a los requerimientos estipulados previamente sobre funcionalidad, confiabilidad y desempeño (de la aplicación y del sistema.)

La manera más efectiva de reducir los defectos en el desarrollo, es comenzar a realizar las pruebas desde el inicio del ciclo de vida de desarrollo y seguir probando incrementalmente toda la aplicación en cada iteración. Con este acercamiento, los defectos se corrigen tan pronto como se introducen en el producto y no al final, cuando su impacto es mayor.

Adicionalmente, el conocer qué funcionalidades se han completado, es decir, codificado y probado, le permite al gerente del proyecto tener un mayor control sobre la aplicación y tener certeza del estado y grado de avance real. Asimismo, el probar la aplicación en cada iteración, promueve una planeación y comunicación entre desarrolladores y testers, lo cual crea una dinámica de equipo altamente enfocada, con recompensas substanciales.

1.4.6 Control de cambios y configuraciones

Controlar el proyecto y eliminar los retrasos

Los cambios son un elemento medular de la vida del desarrollo de software. El trabajo efectivo requiere de una administración formal de los cambios. Cuando se cuenta con una administración de cambios del software realmente efectiva se logra que:

1. Los equipos de desarrollo pueden hacer liberaciones a tiempo, en presupuesto y con una calidad predecible

2. Los líderes de proyecto conozcan en todo momento el estado y avance del desarrollo y tengan certeza en el tiempo de liberación
3. Los testers sepan cuando una nueva construcción requiere ser probada y qué mejoras o correcciones debe presentar
4. Los desarrolladores manipulen y controlen con orden y seguridad sus enormes colecciones de archivos y componentes diferentes para cada aplicación

Las organizaciones de desarrollo exitosas entienden que el control de cambios durante todo el ciclo es la clave para asignar prioridades a las actividades del equipo, así como para controlar la complejidad inherente del desarrollo. Sin ello, el caos de los cambios tomará control de todo proyecto.

En éste primer capítulo se han expuesto los fundamentos del proceso de desarrollo de software y la importancia de contar con buenas prácticas que garanticen una mejora continua en dicho proceso.

Tener presente los siguientes puntos:

- Un proceso de desarrollo de software es aquel que define QUIÉN hace QUÉ cosa CUÁNDO y CÓMO dentro del ciclo de vida de un sistema.
- El modelo IDEAL es un modelo que proporciona un enfoque usable y entendible en la mejora continua del proceso de desarrollo de software cuyas etapas son: Iniciar , Diagnosticar, Establecer, Actuar y Difundir.
- Las "mejores prácticas" para el desarrollo de software son aquellas metodologías que tienen un mayor impacto en la mejora del proceso de desarrollo y son: *Desarrollo Iterativo, Administración de Requerimientos, Arquitectura de componentes, Modelado visual con UML, Verificación continua de la calidad, Control de cambios y configuraciones.*

CAPÍTULO II

FUNDAMENTOS DEL MODELO CMMI (CAPABILITY MATURITY MODEL INTEGRATION)

El modelo CMMI fue creado a fin de proporcionar a las organizaciones consejos que les permitan obtener ganancias en el control de sus procesos de desarrollo y mantenimiento del software siendo, una guía en la selección de estrategias de mejora y como el camino que permite determinar los niveles de madurez dentro de una organización identificando las cuestiones mínimas para la calidad del software y mejora de procesos.

A lo largo de éste capítulo se describirán los componentes, áreas de procesos, estructura, representaciones y niveles de capacidad del modelo a fin de alcanzar el siguiente objetivo particular :

"Dar a conocer los fundamentos del modelo CMMI y cada uno de los lineamientos que toda organización debe cumplir para alcanzar cierto nivel de madurez en sus procesos de desarrollo y mantenimiento de software."

2.1 Breve historia de CMMI

A principios de la década de los 80´s, una firma dedicada al estudio del mercado de Tecnologías de Información pública un reporte sobre el éxito de los proyectos de desarrollo en la industria del software. El reporte basado en encuestas hechas sobre proyectos de software informaban los siguientes resultados estadísticos:

- El 30% de los proyectos se cancelaban
- El 54% de los proyectos excedían ampliamente los tiempos y costos estimados
- El 16% de los proyectos finalizaban exitosamente dentro del tiempo, el costo y la funcionalidad prevista

En respuesta a la situación alarmante del momento, el Departamento de defensa de los EEUU funda el SEI (*Software Engineering Institute*) en la Universidad de Carnegie Mellon, con el propósito de estudiar el problema y encontrar alguna solución.

En 1991, el SEI publica el modelo CMM (*Capability Maturity Model*) que está orientado a la mejora de los procesos relacionados con el desarrollo de software para lo cual, contempla las consideradas mejores prácticas de ingeniería de software que fueron previamente descritas en el capítulo anterior. A partir de ese momento, el Departamento de Defensa exige que sus proveedores estén certificados en CMM lo que impulsa a que el modelo tenga una amplia aceptación y se convierta en un estándar de facto dentro de la industria del software. Algunas de las organizaciones que adoptaron el modelo fueron Accenture, AT&T, Boeing, Ericcson, Fuji, Xerox, Hewlet Packard, IBM, Motorola, Nasa, Samsung, Siemens y muchas otras.

Luego del éxito alcanzado por CMM y debido a su utilización a lo largo de los años 90 también se desarrollaron otros modelos de madurez para otras disciplinas y funciones tales como:

- SE-CMM (*Systems Engineering Capability Maturity Model*) – Modelo de Madurez para la Ingeniería de Sistemas.
- SA-CMM (*Software Acquisition Capability Maturity Model*) – Modelo de Madurez para las Compras y la Gestión de Proveedores.
- IP-CMM (*Integrated Product Capability Maturity Model*) – Modelo de Madurez para el desarrollo Integrado de Software y de Hardware.
- P-CMM (*People Capability Maturity Model*) - Modelo de Madurez para el desarrollo de las personas.

Aunque muchas organizaciones encontraron estos modelos muy útiles para la mejora de otros procesos distintos al proceso de desarrollo del software, la gran mayoría tuvo que luchar con los problemas de integrar los distintos modelos, solucionar las lagunas detectadas, resolver las inconsistencias y aclarar las diferentes terminologías.

Como consecuencia del estudio de estas dificultades y de la preparación de la siguiente generación de sus modelos de madurez, en diciembre de 2001, el Instituto de Ingeniería de Software (SEI) publicó la versión 1.1 del CMMI-SE/SW/IPPD (*Modelo de Madurez de Capacidad Integrado para varias disciplinas*).

El modelo CMMI tiene el propósito de proporcionar una única guía unificada para la mejora de múltiples disciplinas tales como ingeniería de sistemas (SE), ingeniería del software (SW) y el desarrollo integrado del producto y del proceso (IPPD). El nuevo modelo además agrega una nueva forma de representación además de la conocida representación por niveles. La nueva forma de representación se llama *Continua* y está orientada a medir la mejora en los procesos de manera individual en vez de hacerlo de manera conjunta como la representación por niveles.

En paralelo con el desarrollo del CMMI, el SEI elaboró un método para la evaluación formal del modelo denominado SCAMPI (*Standard CMMI Appraisal Method for Process Improvement*). El método define una serie de reglas para la evaluación del modelo, las cuales deben utilizarse para valorar las distintas partes del mismo durante una evaluación formal. Estas reglas hacen que sea necesario utilizar herramientas, ya que el método de evaluación deja de ser una simple encuesta para convertirse en una evaluación detallada.

2.2 ¿Qué es CMMI ?

"EL Modelo de Madurez de Capacidades Integrado ("*Capability Maturity Model Integration*") es un marco de trabajo que describe los elementos claves de un proceso de software eficaz y que además integra los procesos de ingeniería de software así como los procesos de ingeniería de sistemas. Describe un camino de mejoramiento evolutivo para pasar de un proceso inmaduro a un proceso maduro ".[Royce,2002]

Dicho modelo, ayudará a que las organizaciones inmaduraz entendiendo esto cómo aquellas que generalmente improvisan sus procesos durante el proyecto, que exceden sus presupuestos y calendarios al no basarse en estimaciones realistas ó bien que cumplen con las fechas pero comprometen la calidad y funcionalidad del producto; se conviertan en organizaciones maduras caracterizadas por:

- Poseer la habilidad organizacional para administrar los procesos de desarrollo y mantenimiento de sistemas.
- Comunicar en forma precisa el proceso de software al personal existente y a los nuevos empleados.
- Trabajar en base a un plan con procesos consistentes.
- Monitorear la calidad de los productos.
- Contar con información cuantitativa para evaluar la calidad.
- Basar sus calendarios y presupuestos en el desempeño histórico viendo las cosas de forma realista.

Igualmente y a diferencia del modelo CMM; CMMI integra en un único marco de referencia las disciplinas de: Ingeniería de software, Ingeniería de sistemas, Desarrollo integrado de procesos y productos que promueve el establecimiento de un entorno de trabajo en equipo y una Gestión más proactiva y colaborativa de las relaciones con proveedores;

cubriendo además aspectos de los estándares ISO 15504 e ISO 9000. (*Ver detalles de estos estándares en la sección de ANEXOS).*

2.3 Componentes del Modelo CMMI

A continuación, se da una descripción de cada uno de los componentes que conforman al modelo CMMI como lo son : las áreas de proceso, metas y prácticas tanto específicas como genéricas que una empresa debe de cubrir dependiendo del nivel de madurez que desee alcanzar. En la *figura 2.1* se muestran cada uno de éstos componentes.

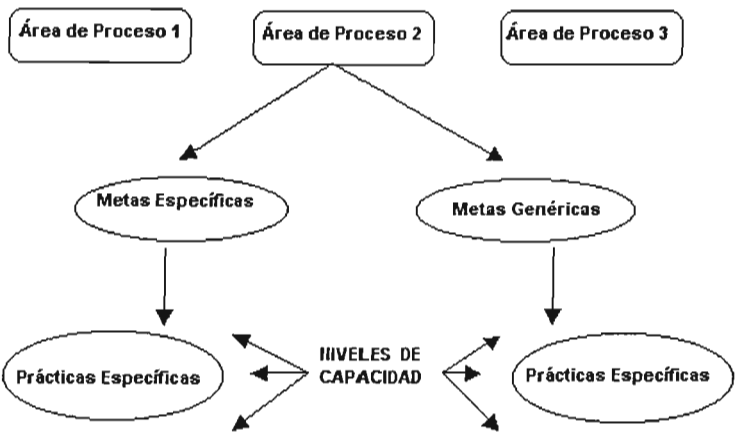


Figura 2.1: Componentes del modelo CMMI.

2.3.1 Áreas de proceso (PA´s)

Las áreas de procesos son aquella que definen la dimensión del proceso del modelo contando CMMI con 25 áreas de proceso las cuales se encuentran distribuidas en las siguientes categorías y según se muestra en la *figura 2.2.*

Figura 2.2. Áreas de Proceso del modelo CMMI

2.3.1.1 Administración de Procesos

Contiene las actividades relacionadas a la definición, planeación, asignación de recursos, despliegue, implantación, monitoreo, control, evaluación, medición y mejoramiento de procesos. Las áreas que ésta categoría incluye son:

Básicos

- Enfoque organizacional en proceso **(OPF)**
- Definición de procesos de la organización **(OPD)**
- Capacitación organizacional **(OT)**

Avanzados

- Desempeño de procesos organizacionales **(OPP)**
- Innovación y despliegue organizacional **(OID)**

A continuación, en la *figura 2.3* se describe como llevar a cabo una Administración de Procesos Básica.

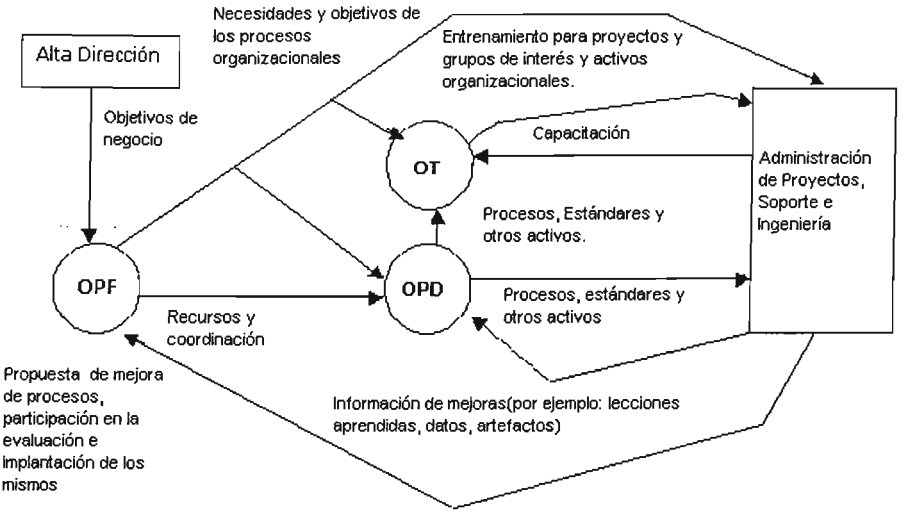


Figura 2.3. Administración de Procesos Básica.

2.3.1.2 Administración de Proyectos

Las áreas de Administración de Proyectos cubren las actividades relacionadas con la planeación, monitoreo y control del proyecto y se clasifican en:

Básicos

- Planeación de proyectos **(PP)**
- Monitoreo y control de proyectos **(PMC)**
- Administración del acuerdo con el proveedor **(SAM)**

Avanzados

- Administración integral de proyectos **(IPM)**
- Administración del riesgo **(RSKM)**
- Administación cuantitativa de proyectos **(QPM)**
- Equipos integrados **(IT)**
- Administración integrada del proveedor **(ISM)**

Una Administración de Proyectos básica es la que se describe a continuación en la *figura 2.4*.

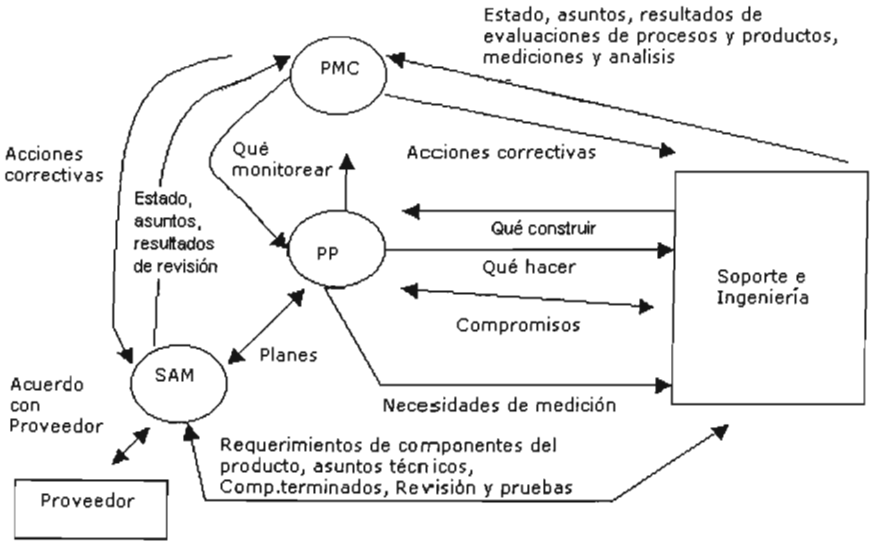


Figura 2.4. Administración de Proyectos básica.

2.3.1.3 Procesos de Ingeniería

Los Procesos de Ingeniería cubren las actividades de desarrollo y mantenimiento que son compartidas por las diferentes disciplinas de ingeniería como lo son:

- Desarrollo de Requerimientos (RD)
- Administración de Requerimientos (REQM)
- Solución Técnica (TS)
- Integración de Producto (PI)
- Verificación (Ver)
- Validación (Val)

La forma en cómo todas éstas actividades interactúan entre sí se describe a continuación en la *figura 2.5*.

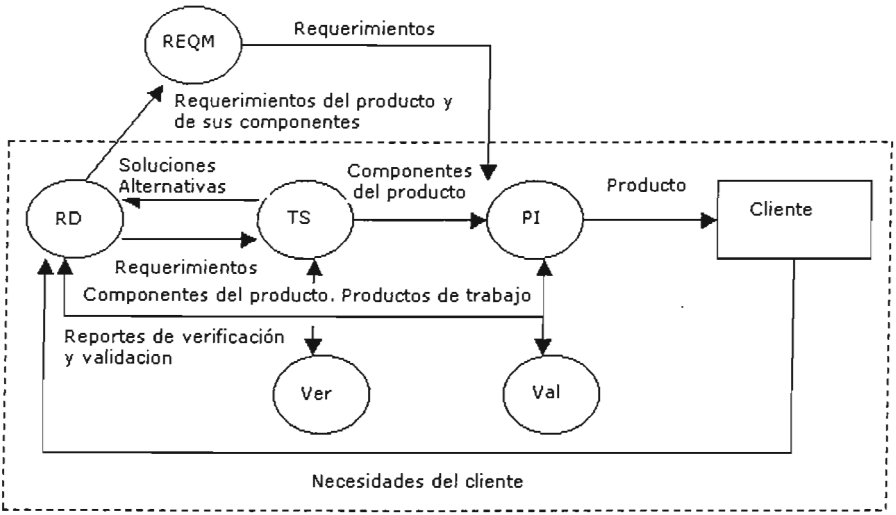


Figura 2.5. Proceso de Ingeniería

2.3.1.4 Procesos de Soporte

Las áreas de apoyo cubren las actividades de soporte del desarrollo y mantenimiento del producto y son las siguientes:

Básicos

- Administración de Configuración (CM)
- Aseguramiento de Calidad de Proceso y Producto (PPQA)
- Mediciones y Análisis (MA)

Avanzados

- Análisis de decisión y Resolución (DAR)
- Análisis Causal y Resolución (CAR)
- Ambiente Organizacional para la Integración (OEI)

El proceso de soporte básico se puede describir como se muestra en la *figura 2.6.*

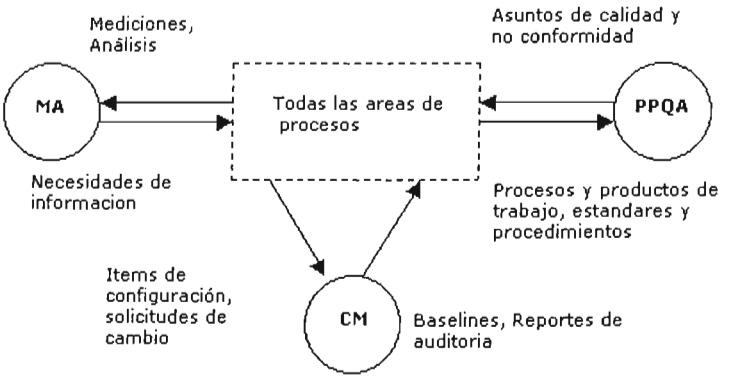


Figura 2.6. Proceso de Soporte básico.

2.3.2 Metas y Prácticas

Cada PA está formada de metas, a su vez que cada meta está formada de prácticas.

Las metas, son aquellas que hacen referencia a los objetivos que deben ser cubiertos en cada una de las áreas de proceso y éstas pueden ser tanto específicas como genéricas. Las metas específicas son aquellas que se aplican a sólo una PA, mientas que las metas genéricas se aplican a más de una de ellas.

Por otro lado, las prácticas, son aquellas actividades consideradas importantes para alcanzar una meta e igualmente se encuentran clasificadas en prácticas específicas y genéricas. Una práctica específica, es una actividad que se enfoca al cumplimiento de una meta específica y las prácticas genéricas son aquellas que se aplican a cualquier PA, ya que pueden mejorar el desempeño y control de cualquier proceso.

Un ejemplo de práctica y meta específica es:

PA: Administración de Requerimientos (REQM)

Meta: "Reflejar en planes, actividades y productos los requerimientos que han sido definidos".

Práctica: Identificar correctamente los requerimientos de cada proyecto.

Un ejemplo de práctica y meta genérica es:

Meta: "Institucionalizar un proceso definido"

Práctica: Proveer Recursos.

Finalmente cabe mencionar que el modelo CMMI tiene un total de 25 Pas, 70 Metas y 417 prácticas.

2.4 Niveles de Madurez del Modelo

El modelo CMMI está organizado en cinco niveles de madurez que permiten evaluar la capacidad y madurez de los procesos de ingeniería de software y de sistemas de una organización . Estos niveles y los cuales se encuentran representados en la *figura 2.7* son:

1. Desempeñado
2. Administrado
3. Definido
4. Administrado Cuantitativamente
5. Optimizado

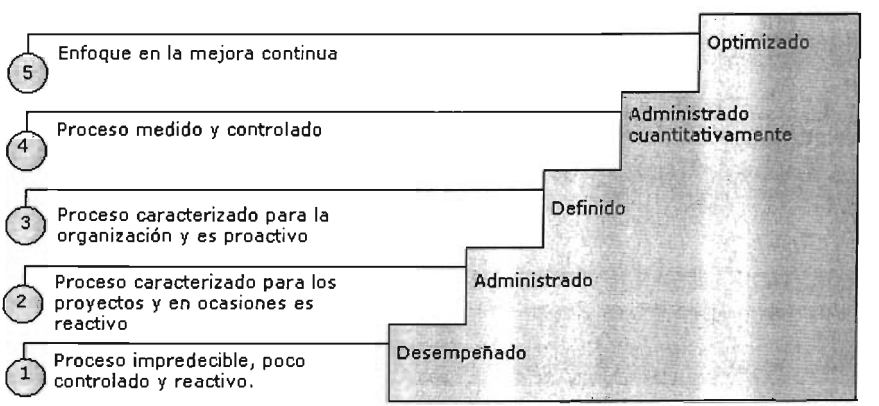


Figura 2.7. Niveles de madurez del modelo CMMI

2.4.1 Nivel Desempeñado

El proceso software está caracterizado por ser improvisado y en ocasiones hasta caótico; pocos procesos son definidos y los sucesos dependen de esfuerzos independientes. En éste nivel, la organización no proporciona un medio estable para el desarrollo y mantenimiento del software ya que carece de prácticas de dirección razonables y por lo tanto los beneficios de unas buenas prácticas de Ingeniería del software se pierden por una planificación ineficiente.

Durante una crisis, los procedimientos planificados de los proyectos son abandonados y vuelven a la codificación y las pruebas por lo que los resultados dependerán completamente de tener un excepcional directivo y un experimentado y efectivo equipo. Ocasionalmente, directivos muy capacitados de software pueden resistir las presiones pero cuando estos dejan los proyectos su estabilizadora influencia también se van con ellos; incluso un fuerte proceso de ingeniería no puede vencer la inestabilidad creada por la ausencia de las prácticas razonables del director.

"La capacidad del proceso software de las organizaciones de nivel 1 es imprevisible porque el proceso software está constantemente cambiando o modificándose conforme se avanza en el proyecto. Calendarios, presupuestos, funcionalidad y calidad del producto son igualmente imprevisibles".[Morales,2004]

2.4.2 Nivel Administrado

En éste nivel se establecen procesos básicos de dirección de proyectos para controlar costes, calendarios y funcionalidad de los sistemas, a su vez que se definen políticas para dirigir un proyecto de software y procedimientos para implementar esas políticas. La planificación y dirección de nuevos proyectos estará basada en la experiencia con proyectos similares.

Un objetivo a alcanzar en el nivel 2 es institucionalizar, es decir, construir la infraestructura y cultura de negocio necesarias para dirigir correctamente los proyectos software, permitiendo a las organizaciones repetir las prácticas desarrolladas satisfactoriamente en próximos proyectos, aunque los procesos implementados en cada proyecto sean distintos.

Los proyectos en las organizaciones de nivel 2 cuentan con procesos efectivos que se caracterizan por ser prácticos, documentados, formados, medibles y capaces de mejorar; así mismo, los alcances y compromisos de los proyectos son realistas ya que están basados en los resultados observados en anteriores proyectos y en los requisitos del actual proyecto.

"La capacidad de los procesos software en las organizaciones de nivel 2 pueden ser resumidos como disciplinados ya que la planificación y seguimiento del proyecto software es estable y puede ser repetido fácilmente".[Morales,2004]

2.4.3 Nivel Definido

El proceso software en organizaciones con éste nivel de madurez es tanto en las actividades de dirección como en las actividades de ingeniería; un proceso bien definido que se caracteriza por incluir buenos y completos criterios, entradas, estándares y buenos mecanismos de verificación que en su conjunto ayudarán al director de software y al personal técnico a ser más efectivos.

Generalmente, éstas organizaciones cuentan con un grupo que es el responsable de las actividades de organización del proceso software. Ej: Grupo de proceso de Ingeniería del software; a su vez que cuentan con un conjunto de programas que aseguran que el personal y la dirección tienen el conocimiento y destreza requerida para cumplir con el rol que les ha sido asignado.

"La capacidad de los procesos software en organizaciones de nivel 3 puede ser resumida en estándares consistentes, porque tanto la ingeniería del software y las actividades de dirección son estables y respetados. Dentro de éstas organizaciones las líneas de productos, los costes, calendarios y funcionalidad están bajo control pues la capacidad de los procesos está basada en un común: la organización comprende las actividades, roles y responsabilidades definidas en el proceso software."[Morales,2004]

2.4.4 Nivel Gestionado

En este nivel, las organizaciones en su conjunto cuantifican la meta de calidad tanto para el producto software como para los procesos; es decir, que disponen de métricas que les permiten medir la productividad y calidad dependiendo de la importancia de las actividades del proceso software a lo largo de todos los proyectos.

"La capacidad de los procesos de las organizaciones de nivel 4, se caracterizan por ser cuantificables y previsibles debido a que el proceso está medido y dirigido dentro los limites establecidos". Este nivel de capacidad de proceso permite a una organización predecir eventos en el proceso y la calidad del producto dentro los límites cuantificados. Cuando alguna circunstancia excepcional ocurre, debido a que el proceso es estable y medido, la causa de la variación puede ser identificada y dirigida. Cuando conocemos que los limites del proceso se han excedidos, se lleva a cabo una acción para corregir la situación. Los productos software son de una previsible alta calidad."[Morales,2004]

2.4.5 Nivel Optimizado

En el nivel optimizado, la organización entera se centra en la mejora continua de los procesos de software por contar con los medios necesarios para identificar las debilidades y fortalezas de los procesos que se están utilizando para a su vez impedir que ocurran defectos en las metas. Los efectivos datos del proceso software se usan para realizar los análisis de coste beneficios de las nuevas tecnologías y de los cambios propuestos para el proceso software de la organización. Se identifican y se transfieren por toda la empresa aquellas innovaciones que explotan las mejores prácticas de Ingeniería del software.

Los equipos del proyecto software analizan los defectos para determinar sus causas y de ésta manera los procesos software son evaluados para prevenir la repetición de defectos conocidos, y las lecciones aprendidas son difundidas para otros proyectos.

"La capacidad del proceso software se caracteriza por la continua mejora, porque las organizaciones de nivel 5 están continuamente esforzándose para mejorar el rango de capacidad de sus procesos; de este modo, mejoran el desarrollo de procesos software de sus proyectos. Las mejoras ocurren por el progreso incremental en los procesos existentes y por

innovaciones usando nuevas tecnologías y métodos. Tecnología y mejoras de procesos están planificadas y dirigidas como actividades de negocio ordinarias".[Morales,2004]

2.5 Representaciones del Modelo CMMI

Existen 2 representaciones del CMMI a fin de atender las diversas necesidades de las organizaciones que quieren realizar la mejora de sus procesos; éstas son:

- Representación Escalonada
- Representación Continua

2.5.1 Representación Escalonada

En la representación escalonada, cada área de proceso se asocia a uno de los cinco niveles de madurez del modelo. Los distintos niveles sirven como punto de referencia para conocer el *nivel de madurez* total que posee una organización. Una organización alcanza un nivel de madurez determinado cuando ha puesto en práctica todas y cada una de las áreas de proceso aplicables a ese nivel y a los niveles inferiores.

Aquellas organizaciones que escojan la representación escalonada, podrán proporcionar una secuencia contrastada de mejoras, comenzando con prácticas de gestión básica de proyectos hasta ir progresando por un camino predefinido y probado de sucesivos niveles que servirán de base para el siguiente nivel, teniendo como fin último la optimización de todos y cada uno de los procesos de la organización. Ver *figura 2.8.*

Figura 2.8. Representación escalonada

2.5.1.1 PA´s para una representación Escalonada

ÁREAS DE PROCESO DE NIVEL 2

- Administración de Requerimientos (REQM)
- Planeación de Proyectos (PP)
- Monitoreo y Control de Proyectos (PMC)
- Administración de Acuerdo con el Proveedor (SAM)
- Administración de Configuración (CM)
- Aseguramiento de Calidad de Proceso y Producto (PPQA)
- Mediciones y Análisis (MA)

ÁREAS DE PROCESO DE NIVEL 3

- Desarrollo de Requerimientos (RD)
- Solución Técnica (TS)
- Integración de Productos (PI)
- Verificación (Ver)
- Validación (Val)
- Enfoque Organizacional en Proceso (OPF)
- Definición de Procesos de la organización (OPD)
- Capacitación Organizacional (OT)
- Administración Integral de Proyectos (IPM)
- Administración de Riesgo (RSKM)
- Análisis de Decisión y Resolución (DAR)

ÁREAS DE PROCESO DE NIVEL 4

- Desempeño de Procesos Organizacionales (OPP)
- Administración Cuantitativa de Proyectos (QPM)

ÁREAS DE PROCESO DE NIVEL 5

- Innovación y Despliegue Organizacional (OID)
- Análisis Causal y Resolución (CAR)

2.5.1.2 Ventajas de la representación escalonada

- Proporciona una guía para implementación de : Grupos de áreas de proceso, secuencia de implementación.
- Estructura familiar para aquellos en transición del CMM.

2.5.2 Representación Continua

En la representación continua, los niveles de madurez no existen como tales. En cambio, los *niveles de capacidad* se designan para cada área de proceso, proporcionando un orden recomendado para acercarse a la mejora dentro de cada área de proceso. Una representación continua favorece la flexibilidad en el orden hacia el cual se dirigen las mejoras tal y como se muestra en la *figura 2.9*.

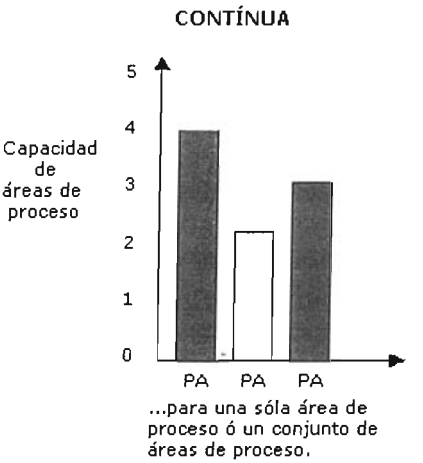


Figura 2.9. Representación Continua

2.5.2.1 PA´s para una Representación Continua.

Categoría	**Área de Proceso**
Administración de Proyectos	Planeación de Proyectos (PP) Monitoreo y Control de Proyectos (PMC) Administración de Acuerdo con el Proveedor (SAM) Administración de Riesgo (RSKM) Administración Cuantitativa de Proyectos (QPM)
Soporte	Administración de Configuración (CM) Aseguramiento de Calidad de Proceso y Producto (PPQA) Acciones y Análisis (MA) Solución Técnica (TS) Análisis Causal y Resolución (CAR)

Ingeniería	Administración de Requerimientos (REQM) Desarrollo de Requerimientos (RD) Solución Técnica (TS) Integración de Productos (PI) Verificación (Ver) Validación (Val)
Administración de Procesos	Enfoque Organizacional en Proceso (OPF) Definición de Procesos de la organización (OPD) Capacitación Organizacional (OT) Desempeño de Procesos Organizacionales (OPP) Innovación y Despliegue Organizacional (OID)

2.5.2.2 Ventajas de la representación Continua

- Proporciona máxima flexibilidad para enfocarse en áreas de proceso específicas de acuerdo a las metas y objetivos de negocio.
- Facilita la integración de nuevas disciplinas.
- Estructura familiar para aquellos en transición de la comunidad de ingeniería de sistemas.

De ésta manera se concluye el marco teórico del modelo CMMI (*Capability Maturity Model Integration*) en el que se hizo principal énfasis en la madurez del proceso y en su importancia dentro de una organización. Así, se explicaron los aspectos más importantes del modelo con el fin de evaluar los procesos de prueba que están vigentes dentro del Departamento de Control de Calidad de una empresa; tema que será desarrollado en el último capítulo.

Tener presentes los siguientes puntos:

- EL Modelo de Madurez de Capacidades Integrado ("*Capability Maturity Model Integration*") es un marco de trabajo que integra los procesos de ingeniería de software así como los elementos claves de un proceso de software eficaz .
- El CMMI está constituido por Áreas de Proceso, Metas y Prácticas.
- El CMMI está organizado en 5 niveles de madurez para evaluar la capacidad en los procesos de desarrollo de software de una empresa; éstos son: Desempeñado, Administrado, Definido, Administrado Cuantitativamente, Optimizado.
- Las 2 representaciones del Modelo son : La representación Escalonada y la representación Continua.

CAPÍTULO III
FUNDAMENTOS DE LA METODOLOGÍA RUP (RATIONAL UNIFIED PROCESS)

Hoy en día, las organizaciones se han hecho conscientes de que contar con procesos bien definidos, vigentes y documentados garantizará el éxito durante el proceso de desarrollo de sus productos de software. Es por ello, que la metodología RUP (*Rational Unified Process*) como proceso de ingeniería de software, brinda a las organizaciones un enfoque práctico, completo y escalable que asegura el uso de las mejores prácticas de ingeniería de software y que además se adapta a las necesidades de cada empresa.

Por lo tanto, el objetivo particular del presente capítulo es:

"Proporcionar una visión general de la metodología RUP como apoyo para el desarrollo de un proyecto informático de software de calidad ".

3.1 Breve historia de la Metodología

El antecedente más importante lo ubicamos en 1967 con la Metodología Ericsson (*Ericcson Aproach*) la cual fue una aproximación de desarrollo basada en componentes que introdujo el concepto de casos de uso. Entre los años 1987 a 1995 Jacobson funda la compañía "Objectory AB" y lanza el proceso de desarrollo Objectory (abreviación de *Objectory Factory*); posteriormente en 1995 "Rational Software Corporation" adquiere "Objectory AB" y es entre los años de 1995 y 1997 que se desarrolla "*Rational Objectory Process (ROP)*" fruto del encuentro y evolución de Objectory 3.8 y la Metodología Rational (Rational Aproach) que adopta por primera vez UML como lenguaje de modelamiento.

A principios de los noventas, la guerra de los métodos hizo evidente la necesidad de unificar criterios y es así como Grady Booch autor del método Booch y James Rumbaugh (desarrollador de General Electric) se unieron en Rational en 1994 para que en el año de 1995 Jacobson se les uniera. Gracias al esfuerzo de varias compañías y metodologistas, UML evolucionó hasta convertirse en un estándar en 1997 el cual es adoptado en todos los modelos ROP.

Desde ese entonces ya a la cabeza de Booch, Jacobson y Rumbaugh; Rational ha desarrollado e incorporado diversos elementos para expandir el ROP, destacándose el flujo de trabajo conocido como modelamiento del negocio para que de ésta manera en Junio de 1998 surgiera el Rational Unified Process 5.0 .

A continuación, en la *figura 3.1* se representa la evolución que el Rational Unified Process ha tenido con el pasar del tiempo.

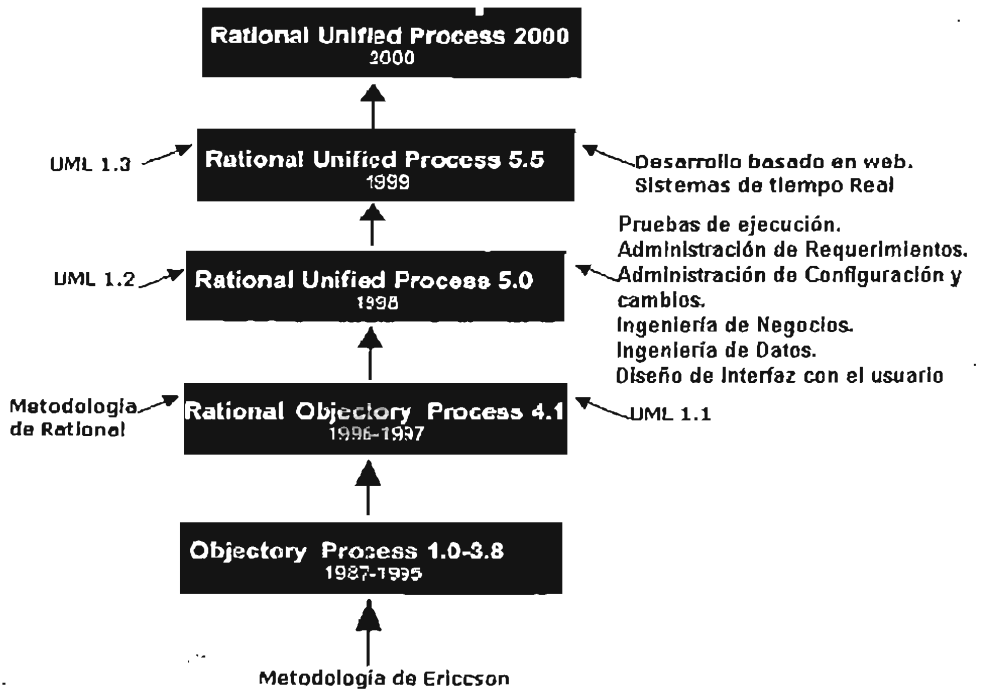


Figura 3.1: Evolución de RUP

3.2 ¿Qué es RUP (Rational Unified Process)

"El Proceso Unificado de Rational es un Proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendarios predecibles".[Kruchten,2001]

En definitiva el RUP es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software.

De igual forma, el Proceso Unificado captura las mejores prácticas de desarrollo de software que en capítulos anteriores se han descrito de una forma tal que es adaptable a un amplio rango de proyectos y organizaciones. En el aspecto de la gestión, el Proceso Unificado proporciona un enfoque disciplinado sobre cómo asignar tareas y responsabilidades dentro del equipo de desarrollo de sistemas.

3.3 Principales características de la Metodología RUP

3.3.1 Proceso dirigido por los casos de uso

Los Casos de Uso reemplazan la antigua especificación constituyendo la guía fundamental para las actividades a realizar durante todo el proceso de desarrollo que incluye: el diseño, la implementación y las pruebas del sistema.

Las actividades de desarrollo bajo el Proceso Unificado están dirigidas por los casos de uso poniendo un gran énfasis en la construcción de sistemas basados en una amplia comprensión de cómo se utilizará el sistema que se entregue. Las notaciones de los casos de uso y los escenarios se utilizan para guiar el flujo de procesos desde la captura de los requisitos hasta las pruebas proporcionando caminos que se pueden reproducir durante el desarrollo del sistema.

En la *figura 3.2* se muestran algunas actividades que pueden ser integradas por los casos de uso.

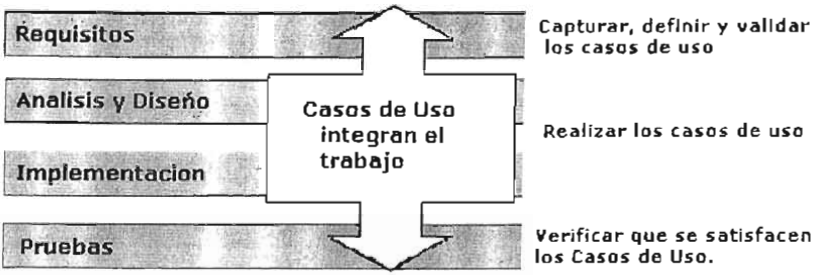


Figura 3.2: Actividades que integran los Casos de Uso.

3.3.2 Proceso iterativo e incremental

Para hacer más manejable un proyecto, RUP recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia que deben ser consideradas como un miniproyecto cuyo núcleo fundamental está constituido por una ó más iteraciones de las actividades principales de cualquier proceso de desarrollo. En concreto, RUP divide el proceso en cuatro fases las cuales serán descritas en el siguiente apartado y dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor ó menor hincapié en las distintas actividades.

El proceso Unificado de Rational se basa en la evolución de prototipos ejecutables que permiten comprender de una forma incremental los problemas por los que atraviesa el software. Como parte del enfoque iterativo se encuentra la flexibilidad para acomodarse a nuevos requisitos o a cambios tácticos en los objetivos del negocio. También permite que el proyecto identifique y resuelva los riesgos lo más pronto posible.

3.3.3 Proceso centrado en la arquitectura

El Proceso Unificado se centra en establecer los elementos más significativos del sistema con el fin de facilitar el desarrollo en paralelo, minimizar la repetición de trabajos, incrementar la probabilidad de reutilización de componentes y el mantenimiento posterior del sistema. Este diseño arquitectónico sirve cómo una sólida base sobre la cual se puede planificar y manejar el desarrollo de software basado en componentes. La arquitectura es como una radiografía del sistema que se está desarrollando lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo.

3.3.4 Proceso configurable

El proceso Unificado es un proceso configurable aunque un único proceso no es adecuado para las organizaciones de desarrollo de software, el proceso Unificado es adaptable y puede configurarse para cubrir las necesidades de proyectos que van desde pequeños equipos de desarrollo de software hasta grandes empresas de desarrollo. También se basa en una arquitectura de proceso simple y clara, que proporciona un marco común a toda una familia de procesos y que, además , puede variarse para acoplarse a distintas situaciones. Dentro del propio proceso Unificado se encuentran las guías sobre cómo configurar el proceso para adaptarse a las necesidades de una organización.

3.3.5 Proceso Integrado

La metodología RUP establece una estructura que involucra los ciclos, fases, flujos de trabajo, mitigación de riesgos, control de calidad, gestión del proyecto y control de configuración. Además, ésta estructura cubre a los vendedores y desarrolladores de herramientas a fin de soportar la automatización del proceso, soportar flujos individuales de trabajo, para construir los diferentes modelos y así ir integrando el trabajo a través del ciclo de vida.

3.3.6 Proceso que impulsa un control de calidad y una gestión de riesgo continua

El proceso Unificado evalúa continuamente la calidad del producto de software ya que ésta va contenida en el mismo proceso, en todas las actividades e implicando a todos los participantes mediante medidas y criterios objetivos. No se trata como algo a posteriori o como una actividad separada. Igualmente, la gestión del riesgo va contenida en el proceso de manera que los riesgos para el éxito del proyecto se identifican y se acometen al principio del proceso de desarrollo cuando todavía hay tiempo de reaccionar.

3.4 Arquitectura de la Metodología RUP

La arquitectura de la metodología RUP se centra en cuatro componentes que son:

- Los Roles (*workers*) que responden a la pregunta ¿quién?
- Las Actividades (*Activities*) que responden a la pregunta ¿cómo?
- Los flujos de trabajo (*workflows*) que responden a la pregunta ¿cuándo?
- Los productos (*artifacts*) que responden a la pregunta ¿qué?

3.4.1 Roles

Un Rol puede definirse como el comportamiento y conjunto de responsabilidades que un individuo ó grupo de trabajo tienen dentro un proyecto. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas.

El Rol es entonces quién define el conjunto de actividades y artefactos a ser desempeñados por cada uno de los implicados en el proyecto software. Algunos *ejemplos* de Roles son: Rol de Diseñador, Rol de Analista y Rol de Programador.

3.4.2 Actividades

Las actividades son el conjunto de tareas (pasos de concepción, realización y revisión) que deben ser desempeñadas de acuerdo al rol que haya sido asignado a cada una de las personas involucradas en el proyecto con el fin de crear ó modificar los artefactos. El objetivo de tales actividades es el de desarrollar ó actualizar un producto de software , incluyendo quizá el uso de herramientas para ayudar a automatizar algunas de ellas. Como *ejemplos* de actividades tenemos: Diseñar la interfaz del usuario, Programar un módulo específico, Diseñar casos de Prueba; e.t.c.
A continuación, en la *figura 3.3* se muestra un esquema de los Roles y Actividades.

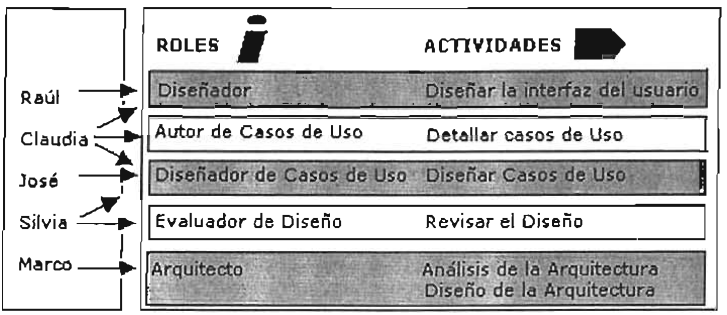


Figura 3.3: Roles y Actividades.

3.4.3 Flujos de trabajo

Los flujos de trabajo son una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos para obtener resultados observables. El RUP define varios flujos de trabajo distintos, entre los que se distinguen dos grupos: los de Ingeniería y los de Soporte.

Las distintas iteraciones a realizar consistirá en la ejecución de éstos flujos de trabajo con una mayor ó menor intensidad dependiendo de la fase de iteración en la que nos encontremos.

A continuación, en la *figura 3.4* se muestra un ejemplo de flujo de trabajo.

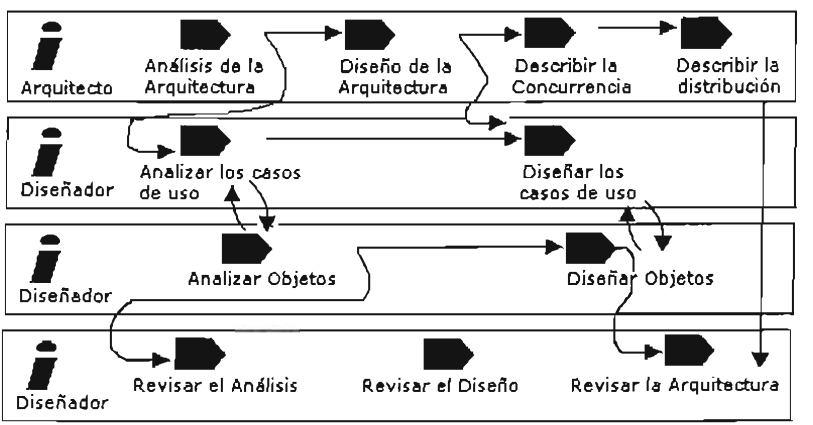


Figura 3.4.Ejemplo de un flujo de trabajo.

3.4.3.1 Flujos de Trabajo de Ingeniería

La clasificación de los flujos de Trabajo de Ingeniería es la siguiente:

a) Flujo de Modelado de Negocio
b) Flujo de Requerimientos
c) Flujo de Análisis y Diseño
d) Flujo de Implementación
e) Flujo de Pruebas y
f) Flujo de Despliegue

a) Flujo de Modelado de Negocio:

Con éste flujo de trabajo se pretende llegar a un mejor entendimiento de la organización en donde se implantará el producto con el fin de asegurar que éste será algo útil y no un obstáculo y para conseguir que encaje de la mejor forma posible en la organización y tener un marco común

para los desarrolladores, los clientes y los usuarios finales. Este flujo de trabajo no será siempre necesario. Si sólo añadimos funcionalidad que no verán los usuarios directamente, no hará falta.

Para modelar el negocio se pueden utilizar las mismas técnicas que para modelar software, es decir, que se tendrán casos de uso de negocio, actores de negocio, y diagramas. Dependiendo del tipo de software que se esté construyendo, el modelado de negocio cambiará aunque es importante destacar que no se trata de modelar la organización de arriba abajo, sólo la parte que corresponda. Una gran ventaja del modelado del negocio es que es una forma clara y concisa de mostrar las dependencias entre el negocio y el sistema que estamos construyendo.

b) Flujo de Requerimientos

Este es uno de los flujos de trabajo más importantes porque en él se establece *QUÉ* es lo que tiene que hacer exactamente el sistema que se construya. En esta línea, los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen.

"Los requisitos se dividen en dos grupos. Los *requisitos funcionales* que se refieren a aquello que el sistema puede hacer y que se modelan mediante diagramas de caso de uso y los *requisitos no funcionales* que son aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo , requisitos de usabilidad, fiabilidad, eficiencia, portabilidad etc."[Ollman,et al,2002]

Para capturar los requisitos es preciso entrevistar y anotar las peticiones de todos los usuarios finales así como de los interesados en el proyecto. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En éste flujo de trabajo y cómo parte de los requisitos de usabilidad se diseña la interfaz gráfica del usuario. Para ello, habitualmente se construyen prototipos de la GUI que se constatan con el usuario final y que resultan ser una buena forma de explicar requisitos . En definitiva, en este flujo de trabajo hay que analizar el problema, comprender las necesidades de los interesados y expresarlos en forma de requisitos; construir diagramas de casos de uso para los requisitos funcionales y los no funcionales describirlos textualmente en especificaciones suplementarias. Además hay que gestionar los cambios en los requisitos a lo largo de todo el proceso.

c) Flujo de Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describa como implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupe por ver *QUÉ* hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva *CÓMO* cumple el sistema sus objetivos.

El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los casos de uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando ésta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también , obteniendo un modelo de datos.

El resultado final más importante de éste flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas. Otro producto importante de éste flujo es la documentación de la arquitectura software, que captura varias versiones arquitectónicas del sistema.

d) Flujo de Implementación

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer los test de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de éste flujo de trabajo es el sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- Planear que subsistemas deben ser implementados y en que orden deben ser integrados formando el Plan de Integración
- Cada implementador decide en que orden implementará los elementos del subsistema. Si encuentra errores de diseño los notifica
- Se prueban los subsistemas individualmente
- Se integra el sistema siguiendo el Plan

La estructura de todos los elementos implementados forma el modelo de Implementación.

La integración debe ser incremental , es decir, en cada momento sólo se añade un elemento. De éste modo es más fácil localizar fallos y los componentes se prueban más a fondo.

En fases tempranas se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías ó diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) ó evolucionarios. Estos últimos llegan a transformarse en el sistema final.

e) Flujo de Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar ó rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida. "El papel del testeo no es asegurar la calidad, pero si evaluarla y proporcionar una retroalimentación a tiempo, de forma que las cuestiones de calidad puedan resolverse de manera efectiva en tiempo y coste".[Kruchten,2001]

Los principales aspectos a ser evaluados en un producto software son la Fiabilidad (resistente a fallos), la Funcionalidad (hace lo que debe) y el Rendimiento (lleva a cabo su trabajo de manera efectiva). Las pruebas pueden hacerse a diferentes niveles dependiendo del objetivo de los mismos, éstos pueden ser: **Test de unidad** (prueba las unidades mínimas por separado y normalmente se hace durante la implementación misma), **de integración** (varias unidades juntas), **de sistema** (sobre la aplicación y el sistema completo) y **de aceptación** (realizado sobre el sistema global por los usuarios ó terceros).

A la representación de lo que será probado y cómo debe de hacerse es a lo que se llama el **modelo de test**. Incluye la colección de casos de prueba, procedimientos , scripts y resultados esperados. Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de la evaluación del proyecto, así como las estrategias y recursos con que se denotará a esta tarea), ó incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida nos sirva para ir refinando el producto a desarrollar.

f) Flujo de Despliegue

El objetivo de éste flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final
- Empaquetar el software para su distribución
- Distribuir el software
- Instalar el software
- Proveer asistencia y ayuda a los usuarios
- Formar a los usuarios y al cuerpo de ventas
- Migrar el software existente ó convertir bases de datos

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito tanto del flujo como de la fase es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Aunque la ejecución de este flujo de trabajo debe empezar en fases anteriores para preparar el camino sobre todo con actividades de planificación pero también con la elaboración del manual de usuario.

3.4.3.2 Flujos de Trabajo de Soporte

La clasificación de los Flujos de Trabajo de Soporte es la siguiente:

a) Flujo de Administración de Proyectos
b) Flujo de Gestión de Configuraciones
c) Flujo de Entorno

g) Flujo de Administración del proyecto

El objetivo de la administración de un proyecto es conseguir equilibrar, completar los objetivos, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requisitos de los usuarios.

"Para conseguir este objetivo, el flujo de trabajo se centra en 3 aspectos:

- Planificar un proyecto iterativo y cada iteración particular
- Administrar el riesgo.
- Monitorizar el progreso del proyecto a través de métricas."[Kruchten,2001]

La planificación de un proyecto debe acometerse en dos niveles de abstracción: un "plan de grano grueso" para las fases y un "plan de grano fino" para cada iteración.

El plan de desarrollo (ó plan de fases) debe contener las fechas esperadas para los hitos principales, cuando se tendrá la arquitectura, cuando estará la primera versión beta e.t.c. Estas fechas coincidirán, generalmente, con el final de las fases. También deberían tener una previsión de las necesidades del personal y medios, así como fechas e hitos menores, sólo si se conocen. Este plan debe obtenerse temprano en la fase de inicio y debe actualizarse siempre que sea necesario.

Debe realizarse un plan de iteración por cada iteración, como cabría suponer. Este plan se elabora hacia la segunda mitad de la iteración, lo que significa que en un momento dado habrá dos planes activos : el de la iteración en curso y el de la próxima que es construido en ésta. En este plan se detallarán fechas importantes para la iteración: complicaciones importantes, revisiones ó llegada de componentes.

La administración de riesgo por lo tanto consiste en ocuparse de las incógnitas de un proyecto y de las cuestiones que pueden llevarlo a pique. En concreto hay que identificar los riesgos, típicamente en la fase de inicio, y hacerles frente; habrá que tratar de mitigar el riesgo y definir un plan de contingencia por si el riesgo se convierte en un problema real. En definitiva la administración del riego consistirá en gestionar una lista de riesgos.

h) Flujo de Gestión de Configuraciones

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

"Las causas por las que la evolución de los artefactos pueden causar problemas son:

Actualización simultánea: Se da cuando dos personas trabajan por separado sobre el mismo artefacto a la vez, el último en hacer las modificaciones sobrescribe lo hecho por el primero.

Notificación limitada: Cuando un problema ha sido resuelto en un artefacto compartido por varios roles y alguno de ellos no son notificados del cambio.

Múltiples versiones: Cuando se trabaja con diferentes versiones del producto del mismo tiempo en diferentes flujos de trabajo, pueden surgir problemas si los cambios no son convenientemente monitorizados y propagados."[Perks,2003]

i) Flujo de Entorno

La finalidad de éste workflow es dar soporte al proyecto con las adecuadas herramientas, proceso y métodos. Es decir, tener a punto las herramientas que se vayan a necesitar en cada momento, así como definir la instancia concreta de proceso unificado que se va a seguir.

En concreto las responsabilidades de éste flujo de trabajo incluyen:

- Selección y adquisición de herramientas
- Establecer y configurar las herramientas para que se ajusten a la organización
- Configuración del proceso
- Mejora del proceso
- Servicios técnicos

El principal artefacto que se usa en este flujo de trabajo es el *caso de desarrollo* que especifica como se aplicará el proceso unificado, que productos se van a utilizar y cómo van a ser utilizados. Además se tendrán que definir las líneas guías (los pasos concretos y políticas a seguir) para los distintos aspectos del proceso, como pueden ser : el modelado del negocio, diseño de los casos de uso, diseño de la interfaz de usuario, la programación y el manual de usuario.

"Las actividades que se deben llevar a cabo durante este flujo de trabajo son:

- Preparar el entorno para el trabajo.
- Preparar el entorno para una iteración.
- Preparar las líneas de guía para una iteración.
- Dar soporte al entorno durante la iteración." [Perks,2003]

3.4.4 Artefactos

Un artefacto es todo aquello que se necesita para producir ó modificar un proceso ; es decir, aquello que se va creando en cada una de las fases del proyecto hasta obtener el producto final. Un artefacto puede ser algún documento, informe ó ejecutable que se produce, se manipula ó se consume.

Cada actividad del Proceso Unificado de Rational lleva algunos artefactos asociados, bien sean requeridos como entradas ó bien sean generados como salidas. Algunos artefactos se utilizan como entradas de las actividades siguientes, se mantienen como recursos de referencia en el proyecto, o se generan en algún formato específico, en forma de entregas definidas en el contrato.

3.4.4.1 Artefactos modelos

Los "modelos" son el tipo de artefacto más importante en el Proceso Unificado de Rational. Hay nueve modelos que en conjunto cubren todas las decisiones importantes implicadas en la visualización, especificación, construcción y documentación de un sistema con gran cantidad de software. Son los siguientes:

1. ***Modelo de Negocio*** : Establece una abstracción de la organización.
2. ***Modelo del Dominio***: Establece el contexto del sistema.
3. ***Modelo de casos de uso***: Establece los requisitos funcionales del sistema.
4. ***Modelo de análisis*** *(opcional)* : Establece un diseño de las ideas.
5. ***Modelo de Diseño***: Establece el vocabulario del problema y sus solución.
6. ***Modelo del proceso*** *(opcional)* : Establece los mecanismos de concurrencia y sincronización del sistema.
7. ***Modelo de despliegue***: Establece la topología hardware sobre la cual se ejecutará el sistema.
8. ***Modelo de Implementación***: Establece las partes que se utilizarán para ensamblar y hacer disponible el sistema físico.
9. ***Modelo de pruebas***: Establece las formas de validar y verificar el sistema.

En cada uno de los flujos del ciclo de vida del desarrollo del software se trabaja con los modelos descritos, pero no todos al mismo tiempo, sino siguiendo una secuencia lógica determinada por el flujo de trabajo y la naturaleza del modelo. En la *figura 3.5* se muestra qué modelos se manejan en cada uno de los flujos de trabajo del proceso de desarrollo.

	Modelo del negocio	Requerimientos	Análisis	Diseño	Implementación	Pruebas	Despliegue
Modelo del negocio	X						
Modelo del dominio	X	X					
Modelo de Casos de Uso		X					
Modelo de análisis			X				
Modelo de diseño				X			
Modelo de procesos				X			
Modelo de despliegue				X			X
Modelo de Implementación					X		X
Modelo de pruebas						X	X

Figura 3.5. Modelos y flujos de trabajo del Proceso Unificado

En la *figura 3.6* se presenta la correspondencia de los modelos con los flujos de trabajo del proceso del ciclo de vida del software del Proceso Unificado. Pero, para simplificar, se muestra solamente los aspectos relacionados directamente con el desarrollo técnico del proyecto, obviando los flujos de modelado de negocio y despliegue. En consecuencia, se han eliminado los modelos del negocio, del dominio y de proceso ya que no están los flujos correspondientes.

	Requerimientos	Análisis	Diseño	Implementación	Prueba
Modelo de casos de uso	X				
Modelo de análisis		X			
Modelo de diseño			X		
Modelo de despliegue			X		
Modelo de implementación				X	
Modelo de prueba					X

Figura 3.6. Modelos y flujos de trabajo del proceso : desarrollo técnico.

3.4.4.2 Otros Artefactos

Los artefactos del proceso Unificado de Rational se clasifican en artefactos de gestión y artefactos técnicos. Los artefactos técnico pueden dividirse en cuatro conjuntos especiales:

- ***Conjunto de requisitos*** : Agrupa toda la información que describe lo que debe hacer el sistema. Esto puede comprender un modelo de casos de uso, un modelo de requisitos no funcionales, un modelo del dominio, un modelo de análisis y otras formas de expresión de las necesidades del usuario, incluyendo pero no limitándose a maquetas , prototipos de la interfaz, restricciones legales etc.

- ***Conjunto de diseño***: Agrupa información que describe cómo se va a construir el sistema y captura las decisiones acerca de cómo se va a realizar, teniendo en cuenta las restricciones de tiempo, presupuesto, aplicaciones existentes, reutilización, objetivos de calidad y demás consideraciones. Esto puede implicar un modelo de diseño, un modelo de pruebas y otras formas de expresión de la naturaleza del sistema, incluyendo, pero no limitándose a prototipos y arquitecturas ejecutables.

- ***Conjunto de implementación:*** Agrupa toda la información acerca de los elementos software que comprende el sistema, incluyendo, pero no limitándose a código fuente en varios lenguajes de programación, archivos de configuración, archivos de datos, componentes software, etc. junto con la información que describe cómo ensamblar el sistema.

- ***Conjunto de despliegue***: Agrupa toda la información acerca de la forma en que se empaqueta actualmente el software, se distribuye, se instala y se ejecuta en el entorno destino. Ver *figura 3.7.*

Figura 3.7 Roles, actividades y Artefactos.

3.5 Fases e iteraciones en RUP

"Una fase es el intervalo de tiempo entre dos hitos importantes del proceso durante el que se cumple un conjunto bien definido de objetivos, se completan artefactos y se toman decisiones sobre si pasar o no a la siguiente fase". [Booch,et al,1999]

El proceso Unificado de Rational consta de las 4 fases siguiente: *Iniciación, Elaboración, Construcción y Transición.* Las fases de iniciación y elaboración incluyen las actividades de diseño del ciclo de vida del desarrollo- La fase de construcción y transición constituyen su producción.

Durante cada fase hay varias iteraciones . "Una iteración representa un ciclo de desarrollo completo, desde la captura de requisitos en el análisis hasta la implementación y pruebas, que produce como resultado la entrega al cliente ó la salida al mercado de un proyecto ejecutable". [Booch,et al,1999]

Cada iteración pasa a través de varios flujos de trabajo del proceso, aunque con un énfasis diferente en cada uno de ellos dependiendo de la fase en que se encuentre. Durante la iniciación, el interés se orienta hacia el análisis y el diseño. Durante la construcción, el interés se orienta hacia el análisis y el diseño. Durante la construcción, la actividad central es la implementación y la transición se centra en despliegue.

A continuación, en la *figura 3.8* se muestran las fases, iteraciones y flujos de trabajo de RUP.

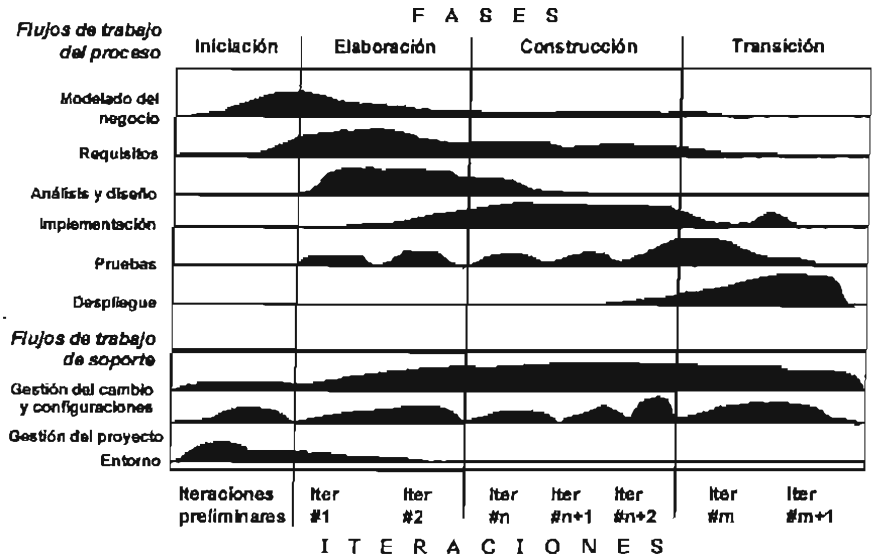


Figura 3.8. Fases e Iteraciones de RUP

El paso a través de las cuatro fases principales constituye un *ciclo de vida del desarrollo* y produce una generación de software. La primera pasada a través de las cuatro fases se denomina *Ciclo de desarrollo inicial*- A menos que acabe la vida del producto, un producto existente evolucionará a la siguiente generación repitiendo la misma secuencia de inicio, elaboración, construcción y transición. Esta es la evolución del sistema, así que los ciclos de desarrollo después del ciclo inicial son los ciclos de evolución.

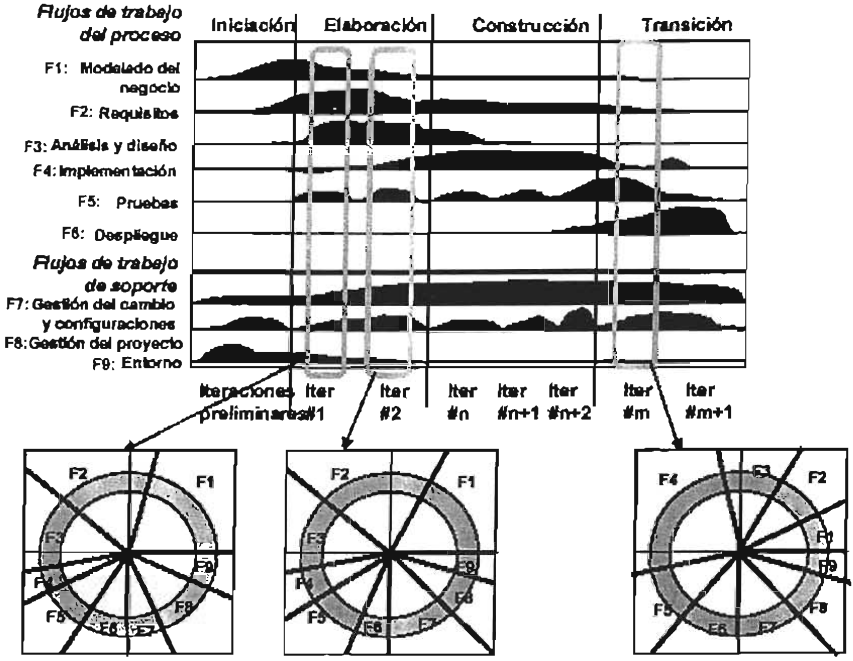


***Figura 3.9.Iteraciones** en cada etapa del ciclo de vida de un sistema.*

Cada fase e iteración se centra en disminuir algún riesgo y concluye con un hito bien definido. La revisión de hitos es el momento adecuado para evaluar cómo se están satisfaciendo los objetivos y si el proyecto necesita ser estructurado de alguna forma para continuar. A continuación se describe cada una de las fases:

3.5.1 Fase de Inicio

Antes de comenzar un proyecto es conveniente plantearse algunas cuestiones :¿ Cúal es el objetivo?, ¿Es factible?,¿ Lo construimos o lo compramos?, ¿Cuánto va a costar?. La fase de Inicio trata de responder a estas preguntas y a otras más. Sin embargo no arroja una estimación precisa ó la captura de todos los requisitos. Más bien se trata de explorar todo el problema lo suficientemente bien para decidir si se va a continuar ó abandonar.

Los *objetivos* de ésta fase son:

- Establecer el ámbito del proyecto y sus límites
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad
-

- Mostrar al menos una arquitectura candidata para los escenarios principales
- Estimar el coste en recursos y tiempo de todo el proyecto
- Estimar los riesgos, las fuentes de incertidumbre

Los *productos* de la fase de inicio deben ser:

- Visión del negocio. Describe los objetivos y restricciones a alto nivel.
- Modelo de casos de uso
- Especificación adicional: requisitos no funcionales
- Glosario: Terminología clave del dominio
- Lista de riesgos y planes de contingencia
- El caso de negocio (*business case*)
- Prototipos exploratorios para probar conceptos ó la arquitectura candidata
- Plan de iteración para la primera iteración de la fase de elaboración
- Plan de fases
- No todos los productos son obligatorios, ni deben completarse al 100% hay que tener en cuenta el objetivo de la fase de inicio

Algunos de los síntomas que nos indican que no se ha entendido la fase de inicio son:

- Dura más de unas pocas semanas
- Se intenta definir todos los requisitos
- Se espera que las estimaciones ó los planes sean muy precisos
- Definir la arquitectura completamente; en lugar de definirla en la fase de elaboración
- No se define el caso de negocio ó la visión
- Los nombres de la mayoría de los casos de uso ó actores no se han definido
- Todos los casos de uso se escriben con detalle

Al terminar la fase de inicio se debe comprobar que los siguientes puntos han sido cubiertos; de ser así será posible continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda
- Entendimiento de los casos de uso principales
- Las estimaciones de tiempo, coste y riesgo son creíbles
- Comprensión total de cualquier prototipo de la arquitectura desarrollada
- Los gastos hasta el momento se asemejan a los planteados

Como se pudo observar, durante la fase de Iniciación se lleva a cabo la planificación del proyecto y se delimita su alcance. La planificación del proyecto incluye los criterios que se

necesitarán y un plan de fase que muestre la planificación de los hitos principales. Durante la iniciación , es frecuente crear un prototipo ejecutable que sirva para probar los conceptos.

Al final de la fase de iniciación se examinan los objetivos del ciclo de vida del proyecto y se decide si proceder con el desarrollo del sistema.

3.5.2 Fase de Elaboración

El propósito de la fase de Elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

Cuando termina ésta fase se llega al punto de no retorno del proyecto ya que a partir de este momento se habrán cubierto las relativamente ligeras y de poco riesgo dos primeras fases para afrontar la fase de construcción que es costosa y arriesgada. Es por ésta razón que la fase de elaboración es de gran importancia.

En ésta fase se construye un prototipo de la arquitectura que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final . Este prototipo debe contener los casas de uso críticos identificados en la fase de inicio. También debe mostrarse que se han evitado los riesgos más graves , bien con éste prototipo, bien con otros de usar y tirar.

Los objetivos de ésta fase son:

- Definir, validad y cimentar la arquitectura
- Completar la visión
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en tiempo razonable

Al terminar deben obtenerse los siguientes productos:

- Un modelo de Casos de uso completa al menos hasta el 80%, todos los casos y actores identificados, la mayoría de os casos desarrollados
- Requisitos adicionales
- Descripción de la arquitectura software
- Un prototipo ejecutable de la arquitectura
- Lista de riesgos y casos de negocio revisados
- Plan de desarrollo del proyecto
- Un caso de desarrollo actualizado que especifica el proceso a seguir
- Posiblemente un manual de usuario preliminar

La forma de aproximarse a ésta fase es tratando de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura ó riesgos importantes.

Los criterios de evaluación de ésta fase son los siguientes:

- La visión del producto es estable
- La arquitectura es estable
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual
- Los gastos hasta ahora son aceptables, comparados con los previstos

Si no se superan los criterios de evaluación quizá será necesario abandonar el proyecto ó replantearlo considerablemente.

3.5.3 Fase de Construcción

La finalidad principal de ésta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante ésta fase todos los componentes, características y requisitos que no hayan sido hechos hasta ahora, han de ser implementados, integrados y testeados obteniéndose una versión del producto que se pueda poner en manos de los usuarios (*prueba beta.*)

El énfasis de ésta fase es que se pueden controlar las operaciones realizadas administrando los recursos eficientemente de tal forma que se optimicen los costes, los calendarios y la calidad.

Los objetivos concretos de ésta fase según incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo ó incluso deshacerlo
- Conseguir una calidad adecuada tan rápido como sea práctico
- Conseguir versiones funcionales (alfa, beta y otras versiones de prueba) tan rápido como sea práctico

Los productos de la fase de construcción deben ser:

- Modelos completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura integra (mantenida y mínimamente actualizada)
- Riesgos presentados mitigados

- Plan del proyecto para la fase de Transición
- Manual inicial de usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio actualizado

Durante la fase de construcción, se desarrolla de forma iterativa e incremental un producto completo que está preparado para la transición hacia la comunidad de usuarios. Esto implica describir los requisitos restantes y los criterios de aceptación, refinando el diseño y completando la implementación y las pruebas del software. Al final de la fase de construcción se decide si el software, los lugares donde se instalará y los usuarios están todos preparados para empezar a funcionar.

3.5.4 Fase de Transición

La finalidad de la fase de Transición es poner el producto en manos de los usuarios finales, para lo que típicamente se requerirá desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto y en general de las tareas relacionadas con el ajuste, configuración, instalación y usabilidad del producto.

En concreto, algunas de las cosas que pueden incluirse en ésta fase:

- Prueba de la versión BETA para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por el proyecto
- Conversión de las bases de datos operacionales
- Entrenamiento de los usuarios y técnicos de mantenimiento
- Traspaso del producto a los equipos de marketing, distribución y venta

Los principales objetivos de ésta fase son:

- Conseguir que el usuarios se valga por sí mismo
- Un producto final que cumpla los requisitos esperados, que funcione y que satisfaga suficientemente al usuario

Los productos que deben obtenerse de la fase de transición son:

- Prototipo Operacional
- Documentos Legales
- Línea de Base del producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la arquitectura completa y corregida

Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión. Las actividades a realizar durante las iteraciones dependerán de su finalidad, si es corregir algún error detectado, normalmente será suficiente con llevar a cabo los flujos de trabajo de implementación y prueba, sin embargo, si se deben añadir nuevas características, la iteración será similar a la de una iteración de la fase de construcción.

La complejidad de ésta fase depende totalmente de la naturaleza del proyecto, de su alcance y de la organización en la que deba implantarse.

De ésta manera se concluye el capítulo 3 en el que se dejan claros los principios de la metodología RUP cuyo enfoque será utilizado posteriormente para mejorar los procesos de prueba de una organización y para tener un referencia de cómo asignar tareas y responsabilidades dentro del equipo de pruebas para poder así administrar y mejorar los procesos con los que se cuenta hoy en día.

Tener en cuenta los siguientes puntos:

- El Proceso Unificado de Rational es un Proceso de ingeniería del software que integra todos los aspectos a tener en cuenta durante todo el ciclo de vida del software y que de igual forma captura las mejores prácticas de desarrollo de software
- Las principales características de dicho proceso son : que es un proceso dirigido por los casos de uso, es un proceso iterativo e incremental, es un proceso centrado en la arquitectura, es un proceso configurable, es un proceso integrado y es un proceso que impulsa un control de calidad y una gestión de riesgo continua
- La arquitectura de la metodología RUP se centra en cuatro componentes que son: los Roles, las Actividades, los flujos de trabajo y los artefactos
- Las fases de RUP son: Iniciación, Elaboración, Construcción y Transición

CAPÍTULO IV

LLEGANDO AL NIVEL 2 DE CMMI (CAPABILITY MATURITY MODEL INTEGRATION) A TRAVÉS DE RUP (RATIONAL UNIFIED PROCESS)

Dado que el modelo de CMMI (*Capability Maturity Model Integration*) indica principalmente "qué" se tiene que hacer pero no especifica el "cómo" ni el "con qué" hacerlo, frecuentemente los lineamientos de CMMI se convierten en algo muy teórico y complejo. Por ésta razón, es necesario adoptar un enfoque práctico y efectivo que haga que los proyectos de software a ser desarrollados tengan éxito en un periodo de tiempo adecuado. Múltiples experiencias a nivel mundial han demostrado que RUP es el camino más apropiado para lograrlo.

El RUP (*Rational Unified Process*) siendo el proceso de desarrollo estándar defacto para desarrollo de software, ya incorpora los elementos y métodos para satisfacer rápidamente el "cómo" y "con qué" que solicita CMMI ya que éste integra las mejores prácticas de desarrollo de software a través de la definición de procesos, flujos de actividades, roles, guías, documentos patrón, ejemplos y métricas.

Por lo tanto, el objetivo particular del presente capítulo es:
"Dar a conocer cómo RUP ayuda a alcanzar el nivel 2 de CMMI dentro de una organización para así obtener un proyecto de software de calidad ".

4.1¿ Porqué transformarnos ?

En la época actual, el éxito de las organizaciones está directamente vinculado con la efectividad del uso de la Tecnología. En este marco, el desarrollo de software es claramente un proceso clave que se verá permanentemente impactado por un lado, por la creciente cantidad de requerimientos y por el otro, por los cambios y variedad de aplicaciones disponibles, ambientes operativos, plataformas, infraestructura de redes, ambientes de desarrollo y lenguajes de programación.

Por lo anterior, se requiere que el Desarrollo y Evolución de Software logre productos de mayor calidad en menor tiempo, respondiendo a requerimientos crecientes, en ambientes tecnológicos cambiantes, con procesos bien definidos, predecibles y en condiciones de ser permanentemente mejorados. Por ésta razón, muchas organizaciones han decidido invertir en mejorar sus procesos de Desarrollo, Evolución y Adquisición de Software para de ésta manera poder ocupar un lugar importante dentro del mercado.

El CMMI ha sido concebido como modelo para determinar y mejorar la capacidad de los procesos en las organizaciones y además ha sido utilizado para identificar fortalezas, debilidades y riesgos asociados durante el proceso de desarrollo de software. Un número creciente de organizaciones que desarrollan, mantienen o proveen servicios de software ya han adoptado las prácticas recomendadas por dicho modelo en sus procesos de producción, lo cual les ha permitido controlar mejor los riesgos inherentes a los proyectos informáticos.

Por su parte, RUP logra enriquecer la productividad y proporciona prácticas óptimas de software a todos los miembros del equipo.

Como punto final, se puede ver que al seguir un esquema como el que presenta RUP, será posible obtener mejoras significativas en plazos razonables y que estarán respaldadas por bases firmes y lecciones aprendidas de las iteraciones anteriores.

4.2. ¿ Cómo lograrlo ?

Repasando un poco, en el Nivel 2 de CMMI (*Nivel Administrado ó Repetible*) se definen las políticas para administrar proyectos de software a su vez que se definen los procedimientos para implementarlas. En éste nivel se establecen procesos básicos de dirección de proyectos para controlar costes, calendarios y funcionalidad de los sistemas.

Un objetivo a cumplir para alcanzar el Nivel 2 es "institucionalizar" el proceso de administración de proyectos de software para que de ésta manera las organizaciones sean capaces de dirigir correctamente sus proyectos y puedan repetir las prácticas que resultaron exitosas de proyectos anteriores.

Por su parte, la metodología RUP integrará todos los aspectos a tener en cuenta para alcanzar el nivel 2 de CMMI y que serán puestos en práctica durante todo el ciclo de vida del desarrollo de software. Proporcionará un enfoque disciplinado sobre cómo asignar tareas y responsabilidades dentro del equipo de desarrollo de sistemas integrando el trabajo de cada uno de sus miembros , tal y como se muestra en la *figura 4.1.*

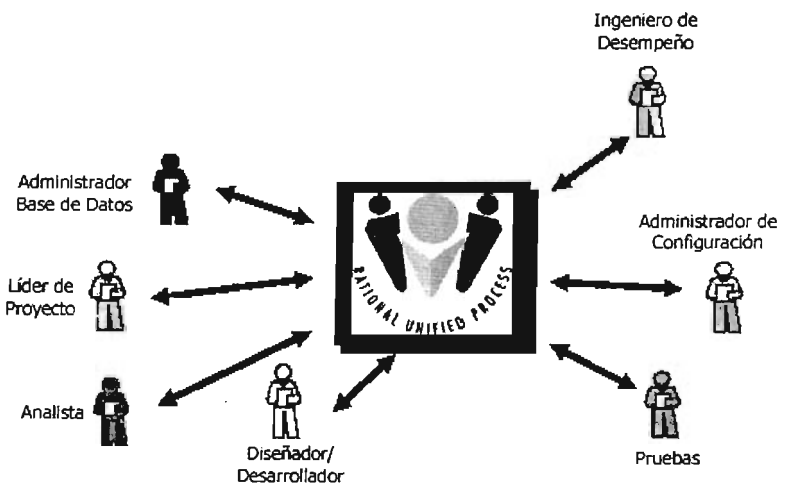


Figura 4.1. Actores en RUP.

A continuación, se explicará como RUP ayuda a las empresas de software a alcanzar el nivel 2 de CMMI atendiendo a cada una de las áreas de Proceso que éste modelo implica y que son:

- *Administración de Requerimientos*
- *Planificación del proyecto de Software*
- *Seguimiento y Control de Proyectos de Software*
- *Administración de Subcontratos de Software*
- *Aseguramiento de calidad de Software*
- *Administración de Configuración de Software*

4.2.1 Administración de Requerimientos

"El propósito de la administración de requerimientos es establecer un entendimiento común entre el equipo de desarrollo de software y las necesidades del cliente. Este entendimiento con el cliente conforma la base para la planificación y administración del proyecto de software." [Kruchten,2003]

El control de las relaciones con el cliente se basa en seguir un efectivo proceso de *control de cambios*. Una de las características claves de RUP es el manejo de *Casos de Uso* los cuales representan una aproximación sistemática a la captura, organización y comunicación de requerimientos del usuario. Estos proveen la manera de documentar requerimientos funcionales que sirven como base para el desarrollo, pruebas y planificación de iteraciones de un proyecto. En RUP , los Casos de Uso son mantenidos en un *Modelo de Caso de Uso* y referenciados consistentemente a través del ciclo de vida del proyecto, desde el análisis hasta las pruebas y mantenimiento.

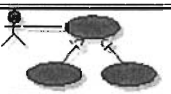
Los *artefactos* que RUP proporciona para capturar los requerimientos del sistema a desarrollar son:

- ***El modelo de Casos de Uso***
- ***Especificaciones del Modelo de Casos de Uso***
- ***Reporte de casos de Uso***
- ***Glosario***

Artefacto: Caso de Uso

Un caso de Uso es una secuencia de transacciones realizadas por el sistema las cuales a su vez, producen un resultado cuantificable de valores para un actor en particular.

Artefacto : Modelo de Casos de Uso



Un Modelo de Casos de Uso es un modelo de funciones deseables para el sistema (casos de uso) y lo que le rodea (actores). El mismo Modelo de Casos de Uso es utilizado en los requerimientos del análisis, diseño y pruebas y lo más importante: resulta ser el medio por el cual se comunica la funcionalidad del sistema al cliente ó usuarios finales.

Los artefactos de RUP que describen los Casos de Uso y escenarios (requerimientos) que serán desarrollados y usados en el contexto de administración son:

- ***Plan de iteraciones***
- ***Plan de Integración de Construcciones***
- ***Plan del Proyecto***
- ***Plan de desarrollo de software***

Artefacto: Plan de Iteraciones

El Plan de Iteraciones es aquel en donde se designan recursos y se detallan las actividades y pruebas a realizarse sobre el sistema.

Todos estos artefactos conforman una línea base y estarán sujetos a una disciplina de Administración de cambios. A continuación, se describirán los objetivos a ser alcanzados dentro de la Administración de Requerimientos y la manera en como RUP los aborda.

Objetivo 1: *Los Requerimientos del sistema deben estar controlados para establecer una línea base en la administración y en el uso de ingeniería de software.*

Con RUP se mantiene un control de configuraciones a medida que los artefactos van evolucionando o sufriendo modificaciones, sin embargo, una línea base formal corresponde a los siguientes hitos:

- Evento Objetivos del Ciclo de Vida (Fase de Concepción)
- Evento Arquitectura del Ciclo de Vida (Fase de Elaboración)
- Evento de Capacidad Inicial de Operación (Fase de Construcción) y
- Evento de Liberación del producto (Fase de transición)

De ésta manera RUP es compatible con CMMI para el acuerdo sobre requerimientos, su administración y seguimiento.

Objetivo 2: *Los Planes de software, productos y actividades deben ser consistentes con los requerimientos del sistema.*

El objetivo de CMMI es asegurar que los sistemas liberados satisfagan los requerimientos del usuario. Con RUP se ayuda a las organizaciones a alcanzar este objetivo de dos maneras:

1. Mediante los *Casos de Uso* que aseguran que los requerimientos del usuario son entendidos y capturados. Una vez que los requerimientos son capturados, estos son representados en diversos modelos visuales RUP (Casos de Uso, Diseño Implementación y Prueba) para asegurar consistencia y adherencia.

2. Mediante un *Desarrollo Iterativo Controlado* que permite disminuir los riegos del proyecto ya que éstos serán entendidos y detectados en forma temprana y serán frecuentemente revisados. Usando metodologías tradicionales en cascada, estos riesgos podrían no ser descubiertos hasta muy tarde dentro del ciclo de vida del proyecto mientras que, una identificación temprana de los riesgos es un beneficio directo para la administración del proyecto pues permite la reevaluación del alcance de los requerimientos.

Los documentos que se utilizan para la administración usando RUP son:

- ***Casos de Negocio***
- ***Plan de Desarrollo de Software***
  - ***-Plan de Mediciones***

-Lista de Riesgos

-Plan del Proyecto

- ***Plan de Iteración***
- ***Evaluación de Iteraciones y Evaluación de Status***

Artefacto : Casos de Negocio

Un Caso de Negocio es aquel que proporciona información sobre las políticas del negocio y que además provee un Plan económico para la realización del proyecto.

Una efectiva administración y Control de Cambios es otra característica de RUP que asegura que el software es desarrollado de acuerdo a los requerimientos que fueron especificados, seguidos y asignados.

Con RUP se aboga por que cada proyecto establezca una Junta de Control de Cambio que pueda arbitrar sobre el alcance e impacto de los cambios propuestos ó defectos no descubiertos durante el curso del desarrollo. Para ayudar a la operación de la Junta de Control de Cambios, RUP recomienda el uso de un medio ambiente y herramientas sólidas en administración de configuraciones y control de versiones.

4.2.2. Planificación de proyectos de software

"El propósito de la Planificación de Proyectos de Software es establecer planes razonables para el desarrollo y para la administración de proyectos. Sin planes realista, no se puede implementar un proyecto efectivo de administración." [Kruchten,2003]

Los objetivos a alcanzar en esta área de proceso son:

Objetivo 1: *Las Estimaciones realizadas sobre los proyectos de software deben estar documentadas para su uso en la planificación y seguimiento de dichos proyectos.*

Uno de los objetivos de RUP, es asegurar que las expectativas de todas las partes involucradas en el proyecto estén sincronizadas y sean consistentes. Esto es asegurado a través de evaluaciones que son realizadas de manera periódica durante el ciclo de vida del proyecto y

cuyos resultados serán documentados en el *Reporte de Evaluación de Status.* Este reporte es utilizado para dar seguimiento a los recursos (humano y financiero) , a los riesgos inherentes del proyecto y al progreso y resultados del desarrollo.

Con RUP se hace uso de las siguientes métricas:

- ***Progreso*** (líneas de código, número de clases, puntos de función por iteración)
- ***Estabilidad*** (volatilidad de requerimientos ó de implementación)
- ***Adaptabilidad*** (costo al hacer cambios de alcance)
- ***Calidad*** (tiempo en la detección de defectos)
- ***Madurez*** (horas de prueba por fallas)

Objetivo 2: *Las actividades del proyecto de software y los compromisos deben ser planificados y documentados.*

Los documentos RUP que contienen los planes y compromisos del proyecto son:

- ***Casos de Negocio***
- ***Plan de Desarrollo de Software***
  - ***-Plan de Medición***
  - ***-Lista de Riesgos***
  - ***-Plan del Proyecto***
- ***Plan(es) de Iteración***
- ***Evaluaciones de Iteración***
- ***Evaluaciones de Status***

4.2.3. Seguimiento y Control del proyecto de software

"El propósito del seguimiento y Control del Proyecto de Software es establecer una visión adecuada del progreso actual, de tal manera que la Administración pueda tomar acciones efectivas cuando el avance del proyecto se desvíe significativamente de los planes originalmente establecidos." [Kruchten,2003]

Objetivo 1: *Los Resultados actuales y avances deben ser medidos a través de los planes de software.*

Como se describe en la sección de Planificación del Proyecto de Software, RUP ofrece varios planes de proyecto así como un Reporte de Evaluación de Status que permiten evaluar el avance actual contra lo que fue planificado. Este reporte , generado para hitos específicos, será responsabilidad del Administrador del Proyecto.

Los principales hitos de RUP corresponden al término de las etapas (Concepción , Elaboración, Construcción y Transición). Por ejemplo: Los objetivos de la etapa de elaboración son analizar el dominio del problema, establecer una arquitectura base que sea segura, desarrollar el plan del proyecto y eliminar los elementos de alto riesgo. Esto implica que la mayoría de los casos de uso deberán estar descritos tomando en cuenta las restricciones que se vayan registrando.

Al final de la etapa de elaboración, los objetivos detallados del sistema y el alcance serán examinados así como también la elección de una arquitectura y la resolución de riesgos mayores. En caso de ser necesario, se tomarán acciones correctivas para evitar que el proyecto se desvíe significativamente de lo planeado.

La ***Lista de Riesgos*** es un artefacto de RUP que nos da una visión de todos los riegos y sirve como entrada para la planificación y evaluación del proyecto. Cada riesgo es descrito en función de su impacto mientras que un plan de contingencia será desarrollado para mitigar el riesgo en cuestión. La Lista de Riesgos es desarrollado junto con los ***Casos de Negocio***, los cuales formarán la base para tomar la decisión de continuar ó no con el proyecto.

Objetivo 2: *Los cambios de alcance en el software deben ser acordados por todo el grupo ó individuos afectados.*

El proceso de desarrollo iterativo controlado como es descrito en RUP, asegura que todos los involucrados en el desarrollo del proyecto de software tendrán una amplia visibilidad sobre el progreso del proyecto.

Los cambios que lleguen a ser propuestos serán revisados por la Junta de Control de Cambios para garantizar que las nuevas propuestas o modificaciones sean realistas y puedan entrar dentro de la programación general del proyecto.

4.2.4. Administración de subcontratos de software

El propósito de la Administración de Subcontratos es el de seleccionar subcontratistas de software calificados y de igual forma administrarlos efectivamente.

Objetivo 1: El contratista principal selecciona subcontratistas de software calificados.
Objetivo 2: El contratista principal y subcontratista de software acuerdan compromisos en ambos sentidos.
Objetivo 3: El contratista principal y el subcontratista de software mantienen comunicaciones continuas.
Objetivo 4: El contratista principal sigue los resultados de los acuerdos y compromisos a los que llegó con el subcontratista.

Estos objetivos van más allá del alcance de RUP pues dependen directamente de la Organización aunque hay que mencionar, que todas las decisiones de subcontratación son documentadas en los **Casos de Negocio**.

Si los subcontratistas siguen el mismo plan de desarrollo que el contratista principal entonces el intercambio de técnicas, hitos principales y evaluación de status serán actividades en común.

4.2.5. Aseguramiento de calidad de software

El propósito del Aseguramiento de Calidad de Software es administrar con una visión adecuada los procesos que están siendo usados para el proyecto de software a su vez de verificar la documentación, cobertura, confiabilidad y facilidad en su mantenimiento. Por supuesto, también incluye el garantizar que el sistema cumpla con las especificaciones y los requerimientos del cliente.

RUP considera "la calidad" como una responsabilidad colectiva de todo el equipo de desarrollo y que debe prevalecer en cada una de las fases del proyecto.

Objetivo 1: *Las actividades de aseguramiento de calidad de software deben ser planificadas.*

La tarea de planificación de Aseguramiento de Calidad de software es una responsabilidad de la Organización; sin embargo, RUP tiene un número de atributos que se

prestan por sí mismos para formar un efectivo programa de Aseguramiento de Calidad de proyectos.

Cada hito en RUP sigue un criterio específico que servirá como base para auditorias. Cada actividad de RUP tendrá asignada una actividad de revisión por separada; y a su vez, cada una de éstas revisiones estará asociada a un conjunto de puntos de chequeo que deberán de ejecutarse antes de iniciar la siguiente actividad.

RUP provee una guía acerca de quien debería revisar determinados artefactos. Por ejemplo, el resultado del "Análisis de Objetos" que fue realizado por un Diseñador, necesita ser revisado por un Desarrollador independiente, Diseñador de Casos de Uso y Revisor de Diseño para así verificar que se esté ajustando a los estándares y guías de desarrollo.

Objetivo 2: *Verificar objetivamente los requerimientos, estándares y procedimientos que estarán asociados al desarrollo del proyecto.*

Este objetivo podría ser resuelto con la adopción de políticas de calidad propias de la organización; sin embargo, RUP provee **templates** de documentos y **checklist** de revisión que pueden ser aplicados como un estándar para los proyectos.

Objetivo 3: *Los grupos afectados por las actividades de aseguramiento de calidad de software deben ser informados de los resultados.*

Como se describe en la Planificación de Proyectos de Software, uno de los objetivos de RUP es asegurar que las expectativas de todas las partes involucradas en el proyecto estén sincronizadas y sean consistentes. Aparte de algunas entradas provenientes de los resultados de auditoria de calidad, RUP genera reportes sobre recursos (humanos y financieros), sobre riesgos mayores, sobre el progreso técnico, etc. para mantener informado a todo el equipo de desarrollo.

Objetivo 4: *Aquellos asuntos que no puedan ser resueltos dentro del proyecto de software deben ser canalizados a la Administración Superior.*

Este objetivo cae fuera del alcance de RUP por lo que será responsabilidad de la propia Organización; sin embargo, el Proceso de Control de Cambio descrito con anterioridad, habilita un mecanismo a través del cual aquellos asuntos que caen fuera del alcance del proyecto ó de lo planeado puedan ser documentados y escalados para su solución.

4.2.6. Administración de Configuración de software

El propósito de la Administración de Configuración de Software es establecer y mantener la integridad del proyecto a través de todo el ciclo de desarrollo. La Administración de Configuraciones de Software es una parte integral de la mayoría de los procesos de administración de ingeniería de software.

Objetivo1: *Las actividades de administración de configuración de software deben ser planificadas.*

Como se describe en RUP, una administración de configuración sólida es un elemento esencial en el método de desarrollo iterativo controlado. Puesto que el software evoluciona en estados, es vital tener disponibles versiones anteriores del software para desarrollos futuros.

RUP tiene dos instrumentos principales para administrar, mantener e integrar las diferentes versiones que sean generadas de un proyecto de software; éstas son:

- ***El plan de administración de Configuración y***
- ***El Plan de Integración de Construcciones***

El plan de Administración de Configuraciones iniciado en la etapa de Concepción describe lo siguiente:

- Se deben Administrar y manejar versiones de software
- Guardando modelos de RUP dados y dividiéndolos en items de configuración;

- Hay que Administrar cambios y releases usando el método de Control de Cambios.

El plan de Integración de Construcciones provee los detalles sobre los items de configuración que serán construidos y el orden en el cual ellos serán integrados en una iteración dada.

Objetivo 2: *Los Productos de software deben estar identificados , controlados y disponibles.*

El plan de Administración de Configuración de RUP trae una descripción de los procesos de configuración, control y administración para asegurar que los productos de trabajo estén efectivamente identificados, controlados y disponibles.

Objetivo 3: *Los cambios realizados al producto de software deben estar estrictamente controlados.*

RUP aboga porque el proyecto mantenga una Junta y un sistema de Control de cambios para que de ésta manera, se manejen adecuadamente costos y se le dé un adecuado seguimiento a la implementación de los nuevos cambios.

Objetivo 4: *Grupos e individuos afectados son informados del estado y contenido del lineamiento base del software.*

RUP aboga por que el lineamiento base de requerimientos, diseño e implementación se mantenga en formato electrónico para que sea dado a conocer a todos los participantes del proyecto. Algún cambio en los lineamientos base será arbitrado por varios niveles del control del proyecto; por ejemplo, cambios a nivel de requerimientos serán considerados por la Junta de Control de Cambios para evaluar su impacto y cambios pequeños en las etapas de diseño e implementación, serán revisados por un nivel apropiado de autoridad técnica. Aprobaciones, niveles de control y la manera en que ellos serán comunicados se describirán en el plan de Administración de Configuración y en el Plan de Desarrollo de Software.

4.3. Factores críticos durante la implantación de CMMI con RUP

Se han identificado ciertos factores críticos durante la implantación de CMMI con RUP debido a que ello implica un "cambio" total en la manera de desarrollar software. Implica cambiar toda una filosofía y adoptar un esquema diferente a lo que se manejaba en la organización.

Durante el proyecto de mejora, se han identificado los siguiente factores críticos a su vez que, se describe la manera de cómo abordarlos. Ver *tabla 4.1*.

Factor crítico de éxito...	**Solucionado mediante...**
Definir las mejores prácticas	La provisión de un marco metodológico basado en RUP que provea los lineamientos para adecuar los procesos de la organización a las mejores prácticas .
Concientizar al personal involucrado	La definición y ejecución de un plan de administración del cambio que asegure que los involucrados en el proyecto de software serán enterados de los objetivos y metas del proyecto.
Capacitar a los involucrados	La definición y ejecución de un plan de capacitación por roles definidos, incluyendo a

	todos los grupos de interés (stakeholders).
Aplicar y seguir las prácticas anteriormente definidas	La definición y ejecución de un sistema de mediciones y reporte de avance que permita evaluar el estado de la aplicación, así como para verificar que se estén cumpliendo los objetivos del proyecto.
Automatizar el proceso	El establecimiento de herramientas integradas de alto desempeño que sustenten totalmente al ciclo de desarrollo de software

Tabla 4.1:Factores críticos en la implementación de CMMI con RUP

De esta manera se concluye el capítulo 4 en donde se deja claro que RUP resulta ser un enfoque claro, concreto y efectivo para la ejecución de las mejores prácticas de CMMI, alcanzando un ambiente integral de desarrollo de software y facilitando la institucionalización y permanencia a largo plazo de las mejoras logradas durante la implantación.

Tener en cuenta los siguientes puntos:

- El *Rational Unified Process* unifica a todo el equipo de desarrollo de software y mejora la comunicación del equipo de trabajo al brindar a cada miembro una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software.
- El *Modelo de Capacidades Integrado*, es un marco de trabajo que describe los elementos claves de un proceso de software eficaz y que describe un camino de mejoramiento evolutivo para pasar de un proceso inmaduro a un proceso maduro.

CAPÍTULO V

APLICACIÓN DE RUP EN LA MEJORA DEL PROCESO DE PRUEBA DE SOFTWARE PARA ALCANZAR EL NIVEL 2 DE CMMI

El presente capítulo tiene la finalidad de evaluar el *Proceso de Prueba de Software* que se sigue dentro del Departamento de Control de Calidad de una empresa dedicada al desarrollo de Sistemas Administrativos, que para fines de éste proyecto y para proteger la confidencialidad de la misma se denominará *"Sistemas Empresariales, S.A de C.V."*

La madurez del Proceso de Prueba del Departamento se evaluará a través de un cuestionario que los consultores del Grupo "ITERA *e-development process*" utilizan para definir el nivel de los procesos de una compañía. De acuerdo a los resultados obtenidos, se determinará el estado actual del proceso para poder establecer las principales acciones a seguir y alcanzar así el Nivel 2 del modelo CMMI (*Capability Maturity Model Integration*). De igual forma, se hará uso de los elementos y métodos que RUP (*Rational Unified Process*) ofrece para poder cumplir los lineamientos que dicho Modelo exige.

Las acciones que se mencionarán, deben ser consideradas como una propuesta de mejora en el Proceso de Prueba de un sistema y que a su vez, han sido identificadas en base a la experiencia que he adquirido en el área durante casi tres años. Con ello, se pretende administrar de una mejor manera las actividades y los recursos con que cuenta el departamento para liberar productos de alta calidad dentro de los tiempos y costos estimados.

El objetivo particular que persigue éste capítulo es :

"Evaluar la madurez del Proceso de Prueba de software de una compañía y proponer estrategias de mejora que permitan alcanzar el Nivel 2 de CMMI utilizando RUP como una herramienta para lograr éste objetivo".

5.1. Etapas del ciclo de desarrollo de sistemas en la empresa "Sistemas Empresariales S.A. de C.V."

Se empezará con definir cada una de las etapas que conforman el Ciclo de Desarrollo de un sistema de la empresa "*Sistemas Empresariales; S.A. de C.V.*" y las cuales se encuentran representadas en la *figura 5.1*.

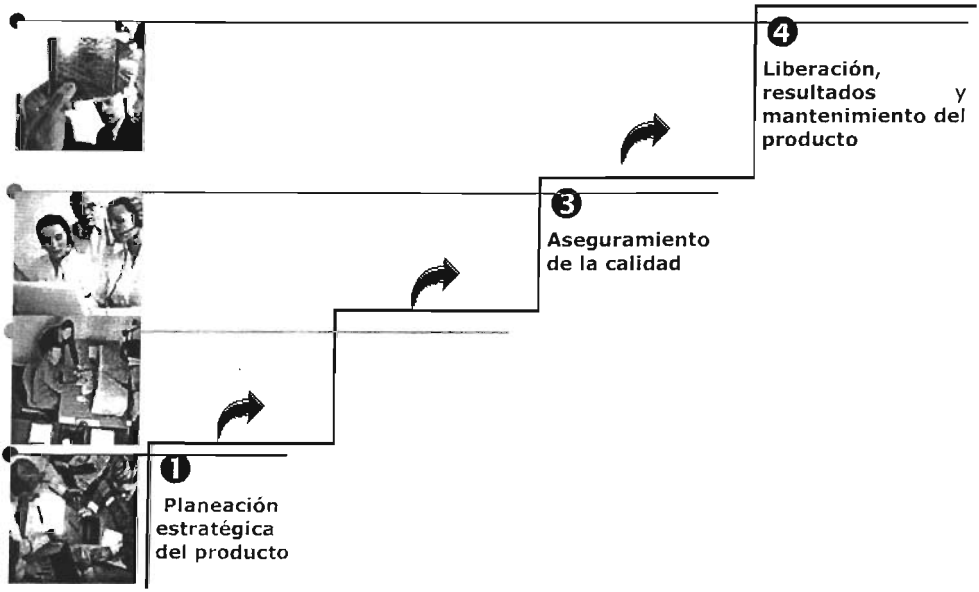


Figura 5.1 : Etapas del Ciclo de Desarrollo de un sistema

5.1.1 Planeación estratégica del producto

Esta etapa se refiere al proceso de reunir información y de establecer los requerimientos ó necesidades del cliente a ser incluidos en el nuevo sistema. Se hace un estudio de factibilidad técnica que consiste en identificar el tipo de tecnología y recursos humanos necesarios para su desarrollo y finalmente, se elabora una Propuesta de desarrollo.

5.1.2 Desarrollo y Programación

En ésta etapa se lleva acabo un diseño lógico y un diseño físico del sistema. En el *diseño lógico,* se describen las especificaciones del nuevo sistema, es decir, aquellas que describen sus características como son: algoritmos, formatos de entradas, la distribución de las salidas a

pantalla, archivos, bases de datos y procedimientos, todo en forma que satisfaga los requerimientos identificados en la primera etapa.

El *diseño físico,* consiste en el desarrollo de programas en un lenguaje de programación que cumplan los algoritmos establecidos, que acepten las entradas proporcionadas por los usuarios, que procesen los datos, que produzcan reportes y que guarden la información en archivos.

5.1.3.Aseguramiento de la calidad

Durante la fase de Pruebas, el sistema se emplea de manera experimental para asegurar que no tenga fallas, es decir, que funcione de acuerdo a las especificaciones y en la forma en que los usuarios esperan que lo haga. En la medida que se disminuyan los defectos y la gravedad de éstos, el sistema estará listo para su liberación.

5.1.4.Liberación, Resultados y Mantenimiento del producto

Esta etapa consiste en lanzar al mercado un sistema de alta calidad llevando estadísticas y un control de fechas de cada versión del sistema liberado. Se llevará acabo el mantenimiento del producto dando solución a cada uno de los errores reportados por los usuarios y, en caso de que las correcciones sean considerables, se analizará la posibilidad de trabajar en un siguiente reinstalable del sistema.

5.2. PA´s del ciclo de desarrollo

En analogía con CMMI, la empresa "*Sistemas Empresariales, S.A DE C.V*" también cuenta con sus propias PA´s (*Process Areas*) para su ciclo de desarrollo de Sistemas y las cuales, describen los pasos necesarios para el diseño, desarrollo y liberación de un producto de software.

En total, ésta empresa cuenta con 10 PA´s distribuidas entre sus 4 etapas de desarrollo tal y como se muestra en la *figura 5.2*.

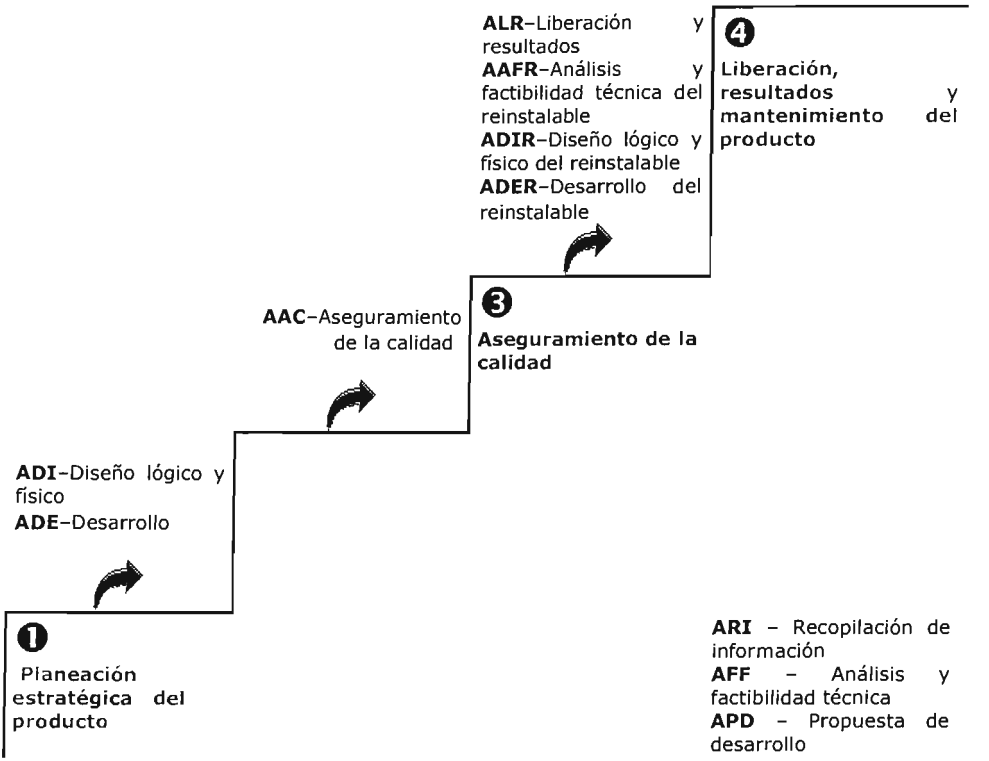


Figura 5.2: PA´s del Ciclo de Desarrollo

Para los fines de éste proyecto, se estudiarán las áreas de Proceso **ACC**(*Aseguramiento de la Calidad*), **ALR** (*Liberación y Resultados*) y **AAFR** *(Análisis y factibilidad técnica del reinstalable)* de las cuales el **Departamento de Control de Calidad** de la empresa es el responsable. En el Departamento de Control de Calidad se lleva a cabo la revisión de los sistemas próximos a liberarse; se verifica su funcionamiento, confiabilidad, facilidad de manejo y por supuesto se valida que el sistema cumpla con las especificaciones y los requerimientos que fueron establecidos en la etapa inicial para garantizar que las necesidades del cliente estarán cubiertas en sus totalidad.

Recordar que cada PA está formada de metas a su vez que cada meta está formada de prácticas. A continuación, se describirán las metas y prácticas que se contemplan dentro de las áreas de Proceso antes mencionadas.

5.2.1 Metas de las PA´S : ACC, ARL, AAFR.

Las metas son aquellas que hacen referencia a los objetivos que deben ser cubiertos en cada una de las áreas de proceso. Las metas a ser alcanzadas en las áreas de Proceso **ACC**(*Aseguramiento de la Calidad*), **ALR** (*Liberación y Resultados*) y **AAFR** (*Análisis y factibilidad técnica del reinstalable)* son descritas en la *tabla 5.1*.

❸ **Aseguramiento de la calidad**	**Metas**
AAC – Aseguramiento de la calidad	- Realizar Pruebas exhaustivas que ayuden a detectar el mayor número de errores en el sistema. - Depurar con la nueva versión los errores encontrados en versiones anteriores del sistema. - Actualizar la lista de errores de la semana. Enviar la lista de errores al líder de Desarrollo del sistema.

❹ **Liberación, resultados y mantenimiento del producto**	**Metas**
ALR – Liberación y resultados	- Llevar un control de todas las versiones liberadas. - Preparar estadísticas generales de la versión. - Hacer entrega del CD de la versión liberada del sistema al Departamento de Producción.
AAFR – Análisis y factibilidad técnica del reinstalable	- Estudiar la lista de errores y sugerencias para determinar cuales pueden desarrollarse para el siguiente reinstalable. - Preparar propuesta de tiempos para la revisión del reinstalable. - Llevar a cabo la revisión del reinstalable.

Tabla 5.1 : Metas de las PA´S : ACC, ARL, AAFR

5.2.2 Prácticas de las PA´S: ACC, ALR Y AAFR

Cada PA cuenta con sus propias Prácticas que consisten en un conjunto de actividades ó grupo de acciones a seguir para alcanzar las metas establecidas. Las prácticas que se siguen dentro del Departamento de Control de calidad para las áreas de Proceso **ACC**(*Aseguramiento de la Calidad*), **ALR** (*Liberación y Resultados*) y **AAFR** (*Análisis y factibilidad técnica del reinstalable)* se describen a continuación en la *tabla 5.2*.

AAC – Aseguramiento de la calidad **Metas**	**Prácticas**
- Realizar Pruebas exhaustivas que ayuden a detectar el mayor número de errores en el sistema.	El líder de Revisión del sistema debe realizar un *Project de Revisión* para cada Proceso ó Módulo a revisar en el sistema. El Project deberá contemplar el mayor número de casos de prueba posibles así como los tiempos asignados a cada tarea. Los probadores deberán apegarse a los tiempos establecidos y preparar las condiciones necesarias para comenzar con la revisión del módulo que les fue asignando como lo es: instalar la última versión del sistema, preparar la base de datos, etc. Cada error encontrado deberá encerrarse a la perfección con la finalidad de contar con una rutina que siempre lo reproduzca. Todos los errores deberán registrarse en una Lista en la que hay que especificar la ubicación del error, su descripción, su Tipo(**D**iseño, **F**uncionamiento, **P**resentación, **A**mbiente), peso

	según la gravedad (1,2,3,4,5), fecha en el que se reporta el error y fecha del instalable con el que se reproduce. En las primeras fases de Revisión deberán realizarse **pruebas de caja negra y caja blanca**. En la última etapa de Revisión se deberán realizar **pruebas de RED, de Volumen, de Concurrencia y de Integración**; así como deberán revisarse el documento Leeme del sistema, el ayuda electrónica y el manual.
- Depurar con la nueva versión los errores encontrados en versiones anteriores del sistema.	Instalar la nueva versión que el líder de Desarrollo del proyecto entrega de manera incondicional cada día Martes a Control de Calidad. El líder de Revisión del sistema imprimirá la lista de errores de cada uno de los probadores. Cada probador deberá verificar que los errores encontrados en versiones anteriores estén corregidos en la nueva versión del sistema. Tanto el líder de Revisión como los probadores deberán reportar en la lista de errores los defectos encontrados en la nueva versión del sistema.

- Actualizar la lista de errores de la semana.	 El líder de Revisión deberá actualizar la lista de errores de la semana con el nuevo estatus: **(N)** error No corregido, **(C)** error Corregido, **(NP)** error que no Procede.  El líder de Revisión junto con el Gerente del departamento determinarán cuales son los errores más importantes por corregir en la semana a fin de asegurar que con ellos se tendrá una mejor versión candidata a liberación.
- Enviar la lista de errores al líder de Desarrollo del sistema.	 Cada día Jueves el líder de Revisión enviará la lista de errores actualizada al líder de Desarrollo del sistema .  El líder de Revisión enviará un correo de Notificación a los involucrados en el proyecto sobre los resultados de la depuración indicando el total de errores reportados y corregidos con la última versión entregada a Control de Calidad.

ALR – Liberación y resultados  **Metas**	**Prácticas**
- Llevar un control de todas las versiones liberadas .	 El líder de Revisión deberá llevar un registro de fechas de liberación del sistema por cada versión, desde la versión original x.xx.0 hasta el último reinstalable que se genere x.xx.n y además deberá anotar el formato de entrega (discos, CD, package).

- Preparar estadísticas generales de la versión.	Una vez que se libera el sistema, el líder de Revisión deberá preparar las estadísticas generales de cada versión donde se presente la distribución de errores, el total de errores reportados , el porcentaje de errores corregidos y el tiempo dedicado a la revisión de la nueva versión del sistema. El líder de Revisión deberá preparar un resumen de los alcances incluidos en la nueva versión así como de aquellos errores que fueron diferidos.
- Hacer entrega del CD de la versión liberada del sistema al Departamento de Producción.	El gerente del Departamento quemará el CD de la versión del sistema a Liberar. El líder de Revisión verificará el "autorun" antes de hacer entrega del CD al departamento de producción. El líder de Revisión del sistema enviará de manera oficial un mail donde se notifique la liberación del producto.

AAFR – Análisis y factibilidad técnica del reinstalable . **Metas**	**Prácticas**
- Estudiar la lista de errores y sugerencias para determinar cuales pueden desarrollarse para el siguiente reinstalable.	El líder de Revisión junto con el Gerente del Departamento estudiarán la lista de errores y sugerencias que quedaron vigentes después de la liberación del sistema; esto, con la finalidad de

	identificar los errores más importantes que deben solucionarse en el reinstalable. Habrá de considerarse que dicha selección asegurará tener en 30 o 60 días una mejor versión para los usuarios.  El líder de Revisión creará una lista única de errores y sugerencias factibles ordenados por prioridad para hacerla llegar al líder de desarrollos.
- Preparar propuesta de tiempos para la revisión del reinstalable.	 El líder de Revisión en base al conocimiento del error o sugerencia deberá estimar el tiempo que será destinado para la revisión del reinstalable
- Llevar a cabo la revisión del reinstalable.	 El líder de Revisión deberá asignar los errores y sugerencias que fueron aprobados para el reinstalable entre el equipo de pruebas, de tal forma que se proceda a su depuración a la llegada de cada versión del reinstalable.  Durante la depuración de cada error, los probadores deben verificar que la corrección del error no haya afectado a otros módulos ó procesos del sistema.

Tabla 5.2 : Prácticas de las PA´S: ACC, ALR Y AAFR

5.3. Evaluación del proceso de prueba

En la sección anterior, se han explicado la áreas de Proceso de las cuales está a cargo el Departamento de Control de Calidad así como de las metas y prácticas que se siguen actualmente pero, ¿el proceso de prueba antes descrito es eficiente y ayuda a cumplir el objetivo de liberar productos de alta calidad en los tiempos y costos estimados?, ¿El proceso actual alcanza el Nivel 2 de CMMI?.

Para responder éstas preguntas a continuación se evaluará el Proceso de Prueba haciendo uso de un cuestionario que el grupo *Itera e - Development process* utiliza para medir la madurez de los procesos dentro de las organizaciones.

PROCESO: ASEGURAR LA CALIDAD DEL PRODUCTO	**SI**	**NO**
Metas Específicas		
¿Se evalúa objetivamente el proceso de Prueba además de tomar en cuenta los recursos humanos y técnicos con que cuenta el departamento para la revisión y ejecución de pruebas ?		X
¿Se da un seguimiento objetivo a los errores del sistema y los problemas de incumplimiento son comunicados asegurando su resolución?	X	
Metas Genéricas		
¿El proceso está institucionalizado en la organización y es un proceso dirigido?		X
¿Existen políticas dentro de la organización que deben mantenerse durante la planeación y ejecución del proceso de Pruebas para así asegurar la calidad del producto?.	X	
¿Se tiene establecido y se mantiene un plan de ejecución de Pruebas para asegurar la Calidad del Producto?.	X	
¿Se proporcionan recursos para la ejecución del proceso de Pruebas sobre el sistema?	X	
¿Se tienen asignadas de manera efectiva las responsabilidades para la ejecución de Pruebas del sistema ?		X
¿Se tienen niveles apropiados de administración para controlar el proceso de Prueba de los sistemas?		X
¿Se tienen identificados e involucrados a los principales interesados ó Stakeholders para asegurar la Calidad del producto?	X	
¿Se lleva un monitoreo y control del proceso de Prueba?		X
¿Se hacen revisiones del proceso de Prueba con la Alta Dirección a fin de mejorar el proceso actual?		X

Los resultados de la evaluación anterior, hacen evidente que el Proceso vigente dentro del departamento no alcanza el Nivel 2 de CMMI pues ello implicaría haber contestado afirmativamente todas las preguntas del cuestionario.

Básicamente se puede observar que el problema radica en que el proceso de Prueba no es un Proceso Institucionalizado ni Controlado , que la administración y asignación de responsabilidades no es del todo adecuada y que además no existe una cultura de mejora de procesos.

Se puede decir que los principales problemas con que se enfrenta el Departamento tras la liberación de un nuevo sistema ó versión son los siguientes:

1) El trabajo ó las responsabilidades están concentradas en una sola persona, para éste caso, hacia el Gerente del Departamento pues aunque existe un líder de Revisión por cada sistema que desarrolla la empresa, el Gerente siempre decide participar en la mayoría de las pruebas y en la toma de decisiones; no sólo para dar el visto bueno ó para analizar los resultados sino que generalmente se remota a las pruebas desde su inicio. Lo anterior, es un grave problema ya que el proceso de Pruebas no es sólo hacia un sistema sino que en paralelo se revisan tres en promedio; no se delegan responsabilidades y ello ocasiona que el cuello de botella sea el Gerente del Departamento provocando retrasos considerables.

2) No hay una asignación adecuada de las actividades y del personal para cada proyecto . Se da el caso en que algunas personas del equipo de pruebas llegan a estar involucrados en la revisión de varios de los sistemas lo cual provoca que su carga de trabajo sea mayor y que a su vez lleven retrasos en cada una de las tareas que le han sido asignadas. En ocasiones, también el número de personas involucradas en el equipo de pruebas resulta ser deficiente pues no cubre la dimensión del proyecto ni ayuda para que los tiempos establecidos sean respetados.

3) Del mismo modo, no hay una asignación correcta de tiempos para cada actividad, en ocasiones es mayor pero por lo general es menor. El problema aquí es que la liberación del sistema se estima de acuerdo a los tiempos que han sido establecidos (tanto de desarrollo como de pruebas) que al no ser realistas provocan que la liberación de los productos se retrase. Cuando esto sucede, comienzan las presiones Comerciales ya que la cuestión de mercadotecnia (anuncio del lanzamiento del producto), la difusión de cursos y capacitación a usuarios y distribuidores e.t.c se pone en marcha aún y cuando no se tiene listo el sistema para su lanzamiento al mercado.

4) No se tiene una cultura de mejora de procesos dentro del departamento pues el Proceso seguido liberación tras liberación es exactamente el mismo. Si se cometieron errores tras la liberación de un sistema, éstos mismos se arrastrarán en proyectos posteriores. No se logra aprender de los errores y la actitud es de un "NO" al cambio.

5.4 Propuesta de solución utilizando RUP

Una vez que se han identificado los problema del Departamento, se procederá a proponer una solución a cada uno de ellos valiéndose de los fundamentos de la metodología RUP. Se optó por utilizar *Rational Unified Process* por las siguientes razones:

1) RUP es explícito en la definición de artefactos y su trazabilidad, es decir, contempla la relación causal de los artefactos creados desde los requerimientos hasta la implementación y pruebas de los sistemas.

2) RUP ayuda a identificar claramente a los actores involucrados en el proceso de Pruebas de software, sus responsabilidades y su flujo de trabajo en cada una de las actividades. Además, explícitamente indica qué actor es responsable de qué artefacto en cada actividad.

A continuación, dentro del Proceso de Prueba que se sigue en el Departamento de Control de Calidad , se identificarán cada uno de los 4 componentes que la Metodología RUP requiere sean identificados y los cuales son:

1)Roles (*workers*)
2)Actividades(*activities*)
3)Flujos de Trabajo (*workflows*) y
4)Productos (*artefactos*).

5.4.1 Identificando y definiendo nuevos Roles

En el proceso de Aseguramiento de Calidad se tienen los siguientes roles:

1)Gerente: Persona responsable de Administrar los recursos del departamento para la puesta en marcha del proceso de Prueba en el(los) sistema(s) próximo(s) a liberarse. Es quién de manera oficial, informa a la Alta Dirección sobre la liberación de un producto. De igual forma, participa en la elaboración de Casos de Prueba y en la Revisión de los sistemas.

2) Líder de Revisión: Persona responsable de evaluar el estado de la nueva versión de un sistema con la finalidad de tener una visión de la magnitud de la revisión. Se encarga de dar seguimiento al plan de trabajo establecido de manera conjunta con la Gerencia así como de proporcionar toda la información que se deriva de las Pruebas al área de Sistemas. El Líder de Revisión es también responsable de identificar y de definir los requerimientos de cada Caso de Prueba, lleva acabo el monitoreo del progreso y de los resultados que arroje la revisión del sistema; igualmente participa en el proceso de Revisión.

3) Probadores (testers): Son las personas responsables de llevar acabo las diferentes pruebas que les fueron asignadas en su Project de revisión apegándose lo más posible a los tiempos definidos. Son quienes describen los errores detectados, los clasifican y a la llegada de una nueva versión del sistema son quienes efectúan la depuración de los mismos.

La situación actual del Departamento, hace evidente que los roles antes mencionados son pocos en comparación a la cantidad de actividades que se realizan durante la revisión de un sistema; provocando, que muchas de las actividades se concentren en una sola persona y que se generen cuellos de botella; ó bien, también se llega al punto en el que todos los integrantes del equipo de Pruebas "hacen de todo" perdiéndose hasta cierto punto las responsabilidades de cada cual.

Propuesta de mejora

Valiéndose de RUP, a continuación se propone que dentro del departamento se dé lugar a la creación de 2 roles adicionales con el fin de agilizar y controlar el Proceso de Revisión; éstos roles son: Rol de Diseñador de Pruebas y el Rol de Probador-Beta.

Diseñador de Pruebas: Será el encargado de preparar las condiciones necesarias para la ejecución de pruebas sobre un sistema; es decir, estará a cargo de las configuraciones de Red, del diseño de la base de datos a utilizarse en las pruebas, para el caso del trabajo con Bases de datos remotas (*Oracle, Sql y Db2*) será quién configure el cliente y el servidor de tales manejadores, será quién se documente en las últimas tecnologías (*Terminal Server, nuevas versiones de manejadores de bases de datos*) y en la Administración de los diferentes sistemas operativos (*WinNT, WinXP, Win2000, etc.*) para ser considerados dentro de las pruebas. Tener conocimientos sobre Configuraciones resulta indispensable ya que es importante mencionar que no todos los errores detectados son de sistema sino que tienen que ver con ésta cuestión. Finalmente, dentro de sus actividades estará el llevar una documentación de los aspectos antes mencionados.

Probador Beta: El probador Beta no será propiamente una persona del Departamento sino que más bien será un Usuario Final (*cliente ó distribuidor*). Se intenta que una persona externa al equipo de desarrollo y de pruebas comience a trabajar con la nueva versión del sistema (*versión Beta*) antes que todos los demás usuarios. El sistema que será distribuido a los probadores Beta será una versión que casi en su totalidad estará desarrollada y probada. La intención de contar con probadores beta será el de verificar que efectivamente la versión a liberarse es confiable , funcional, amigable y que ciertamente se apega a las necesidades del cliente. Del mismo modo, distribuir versiones Betas servirá de apoyo al Departamento de Control de Calidad en la detección de nuevos errores; los cuales, serán identificados bajo condiciones y datos reales.

En Resumen, los Roles del Departamento quedarían tal y como se muestra en *la figura 5.3.*

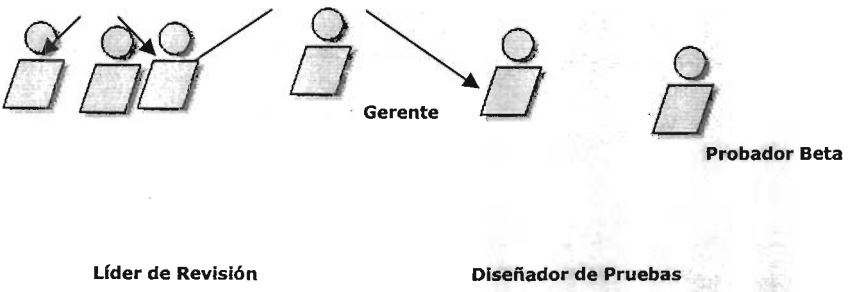


Figura 5.3: Roles del Departamento

5.4.2 Identificando y re-definiendo Actividades

Como se ha visto en capítulos anteriores, RUP como una metodología para el Proceso de desarrollo de Software permite asignar de manera adecuada las tareas y responsabilidades de acuerdo al rol de cada persona. A continuación, serán descritas las actividades que se desempeñan dentro del equipo de Pruebas y la forma en como éstas son distribuidas, que cómo se ha estudiado , la asignación de las mismas no es del todo correcta. La propuesta de mejora para éste punto irá enfocada principalmente a la re-asignación de tareas y responsabilidades.

ACTIVIDAD 1: Determinar los resultados de las pruebas
Roles: Gerente y Líder de Revisión

Actualmente, los responsables de determinar los resultados de las Pruebas son el Gerente del Departamento en colaboración con el Líder de Revisión. En base a los resultados obtenidos, ambos son capaces de evaluar el estado del producto, identifican los detalles de cada Prueba y conocen los defectos contenidos en cada módulo del sistema .

Problema: El problema en la asignación de ésta tarea es que, el Gerente aunque participa en algunas de las pruebas su visión no resulta ser lo suficientemente amplia acerca de las características de las mismas y mucho menos tiene un juicio exacto sobre el estado del sistema en liberación. Al final, el Gerente logra tener una visión general de la situación y en los casos más críticos ó graves logra tener el dominio. Lo anterior sin embargo, habrá significado tiempo y esfuerzo que bien podrían haber sido empleados en otras actividades.

Propuesta de mejora
El líder de Revisión debería ser la persona encargada de Evaluar los resultados de cada prueba realizada sobre el sistema y posteriormente, llevar su evaluación con el Gerente para discutir el estado actual del sistema y de ésta manera tomar decisiones que definan el rumbo del proyecto.

ACTIVIDAD 2: Definir el procedimiento de prueba
Roles: Gerente y Líder de Revisión

En ésta actividad se intenta definir el Plan de Revisión y los Procedimientos que se emplearán para la ejecución de las pruebas.

Problema: Nuevamente nos encontramos que tanto el Gerente como el Líder de Revisión comparten ésta responsabilidad; sin embargo, es importante mencionar que el Líder de Revisión por ser quién tiene dominio sobre el sistema es la persona ideal para realizar ésta actividad. Actualmente, el Gerente y el Líder de Revisión diseñan y discuten el Plan de principio a fin y aunque esto resulta benéfico pues contendrá el punto de vista de dos miembros del equipo, sin duda implicará ocupar el tiempo de dos personas para exactamente la misma actividad sobre un mismo sistema.

Propuesta de mejora
El líder de Revisión debería ser el encargado de diseñar el Plan de Pruebas para que después éste presente su propuesta al Gerente del Departamento. El Gerente por su parte, sólo tendría que evaluar el Plan propuesto y en caso de tener observaciones y sugerencias al respecto solicitaría las modificaciones.

ACTIVIDAD 3: Ejecución de las pruebas

Roles: Gerente, Líder de Revisión y Probadores

Esta actividad consiste en llevar acabo todas las pruebas que fueron establecidas en el Plan de Revisión. Se habrán de abarcar pruebas de Caja Negra, pruebas de Concurrencia, Pruebas de Volumen y Pruebas de Red.

Problema: Sin duda alguna, todo el Departamento debe estar involucrado en la Revisión de la nueva versión del sistema, el único problema detectado en ésta actividad es que en ocasiones el Gerente está concentrado en otras actividades por lo que al querer participar en las pruebas éstas se retrasan.

Propuesta de mejora
El Gerente al no tener la posibilidad de participar en las pruebas por falta de tiempo ó porque surgió un imprevisto, éste deberá delegar la responsabilidad de la Revisión a la persona que está a cargo y posteriormente que ésta misma persona sea quién le dé a conocer los resultados. En caso de que el Gerente requiera ver el transcurso de las pruebas se podría repetir un caso ó incluir uno nuevo.

Resumiendo, las actividades asociadas a cada uno de los roles del Departamento quedarían distribuidas tal y como se muestran en la *figura 5.4*.

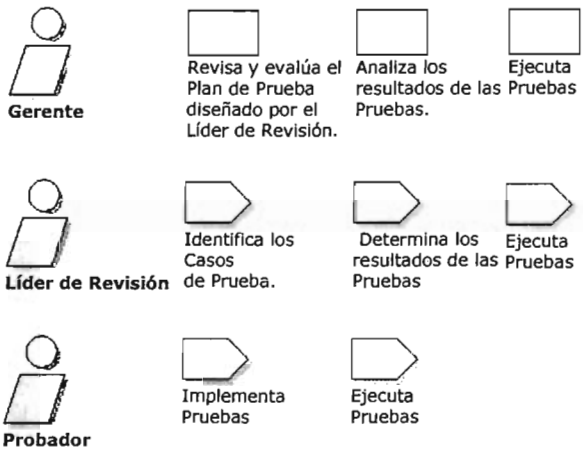


Figura 5.4: Roles y Actividades del Departamento.

5.4.3 IDENTIFICANDO Y DEFINIENDO NUEVOS FLUJOS DE TRABAJO

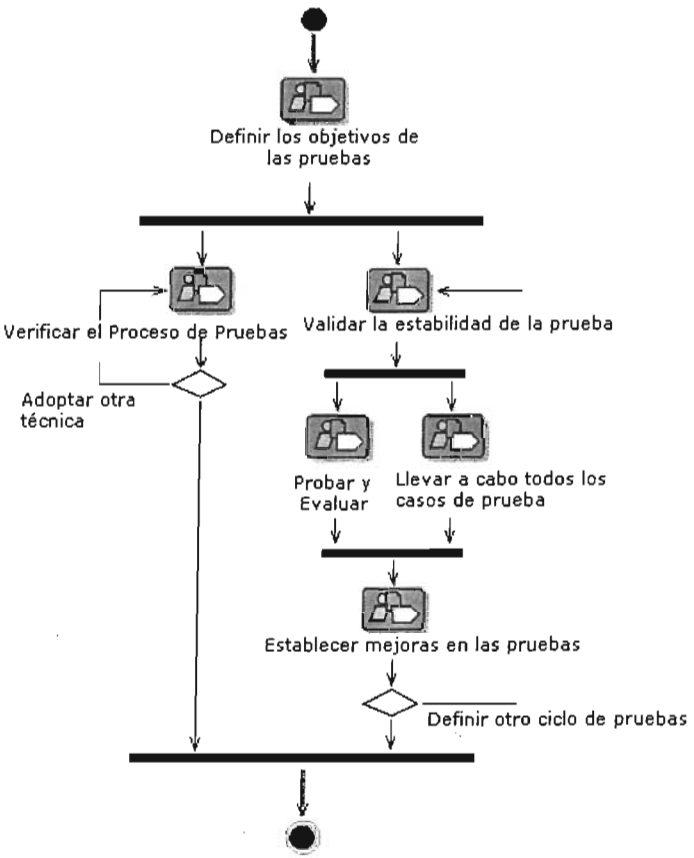
Los flujos de trabajo son los que definen la secuencia de las actividades involucradas en el proceso de Prueba de un sistema. El principal problema detectado dentro del departamento es que no siempre se valida que las pruebas contemplen el mayor número de casos posibles, también hay ocasiones en que las pruebas se salen de control pues sobre la marcha y de manera improvisada se involucran nuevas variables ó condiciones de tal forma que al momento de reproducir un error, no resultará sencillo determinar que fue lo que lo ocasionó.

Otro mal que se padece es que se empieza por preparar las condiciones para llevar a cabo las pruebas sin antes tener conocimiento de la estabilidad del módulo ó función del sistema que se revisará; con ello, se corre el riesgo de que las pruebas se vean interrumpidas ó en el peor de los casos que ni siquiera puedan comenzarse.

Propuesta de mejora

Dentro del flujo de trabajo del departamento se propone el poder Verificar el Proceso de Pruebas así como Validar la estabilidad de la misma antes que preparar las condiciones para la puesta en marcha de la Revisión y poner al equipo de Pruebas a trabajar.

El flujo de trabajo propuesto queda representado en la *figura 5.5.*



5.4.4 Identificando y definiendo nuevos Artefactos

Como último componente de la metodología RUP encontramos a los artefactos que son aquellos elementos que se producen durante y después del proceso de prueba. Cada una de las actividades que fueron anteriormente mencionadas llevan algunos artefactos asociados y algunos de los cuales serán requeridos como entradas mientras que otros serán generados como salidas.

Dentro del Departamento se cuenta con 2 principales artefactos y que son : el Project de Revisión y la Lista de errores del sistema.

1) Project de Revisión: Es un Documento elaborado en Microsoft Project y en el cual se especifican cada uno de los módulos a revisar en el sistema, los casos de Prueba a tener en cuenta durante la revisión y en donde además se asignan los tiempos de cada actividad.

2) Lista de Errores: El Departamento cuenta con una Lista de errores elaborada en Excel y mediante la cual se lleva el registro y control de los errores detectados por las Pruebas y Revisiones del sistema. En dicha lista se especifica la ubicación del error, se describe la rutina para reproducirlo, se le asigna un peso que va desde el *peso 1* al *peso 5* dependiendo de la gravedad e impacto que tenga sobre el sistema, se clasifica de acuerdo a su tipo (*error de Diseño, error de Presentación, Error de Funcionamiento*) , se define la fecha y hora del ejecutable con el cual se reproduce el error y además se registra el nombre de la persona que lo reportó.

Tras la llegada de una nueva versión del sistema está lista es depurada; es decir, el equipo de pruebas trata de reproducir sus errores con el nuevo ejecutable y en caso de volverse a suscitar el estatus del error dentro de la lista queda como (**N**)o Corregido y en caso contrario el estatus asignado será (**C**)orregido. De igual forma , está lista será alimentada con los nuevos errores que sean detectados durante la depuración.

Finalmente, éste artefacto es generado como una salida en la que se reflejarán los resultados del proceso de Revisión y que además será el medio por el cual el área de Desarrollos se dará cuenta de los errores y del estado del sistema.

Los artefactos con que actualmente cuenta el departamento resultan ser efectivos pero no suficientes, a continuación se propondrá la generación de algunos otros para apoyar el trabajo del equipo de Pruebas.

Propuesta de mejora

Valiéndose de los fundamentos de la metodología RUP se propone considerar nuevos artefactos como lo son : archivos .logs y scripts de pruebas.

Archivos Logs: Los archivos *logs* son aquellos que registran una bitácora de eventos y los resultados de su ejecución; de ésta manera cada uno de los probadores podrá encerrar de una mejor manera cada uno de sus errores y determinar con exactitud el proceso que está generando el conflicto.

Scripts de pruebas: Los scripts son programas que definen una secuencia dentro del sistema y los cuales pueden ser programados para que se realicen de manera iterativa. Por ejemplo, cuando se requieren hacer pruebas de Volumen, el encargado de preparar la base de datos se ve en la necesidad de dar de alta una infinidad de registros lo cual resulta tardado y engorroso. La generación de un script, permitiría hacer este proceso de manera automática y se tendría la posibilidad de dejar corriendo dicho programa en una estación de trabajo mientras que se preparan otras condiciones para realizar la prueba.

Resumiendo, los artefactos que serán generados por las actividades de cada rol del departamento quedarían tal y como se muestra en la *figura 5.6*.

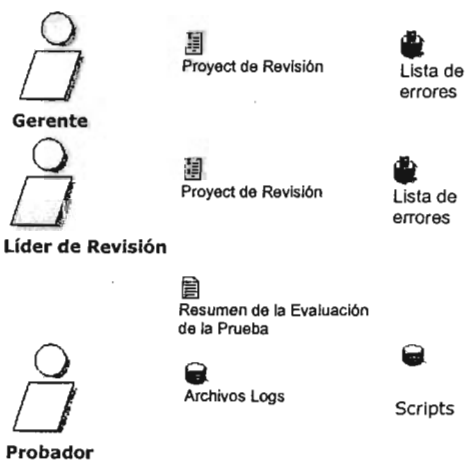


Figura 5.6: Artefactos del Departamento.

ANEXOS

Requisitos del Sistema de Calidad

ISO 9001

La Norma internacional ISO-9001 especifica los requisitos que debe cumplir un sistema de Calidad cuando es necesario demostrar la capacidad de un proveedor para diseñar y suministrar productos conformes. La lista de esos 20 requisitos o direcciones evaluadas es:

1) Responsabilidad de la dirección

- Política de Calidad
- Organización
- Responsabilidad y autoridad
- Recursos
- Representante de la Dirección
- Revisión por la Dirección

2) Sistema de Calidad

- Generalidades
- Procedimientos
- Planificación de calidad

3) Revisión de contratos

- Revisión
- Modificación de contratos
- Registros

4) Control de diseño

- Planificación del diseño y del desarrollo
- Interfaces organizativas y técnicas
- Elementos de entrada
- Elementos de salida
- Revisión del diseño

- Verificación del diseño
- Validación del diseño
- Control de cambios

5) Control de documentación y de los datos
- Aprobación y distribución de documentos y de datos
- Cambios en documentos y datos

6) Compras
- Evaluación de subcontratistas
- Datos sobre las compras
- Verificación de los productos comprados
- Verificación en la casa del proveedor realizadas por el comprador
- Verificación realizada por el cliente sobre los productos que subcontrata el proveedor

7) Control sobre los productos que suministramos

8) Identificación y trazabilidad del producto

9) Control de procesos

10) Inspección y ensayos
- En la recepción
- En el proceso
- Finales
- Registros de inspección y ensayos

11) Control de equipos de inspección, medición y ensayos
- Procedimientos de control

12) Estado de inspección y ensayo

13) Control de productos no-conformes
- Revisión y tratamiento

14) Acciones correctivas y preventivas

15) Manipulación, almacenamiento, embalaje, preservación y entrega

16) Control de registros de calidad

17) Auditorías internas

18) Entrenamiento

19) Servicio posventa

20) Técnicas estadísticas

- Identificación de la necesidad
- Procedimientos

ISO/IEC TR 15504

El estándar ISO/IEC 15504 proporciona un marco para todos los aspectos de una evaluación de proceso que se puede utilizar para evaluar la capacidad de los procesos de una organización.

Define el modelo de referencia de procesos de software y de capacidades de procesos que constituyen la base para la evaluación de procesos de software. Se compone de 9 partes de las cuales la 2, 3 y 9 son normativas y las demás informativas.

Ventajas

- Específico para el desarrollo y mantenimiento de software.
- Fácil de entender (24 procesos, 16 páginas).
- Definido como un conjunto de procesos.
- Orientado a mejorar los procesos para contribuir a los objetivos del negocio.

Desventajas

- No es práctico ni fácil de aplicar.
- Tiene solamente lineamientos para un mecanismo de evaluación.
- Todavía no es norma internacional.

CONCLUSIONES

Con el desarrollo de éste proyecto se alcanzó el objetivo de presentar una iniciativa encaminada a la mejora del Proceso de Prueba dentro del Departamento de Control de Calidad de la Empresa "*Sistemas Empresariales, S.A. de C.V.*"

Dicha propuesta ha sido sustentada y justificada ante la Gerencia para que en un futuro ésta misma sea expuesta a la Dirección y Alta Gerencia y obtener, así los recursos e infraestructura necesarios para la puesta en marcha de éste programa de mejoramiento.

Se ha concientizado al Departamento sobre la necesidad de adoptar una cultura de procesos por lo que parte de la solución ha sido cambiar la mentalidad del equipo de pruebas partiendo desde luego por el Gerente del Departamento. Parte de la solución también tiene que ver con educar adecuadamente a los involucrados en el proceso de desarrollo de modo que algunas personas del equipo han asistido a cursos y diplomados sobre CMMI.

El año 2005 es un año que promete mucho para ésta empresa; habrá muchas innovaciones que van desde el cambio de imagen, liberación de productos con nuevos alcances y mejor apariencia, nuevo esquema de seguridad en los sistemas a fin de combatir la piratería, se implementará una nueva política de licenciamiento y bien éste año podría ser el comienzo para tomar el rumbo hacia una cultura de procesos.

Sin duda, es largo el camino que hay que recorrer y muchas las áreas que deben adoptar éste mismo esquema; sin embargo, vivir los problemas y las pérdidas por contar con procesos inmaduros motivan a cambiar la forma en como se desarrolla y prueba el software.

REFERENCIAS BIBLIOGRÁFICAS

[Watts ,1989] WATTS,Humphrey S. Managing the software process. Ed. Addison Wesley Iberoamericana. Estados Unidos 1989.

[Paulk,1985] PAULK, Mark ; Weber,C; Curtis, B. Best practices for softwre development teams. Ed. Addison Wesley. Estados Unidos 1985.

[Brown,1996] BROWN, Alan W. Component Based software engineering, IEEE Computer society,1996.

[Perks,2003] PERKS, Mike. Best Practices for software development projects. Ed Addison Wesley. Estados Unidos 2003.

[Royce,2002] ROYCE, Walker . CMM VS CMMI , Addison Wesley Longman. Estados Unidos 2002.

[Morales,2004] MORALES, Adriana Cueto. Diplomado en Calidad de software. AMSIS 2004.

[Kruchten,2001] KRUCHTEN, Philippe. The Rational Unified Process, an Introduction. Ed. Addison Wesley. Estados Unidos 2001.

[Ollman,et al,2002] OLLMAN, Roger; P, Leslee; E,María. Applying Requireents manageent with use cases. Ed. Addison Wesley. Estados Unidos 2002.

[Booch,et al,1999] BOOCH G; Rumbaugh J; Jacobson I. El lenguaje Unificado de Modelado; Addison- Wesley; Madrid; 1999.

[Kruchten,2003] KRUCHTEN, Phillipe. Reaching CMMI Level 2 with the Rational Unified Process. Addison Wesley Iberoamericana. Estados Unidos 2003.

BIBLIOGRAFÍA

PRESSMAN, Roger S. Ingeniería del Software. Un enfoque práctico; ed.Mac Graw Hill;Cuarta edición; 2001.

BOOCH, G; Jacobson, I; Rumbaugh J. The Unified Software Development Process. First Edition. Addison Wesley.1999.

BOOCH, Grady.UML User´s Guide. Second Edition. Addison Wesley 2001.

KUNTZMANN, Annie Combelles and Philippe Kruchten. The Rational Unifed Process – An Enabler for higher Process Maturity. First Edition; Addison Wesley 2002.

PULFORD, K; A. Kuntzmann-Combelles and S.Shirlaw. A quantitative Approach to Software Management. Addison Wesley Longman;2001.

W, Barry; Boehm. Anchoring the Software Process. IEEE Software; 1996.

MCFEELEY, Robert. IDEAL: A User's Guide for Software Process Improvement. Software Engineering Institute, Pittsburg. 1998.

MCCONNEL, Steve I. Software Project Survival Guide. Redmond Microsoft Press;1997.

PAULK, Mark. Capability Maturity Model for Software, Version 1.1. Software Engineering Institute, Pittsburg;1998.

ROYCE, Walker. Software Project Management: A Unified Framework. Addison Wesley Longman;1998.

White papers

RUP implementation guide
Part 1:Recommended strategy and typical issues and risk by the RUP implementartion Workshop Group.

RUP implementation guide
Part 2:Best practices and key lessons learned from RUP implementation projects by the RUP implementartion Workshop Group.

What is the Rational Unnified Process?
Philippe Krutchten. Rational Fellow Rational Software Canada.2002

The Ideal Model: A practical guide for improvement.
Jennifer Gremba and Chuck Myers. Carnegie Mellon Software Engineering Institute.2001.

Reaching CMM Levels 2 and 3 with the Rational Unified Process.
Jas Madhur.Rational Software;1998.

SPICE, the Theory & Practice of Software Process Improvement.
IEEE Computer Society;1999.

Managing the software Process .Watts S. Humphrey.; 1989.

Best Practices for software development Teams. Paulk, M.C; Weber, C.V; Curtis,B y Crisis,M.B.; 1985.

Component -Based software Engineering . Alan W.Brown. IEEE Computer Society; 1996.

Best practices for Software development Projects . Mike Perks; 2003.

CMM vs CMMI: From Conventional to Modern Software Management.
Walker Royce. Ratonal Software Corporation.2000.

Referencias Electrónicas:

www.itera.com.mx. 2005
www.rational.com. 2005
www.sei.cmu.edu. 2005