INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA,
COMPUTACIÓN, INFORMACIÓNY COMUNICACIONES

ANALISIS DE FACTIBILIDAD PARA LA APLICACION DE LA ADMINISTRACION MODERNA DE PROYECTOS DE DESARROLLO DE SOFTWARE, BASADO EN LOS PRINCIPIOS DE LA MODELACION ORIENTADA A OBJETOS

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRO EN ADMINISTRACION DE TECNOLOGIAS DE INFORMACION

POR:

MARCO ANTONIO VALDEZ ACEVES

MONTERREY, N.L. ABRIL 2004

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

DIVISIÓN DE ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

PROGRAMAS DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

Los miembros del comité de tesis recomendamos que la presente tesis del Lic. Marco Antonio Valdez Aceves sea aceptada como requisito parcial para obtener el grado académico de Maestro en Administración de Tecnologías de Información.

Comité de tesis:

_____

Dolores Guadalupe Lankenau Caballero, MA., MSI

Asesor

_____

Bertha Laura García de la Paz, MA., MTI

Sinodal

_____

Ing Cesar Centeno Arriaga, MTI
Sinodal

_____

David Alejandro Garza Salazar, PhD.
Director del Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.

Abril de 2004

ANALISIS DE FACTIBILIDAD PARA LA APLICACIÓN DE LA ADMISNITRACION MODERNA DE PRO YECTOS DE DESARROLLO DE SOFTWARE, BASADO EN LOS PRINCIPIOS DE LA MODELACION ORIENTADA A OBJETOS

POR:

MARCO ANTONIO VALDEZ ACEVES

TESIS

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.

Este trabajo es requisito parcial para obtener el grado de Maestro
en Administración de Tecnologías de Información

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY

ABRIL 2004

Dedicatoria

A Dios por guiarme en el camino de la verdad y del bien.

A mis padres, Maria del Carmen y José Leonardo, por darme la vida y e impulsar los proyectos emprendidos a lo largo de ella.

Al amor de mi vida, Claudia, por ser mi inspiración.

A mis hermanos Jacqueline, Noemí, Mónica y Leonardo por alentarme siempre.

A mis abuelos, Tirso, Juanita y Fernando que a pesar que ya no están conmigo, siempre estarán en mi corazón, y muy especialmente a mi abuela Porfiaría a quien tanto quiero.

Gracias

Agradecimientos

A mis preciados asesores, Lic Dolores Lankenau, Lic Bertha García y el Ing Cesar Centeno por compartir su tiempo, experiencia, conocimientos y su invaluable apoyo para la realización de esta tesis.

A Josué Medina, quien ha sido un excelente amigo y me ha ayudado a lo largo de la maestría.

A mis amigos Diana, Vlad, Jorge, Iván, Mac, Lee y Luis por brindarme su compañía tanto en los buenos como en los malos momentos.

A Dios, por guiarme por el mejor camino.

Gracias

Resumen

A través de los años se han realizado numerosos estudios sobre las causas del fracaso de los proyectos de desarrollo de software, todos estos estudios han tenido resultados similares, a pesar de los numerosos esfuerzos que aplican los genios en desarrollare nuevas metodologías persiste el problema. El software continua siendo tan complejo que es imposible estimar a ciencia cierta el costo y la duración de tiempo.

Es así como en los últimos años una nueva metodología hace su aparición, con el objetivo de proponer una metodología que comprende las mejores practicas de la industria del software. Y que permita crear un ambiente propicio con la ayuda de herramientas, casos de uso, requerimientos, automatización de procesos y desarrollando un entorno apropiado. Que permita desarrollar el software en serie.

Es así como este trabajo de tesis pretende demostrar que con los beneficios de una metodología robusta, se puede incrementar el grado de éxito en los proyectos de desarrollo y que en México existe una infinidad de posibilidades de implantación de una metodología que asegure el éxito de los proyectos de desarrollo de software, debido a que las empresas mexicanas de la actualidad aun cuentan con muchas diferencias respecto a la metodología que utilizan.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Graficas

Capitulo I Situación Problemática

Introducción

Las empresas, hoy más que nunca, sin importar su tamaño, giro de negocio o ubicación geográfica, se encuentran ante un reto sin precedente: la competitividad. La competencia global, los tratados de libre comercio entre las naciones, la creación de bloques económicos incluso entre países, las fusiones y la liberación de mercados, entre otros, son situaciones que sin duda alguna están poniendo a prueba la capacidad de reacción de cualquier organización.

Solamente aquellas empresas que han forjado culturas de aprendizaje continuo, de constante revisión de sus procesos, aquellas que han trabajado, o están trabajando para ser consideradas de clase mundial, tendrán la capacidad de sobrellevar estas situaciones.

En este sentido sólo aquellas que capten, gestionen y utilicen la información de una manera excelente, estarán destinadas a ver más allá del horizonte.

No hay duda que cada día que pasa las empresas dependen más de la tecnología de información para poder competir en un mundo cada vez más globalizado. Estamos en una era, donde para funcionar, las empresas deben de contar con una infraestructura digital que les ayude a desarrollar un nuevo tipo de inteligencia basada en la informática (Quintana 2000). El reto como lo menciona Gates (1999), es contar un sistema nervioso digital que le permita a las empresas reaccionar a la velocidad del pensamiento.

Es así como Feingenbaum y McCoruck referenciado por Pressman (2000) en su libro de administración de proyectos hablan sobre el impacto que tienen las aplicaciones de software en Estados Unidos de Norteamérica y el resto del mundo, donde plantea lo siguiente:

"El conocimiento es poder, la computadora a través del software ha sido el amplificador de ese poder. La industria norteamericana de software ha sido innovadora, vital, fructífera. Es decir de alguna manera, la industria ideal. Crea valor mediante la transformación de poder cerebral de los trabajadores del conocimiento, con poco consumo de energía y materiales nuevos".

Por lo que Pressman (2000) concreta "Lo que diferencia a una compañía de su competencia es la suficiencia y oportunidad de la información dada por el software. El software es el que marca la diferencia".

La industria del software, como todas las nuevas industrias debe de pasar por un proceso de maduración con el fin de alcanzar el éxito y la estabilidad. Como se vera a continuación también ha tenido problemas.

La industria de desarrollo de software ha estado evolucionado de manera importante en los últimos años, muchas empresas han buscado desarrollar software para comercializar o para uso estratégico en el negocio, por lo que invierten grandes cantidades de dinero en proyectos de software, tal como nos explica Standish Group en su Reporte Caos 1998, donde menciona: "los corporativos en América gastan mas de 250 billones cada año en aproximadamente 200,00 proyectos de software". Muchos de esos proyectos fallaran, no por falta de dinero o de tecnología; la mayoría fallaran por la falta de administradores de proyectos capacitados (Caos 1998).

A través de los años han sido realizados numerosos estudios con el objetivo de encontrar los principales causas que han llevado a los proyectos de software a fallar, entre dichos estudios se encuentra Caos(1998), "Patrones del fallas de software y éxito" (Jones 1996), entre otros, los cuales han demostrado el índice y las causas de fracaso de los grandes proyectos de software, como comenta pressman(2000) :"los mismo problemas que se han tenido durante años , son los mismo problemas que acontecen en la actualidad, todo esto debido a que realmente no se implementan metodologías de ingeniería de software".

Se piensa que hacer software es solamente programar y no se aplican metodologías que contemplen todo el proceso de planeación y diseño que tienen las disciplina de ingeniería de software.

Gates (1999) comenta "El principal objetivo es mejorar la calidad y reducir el costo de las soluciones basadas en software para computadoras."

Globerson; Zwikael, 2002 predice "El éxito de los proyectos esta medido por la habilidad de completar el proyecto de acuerdo a las especificaciones deseadas, dentro del presupuesto especificado y de acuerdo a los tiempos prometidos, a la vez que se mantienen felices a los clientes y a los inversionistas". No obstante año tras año, las cuestiones sobre desarrollo de software aparecen en los primeros puestos en las listas sobre temas críticos a los que se enfrenta la comunidad dedicada al desarrollo de software (Symons 1991).

También Sommerville (2000) comenta que "se debe aplicar una Administración de Proyectos de desarrollo Software complementado con una Ingeniería de software para asegurar que los proyectos sean exitosos".

En años recientes, se han estudiado y diseñado distintos paradigmas respecto a la administración de proyectos, de ahí la necesidad de estar en constante evaluación y mejora continúa de estos, teniendo como fin hacer más eficiente la administración de proyectos y lograr la exitosa implementación de los mismos.

El proceso unificado de Racional (RUP) enfatiza el desarrollo y el mantenimiento de modelos, representaciones semánticamente enriquecidas del sistema de software bajo desarrollo. (Boehm, 1996)

Su meta es la de asegurase una producción de alta calidad en el software y conocer las necesidades del usuario final, con un calendario y presupuesto predecible.

Desde este punto de vista Royce Walter, en su libro "Administración de Proyectos de Software: Un Enfoque Unificado", toma como base, la metodología del RUP, para hacer una nueva propuesta de administración de proyectos enfocados a la Ingeniería de Software (Royce, 1999).

1.2 Objetivo

Con el presente estudio se pretende confrontar la nueva forma de administrar el desarrollo de software basada en los principios de modelación orientada a objetos, comparándola con las metodologías de administración de proyectos de desarrollo de software que se utilizan actualmente en las empresas, con el propósito de contar con un marco de referencia, que muestre las ventajas y áreas de oportunidad de la empresas para poder aplicar la metodología moderna.

Para precisar, el objetivo de esta tesis es: analizar los beneficios y las posibles oportunidades que resultan de la aplicación de las nuevas metodologías de administración de proyectos de desarrollo de software, que utilizan procesos de instrumentación recomendadas en la ingeniería de software, específicamente recomendados por el Proceso Unificado de Racional (RUP).

1.3 Restricciones

Debido a que la investigación estará centrada en la administración de proyectos enfocados al desarrollo de software, la tesis se limitara solo a empresas que tengan la función de desarrollo de software en su organización para su uso interno o comercial.

Debido al tiempo en que se realizará la tesis y a los recursos disponibles, solo se realizara el estudio en empresas que se encuentren en el área metropolitana de la ciudad de Monterrey, N.L.

No se contemplaran empresas pequeñas, dado que difícilmente se encontrara alguna con la capacidad de inversión en metodologías e instrumentos para el desarrollo de software.

La cantidad y la calidad de la información dependen en gran medida de la disponibilidad de la empresa para proporcionarla.

1.4 Estructura de la Tesis

La información presentada en esta tesis está dividida en 6 capítulos, los cuales integran los siguientes aspectos relevantes:

Capítulo 2. En este capitulo se concentra el sustento bibliográfico de la presente investigación o lo que es lo mismo el marco teórico.

Capítulo 3. En este capitulo se presenta el modelo particular de la presente investigación, en el cual la pretende demostrar que existe áreas de oportunidad en las empresas mexicanas de implementar metodologías robustas, como la metodología de enfoque iterativo, que ayudaran a minimizar los problemas que actualmente cuentan dichas empresas.

Capítulo 4. Se presenta la metodología de Investigación llevada a cabo para la recopilación de información, el número de encuestas aplicadas; así como la estructura del cuestionario,

Capítulo 5. Se presentan los resultados y observaciones obtenidos de la investigación de campo, analizando cada uno de los elementos cuestionados, particularmente los factores que inhiben el desarrollo de los proyectos de informática.

Capítulo 6. Se establecen las conclusiones de la tesis, así como la elaboración de trabajos futuros.

Capitulo II. Marco teórico

En este capítulo se expondrá la bibliografía que soportará el modelo particular de la presente investigación.

Entre los principales apartados, se tiene a las situación actual del desarrollo de software, la problemática, las oportunidades presentadas en el desarrollo del software, las metodologías tradicionales de desarrollo de software, la administración de proyectos y por último se detallará la estructura de la metodología propuesta por Royce (1998) y las disciplinas de administración de proyectos propuestas por el mismo autor.

2.1 Situación actual en el desarrollo de Software

Aunque el término Software es utilizado diariamente, es importante definirlo con Claridad. De esta manera brooks (2000) nos da su definición de software:

"El software son las instrucciones electrónicas que van a indicar al ordenador que es lo que tiene que hacer. También se puede decir que son los programas usados para dirigir las funciones de un sistema de computación o un hardware."

Una vez que el término ha sido definido se procede a describir la situación actual del desarrollo de software. Haciendo una búsqueda se encontrara que muchos autores han hablado de las oportunidades que han surgido en la industria de desarrollo de software ese aspecto Booch (1997) comenta "A nivel mundial existe una demanda insaciable de software, a primera vista es una buena noticia. Se pensaría que son los tiempos adecuados para que los desarrolladores de software tengan éxito, debido a existirán un sinnúmero de de oportunidades, pero son malas noticias, debido a que ningún número de programadores será suficiente para satisfacer esa demanda".

Además de esta demanda insaciable, la tasa de cambio en la tecnología de desarrollo de software es casi instantánea. Dieciocho meses en el calendario es un Siglo en años de software. Una sola distracción y se pierde el próximo gran cambio. Solo unos años atrás, C++ era lo novedoso, en estos días es Java y Visual Basic, la programación visual y la programación basada en componentes han cambiado sus reglas. Es difícil predecir lo que vendrá después. Booch (1997)

Aunque es difícil predecir el cambio tecnológico, con toda seguridad se puede predecir que en el futuro el software va a ser más complicado y esto lo llevara a ser más distribuido. Por mucho será más complejo, principalmente por el empuje del suministro y la demanda. Las expectativas del usuario con respecto a lo que puede hacer el software son muy altas. El costo de la computación ha venido en descenso en los últimos años, pero el costo y la complejidad de desarrollo informático han continuado aumentando. En el futuro, el software será más distribuido y difícil de manejar, esto aumentara el tiempo y consto de su conclusión.

Además, la ingeniería de software es dominada por las actividades intelectuales que son enfocadas a resolver problemas de inmensa complejidad con numerosa incertidumbre en perspectivas competitivas.

Debido a lo anterior, ha surgido la necesidad de buscar nuevas metodologías y tecnologías que traten de controlar el crecimiento de la complejidad del software, y que ayuden a que los proyectos se terminen en el tiempo y costo asignado inicialmente, lo que llevara a incrementar el porcentaje de proyectos exitosos de software.

También se observa como la industria del software esta alcanzando su 50 aniversario en el mercado y los mismos problemas que se tuvieron hace 20 años, persisten en la actualidad. Las empresas se han propuesto elaborar software de alta tecnología que no funciona, tampoco puede ser utilizado, mucho menos modificado o mantenido. Muchos de esos presupuestos altos y programas caducados han sido cancelados después de alcanzar una producción a gran escala con millones de dólares desperdiciados.

2.2 Problemática

Los problemas que afligen al desarrollo del software se pueden caracterizar bajo perspectivas diferentes, pero los responsables de los desarrollos de software se centran sobre los aspectos de fondo: la planificación y estimación de costos son frecuentemente muy imprecisas: la productividad de la comunidad de desarrolladores de software no corresponde al de la demanda de servicios y la calidad del software no llega a ser ni siquiera a veces aceptable. Se ha errado en la planificación en meses o años, se ha hecho muy poco para mejorar la productividad de los trabajadores del software. Los errores en los nuevos programas producen insatisfacción y una falta de confianza, tales problemas son solo las manifestaciones más visibles de otras dificultades del software. (Pressman 2000)

Principales problemas.
Falta de tiempo al reconocer datos sobre e proceso de desarrollo de software.
Insatisfacción del cliente con el sistema terminado se produce demasiado frecuente
La calidad del software es normalmente cuestionable
El software existente puede ser muy difícil de mantener

Tabla 1 - Problemas Comunes en el desarrollo de Software.

Fuente Pressman (2000)

A través de los años, estos problemas han sido estudiados por expertos de la industria y del gobierno, todos alcanzando la misma conclusión. La inhabilidad de construir software redituable y económico no es debido a problemas técnicos, sino debido a una práctica administrativa pobre. Aquí se aprecia que los proyectos fallan debido a:

- La complejidad inherente del software.
- La deficiente estimación de tamaño.
- La ineficiente estimación de tiempo y costo. (Pressman 2000)

A mediados de los 90's al menos tres importantes análisis del estado de la industria de la ingeniería de software fueron realizados. Los resultados fueron presentados en los reportes "Patrones del fallas de software y éxito" (Jones 1996), "Caos" (Standish Group 1995) y el"Reporte de la fuerza de tareas en adquisición de software comercial de defensa por el ministerio defensa".

En el reporte de Jones (1996) se detectaron los factores observados en los proyectos de software, a continuación se muestra en tabla 2 los resultados publicados.

Proyectos sin éxito	Proyectos exitosos
Excesiva presión en el calendario	Expectativas de calendario realistas
Ejecutivos rechazando los estimados	Ejecutivos entendiendo lo estimados
Severas fricciones con el cliente	Cooperación con los clientes
Políticas corporativas de división	Metas administrativas congruentes
Pobre comunicación del equipo	Excelente comunicación del equipo
Ejecutivos con poca experiencia	Ejecutivos experimentados
Staff técnico descalificado	Staff Técnico capaz
Mala practica en administración de proyectos	Administración de proyectos capaz
Personal general usado para tareas criticas: aseguramiento de calidad, pruebas planeación y estimaciones	Personal especializado usado para tareas criticas: aseguramiento de calidad, pruebas planeación y estimaciones

Tabla 2 - Factores de Administración de proyectos

Fuente: Jones 1996

Por otro lado Standish Group en su reporte Caos (1995) se enfoca en la industria de software comercial y muestra las siguientes conclusiones:

- Las compañías de Estados Unidos gastaron $81 billones en cancelar los proyectos de software en 1995.
- El 31% de los proyectos de software estudiados fueron cancelados antes de haber sido terminados.
- El 53% de los proyectos de software se sobrevaluaron por mas de 50%.
- Solamente el 9% de los proyectos de software en compañías grandes fueron terminados en tiempo y con el presupuesto asignado, para empresas medianas y compañías pequeñas, los números mejoraron a 16% y 28% respectivamente.

El Standish Group (CAOS 1998) clasifica los proyectos en:

Exitosos: Son los proyectos que son completados en tiempo y con el presupuesto asignado, con todas las características y funciones que inicialmente se especificaron

Emplazados: El proyecto es completo y operacional, pero por arriba del presupuesto y por encima del tiempo estimado, con menos detalles y funciones de las que inicialmente se especificaron.

Fallidos: El proyecto es cancelado antes de su terminación

El Grupo reporto que bs proyectos fallidos han disminuido del 40% al 23 % en los últimos 5 años, pero los proyectos que se emplazaron se incrementaron del 33% al 49% en el mismo periodo, esto no es bueno porque los proyectos emplazados perjudican mas que los proyectos que simplemente fallan, pues se están modificando y ajustando lentamente en lugar de hacer un desarrollo rápido. Estos proyectos a menudo son fracasos en potencia, donde se pierde gran cantidad de tiempo y dinero para poder completarlos.

Estos análisis proveen una buena introducción de la magnitud del problema en el desarrollo de software y de las normas el desempeño de las metodologías convencionales de administración del desarrrollo de software. (Royce 1998)

La industria del software a pesar de haber cumplido sus primeros 50 años, manteniendo los mismos problemas que en el pasado (Insatisfacción del cliente, calidad del software es cuestionable, difícil mantenimiento del software), así lo demuestran los distintos Estudios como CAOS 1995 de Standish Group en donde dice que el 84% de los proyectos se deben considerar un fracaso. A lo que Booch (1997) concreta la mayoría de practicas de administración en la industria del software de los 90's sigue reflejando un proceso inmaduro caracterizado por el desperdicio y el re-trabajo.

2.3 Oportunidades en el desarrollo de software

"El software de hoy en día no solamente automatiza procesos; debe crear valor al negocio por medio de mejorar el servicio al cliente o lograr ventaja competitiva. El retorno de inversión (ROI) aumenta los riesgos de cada desarrollo de proyectos a gran escala; el software debe tener un impacto predecible en la línea de negocios de la compañía". (CAOS 1998)

Booch (1997) cometa que en su experiencia "hay un número de mejores prácticas encontradas en común entre los proyectos que tienen éxito. Dos de esas prácticas que se destacan son enfocarse en la arquitectura, y enfocarse en un proceso iterativo e incremental de desarrollo."

Enfocarse en la arquitectura significa no sólo construir grandes clases y algoritmos, sino que también deben de elaborar colaboraciones simples y expresivas de esas clases y algoritmos. Todos los sistemas de calidad parecen llenos de estos tipos de colaboraciones y trabajo corriente en el área de patrones informáticos comienza a nombrarlos y clasificarlos a fin de que ellos fácilmente puedan ser reutilizados (Gamma et al. 1995). Las mejores arquitecturas utilizadas tienen lo que Fred Books llama la "integridad conceptual," eso se deriva de enfocar el proyecto a sacar provecho de estos patrones y hacerlos simples, lo cual resulta muy difícil de conseguir.

Un proceso iterativo e incremental de desarrollo refleja el ritmo del proyecto. Los proyectos en crisis no tienen ritmo, pues tienden a ser oportunistas y reactivos en su trabajo. Los proyectos exitosos tienen un ritmo, reflexiona en un proceso regular de liberación en el que tiende a enfocar la atención en el refinamiento sucesivo de la arquitectura de los sistemas. (Royce 1998)

Debido a lo anterior, muchos países están considerando a la industria de desarrollo de software como una oportunidad de crecimiento al implantar mejores metodologías de desarrollo que les ayudan a asegurar un producto de calidad.

Merchant (2001) comenta “las compañías de software de la India esperan trasladar la recesión que sufre los Estados Unidos en una oportunidad de hacer negocio. Las compañías Indias actualmente poseen el 0.5% de mercado de software. Este porcentaje posiblemente se elevará debido a que estas compañías han adoptado procesos de madurez como el Modelo de Madurez de Capacidades (CMM) con el fin de desarrollar software de manera eficiente, esto se ve reflejado en que 20 de las 37 empresas que poseen el nivel 5 de madurez del CMM, son de éste país.”

La industria del software representa en la actualidad alrededor del 20% del mercado mundial de la informática, las ganancias por las ventas exceden los cien mil millones de dólares norteamericanos anualmente. (Álvarez, s.f.)

Álvarez (0000) argumenta”Los países en desarrollo tienen la posibilidad de desarrollarse en el área del software, ya que los productos del software son de valor agregado puramente. La India con una venta de 1000 millones en 1997 es el país de más logros entre los países en desarrollo.”

El desarrollo de la industria del software para los países en vías de desarrollo es una posibilidad que ha tenido éxito en algunos países como la India. Sin embargo se requiere para ello mejorar, entre otros factores, la productividad y la calidad del desarrollo. Lo antes dicho, requiere del establecimiento de sistemas de calidad. (Álvarez s.f.)

En el 2000 se estimaba que había un déficit de 600,000 expertos en desarrollo de software en estados unidos.

Según Zermeño (2003) en México existen oportunidades de desarrollar una industria de desarrollo de software debido a:

- Ubicación privilegiada
- Costos competitivos
- Cultura empresarial similar a la de estados unidos

Que generara beneficios como:

- Atrae inversionistas de TI
- Fuerte efecto multiplicador, lo que mejora la economía de la región
- Puente tecnológico entre estados unidos y Latinoamérica
- Fomenta la desconcentración de población
- Puede llegar a cambiar el perfil exportador
- Promueve el empleo masivo
- Evita fuga de cerebros
- Permitirá ser jugadores importantes en el mundo de las tecnologías de TI
- Permite pasar al lado correcto e la vertiente digital.

Actualmente en México existe poca oferta de desarrollo de software, el nivel de calidad o certificación en las empresas dedicadas al desarrollo es muy bajo o casi inexistente, y la industria se encuentra muy dispersa, sólo una empresa en México tiene más de 1,000 empleados, muy pocas más de 100 y muchas debajo de 50 (AMITI,0000)

Para que México pueda exportar software de calidad y escapar de la brecha respecto al desarrollo de software, se tiene que promover el desarrollo de compañías de excelencia que usen procesos reconocidos, que puedan crecer para lograr las economías de escala, que desarrollen el mejor personal, y además que generen experiencia para la industria. (AMITI, 0000)

Es así como México también cuenta con la oportunidad de "encontrar mejores metodologías de desarrollo de software que ayuden a producir un producto de calidad a bajo costo, que permita competir en un mercado global y lo impulse como un país exportador de conocimiento". Zermeño (2003)

Por ultimo Booch (1997) concreta "Es importante pasar de alguien que solo escribe líneas de código a un ingeniero que hace mucho mas al preguntarse que estoy haciendo con y por el proyecto con el que estoy trabajando." Para lograre lo anterior es necesario que se aplique una ingeniería de software capaz de asegurar un producto de calidad.

2.4 Ingeniería de Software

Galáz (2002) comenta "El término de ingeniería de software fue introducido a finales de los 60 a raíz de la crisis del software. Esta crisis fue el resultado de la introducción de la tercera generación del hardware. El hardware dejo de ser un impedimento para el desarrollo de la informática; redujo los costos y mejoro la calidad y eficiencia en el software producido."

La crisis se caracterizo por los siguientes problemas:

- Imprecisión en la planificación del proyecto y estimación de los costos.
- Baja calidad del software.
- Dificultad de mantenimiento de programas con un diseño poco estructurado, etc.

"Para lograr que el producto de software esta bien diseñado se debe aplicar la ingeniería de software. Al igual que otros tipos de ingeniería, la ingeniería de software no solamente consiste en producir productos, más bien significa producir los productos de la mejor relación costo-beneficio posible. Asignando una cantidad de recursos ilimitados, la mayoría de los problemas de software podrían ser resueltos, pero el reto de la ingeniería de software es producirlo con alta calidad y una cantidad finita de recursos en el tiempo estimado." Sommerville (1989)

Sommerville (1989) también menciona cuatro atributos que cualquier producto de software bien diseñado deben tener las siguientes características:

- **Fácil Mantenimiento:** Un software con una larga vida útil, esta sujeto a un cambio regular, por lo que es importante que el software sea escrito y documentado de modo que los cambios puedan ser hechos sin costos inmerecidos.

- **Confiable:** Un nivel apropiado de confiabilidad es esencial, si el software es de uso general.

- **Eficiente:** Esto no necesariamente significa que el desempeño debe ajustarse al hardware donde el software se ejecuta. Ciertamente, tratar de maximizar la eficiencia puede hacer al software mucho más difícil de cambiar. Sin embargo, que sea eficiente significa que el software no debe desperdiciar el uso de recursos del sistema como la memoria y el procesador.

- **Fácil de Usar**: La mayoría de los usuarios no utilizan el potencial completo del software, porque la interfaz en la cual se ofrece, es difícil de utilizar. En el diseño de la interfaz se debe de tomar en cuenta las capacidades y los antecedentes de los usuarios a los que va dirigido el software, es decir de debe estar hecho a la medida de los usuarios.

En este aspecto, la ingeniería de software a través de las metodologías de desarrollo de software provee una guía que sirve como base para el desarrollo de software de manera exitosa.

2.5 Metodologías de Desarrollo de Software

La función primaria de una metodología de proceso de desarrollo de software es el determinar el orden de las etapas envueltas en el desarrollo, evolucionar y establecer el criterio de transición para procesar de una etapa a la siguiente. (Laplante 1999)
Las metodologías de proceso de software son importantes porque proveen una guía del orden de las etapas, incrementos, prototipos, tareas de valuación, entre otras (Laplante 1999)

Castro (2000) comenta "Hoy en día las empresas dedicadas al desarrollo de productos software invierten grandes cantidades de dinero en la implementación de metodologías para la ingeniería de proyectos, o simplemente no cuentan con un marco de trabajo en el cual sus desarrolladores se puedan basar para la creación de productos computacionales que cumplan con los estándares de calidad, al menos los establecidos por la misma empresa.

La estimación del calendario es y ha sido la mayor dificultad en la administración de desarrollo de proyectos. (Laplante 1999)

2.5.1 El Modelo de Cascada

El modelo de cascada es atribuido a Walter Royce, (Van 2000) el cual es descrito por Pressman a continuación:

La Figura 1 ilustra el paradigma del ciclo de vida clásico para la ingeniería del software. Algunas veces llamado 'modelo en cascada', el paradigma del ciclo de vida exige un enfoque sistemático y secuencial del desarrollo de software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento. Conceptualizado a partir del ciclo convencional de una ingeniería, el paradigma del ciclo de vida abarca las siguientes actividades:

Ingeniería y análisis del sistema. Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos del software. Este planteamiento del sistema es esencial cuando el software debe interrelacionarse con otros elementos, tales como hardware, personas y bases de datos. La ingeniería y el análisis del sistema abarcan los requisitos globales a nivel del sistema con una pequeña cantidad de análisis y diseño a nivel superior.

Análisis de los requisitos del software. El proceso de recopilación de requisitos se centra e intensifica especialmente para el software. Para comprender la naturaleza de los programas que hay que construir, el ingeniero de software ('analista') debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas. Los requisitos, tanto del sistema como del software, se documentan y se revisan con el cliente.

Diseño. El diseño del software es realmente un proceso múltiple que se enfoca sobre cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. Al igual que los requisitos, el diseño se documenta y forma parte de la configuración del software.

Codificación. El diseño debe traducirse en forma legible para la máquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

Prueba. Una vez que se ha generado el código, comienza la prueba del programa. La prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado, se realizan pruebas que aseguren que la entrada definida produce los resultados que realmente se quieren.

Mantenimiento. El software, indudablemente sufrirá cambios después de que se entregue al cliente una posible excepción es el software empotrado. Los cambios ocurrirán debido a que se hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo por ejemplo, un cambio solicitado debido a que se tiene un nuevo sistema operativo o dispositivo periférico, o debido a que el cliente requiera ampliaciones funcionales o de rendimiento. El mantenimiento del software aplica cada uno de los pasos procedentes del ciclo de vida a un programa existente en vez de a uno nuevo.

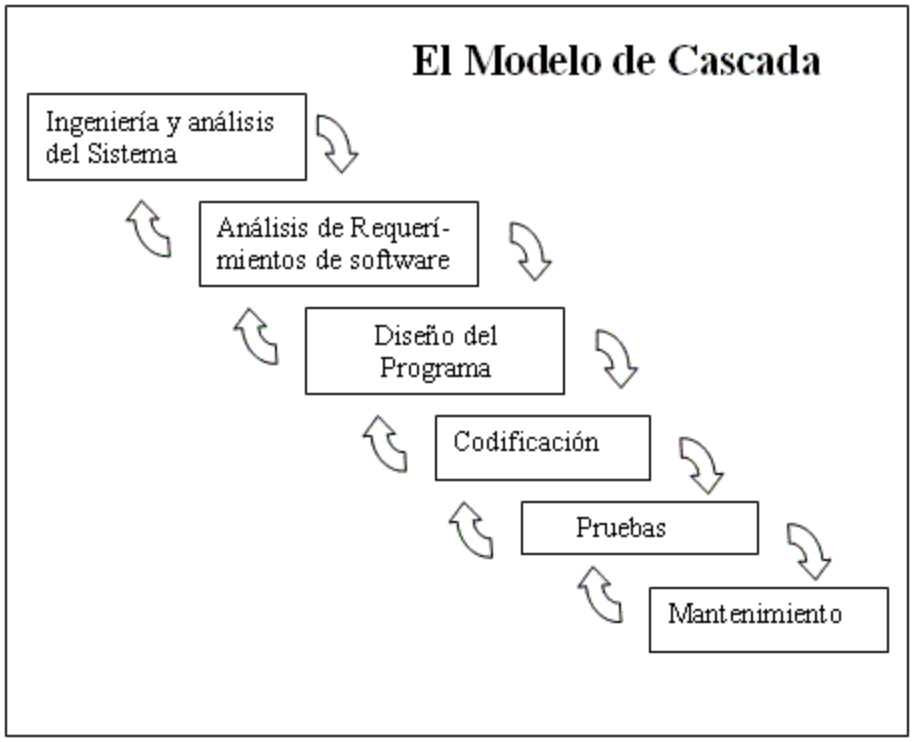


Figura 1- Esquema de Fases del Modelo de Cascada

Fuente: Van 2000

El modelo de cascada expresa particularmente la relación entre las fases subsecuentes:

- Requerimientos
- Diseño
- Implementación
- Pruebas
- Mantenimiento
- Documentación

Esta metodología según Walter Royce (1998) presenta algunos problemas que tienen las siguientes áreas de oportunidad.

Los 10 principios (Problemas) de la administración convencional (Waterfall Model) de software.

1.- Congelar los requerimientos antes de diseñar: Es la esencia de un proceso que se basa en tener primero los requerimientos antes que cualquier cosa: El equipo del proyecto se esfuerza por proveer una definición precisa de los requerimientos y luego implementar exactamente tales requerimientos.

2.- Evitar codificar antes de hacer una revisión detallada del diseño: Debido a que los cambios en el diseño pueden causar una fractura significativa en las fases de codificación y pruebas, el equipo necesita asegurarse que el diseño este completo en una etapa completa y madura antes de comenzar con la fase de codificación, en donde hay mucho más resistencia al cambio.

3.-Usar un lenguaje de programación de alto-nivel: Los lenguajes de programación de alto nivel evitan tener contacto con un conjunto significativo de fuentes de errores (empleando la escritura avanzada de datos, la separación de la interfaz, y la construcción del empaquetado y la programación) y permite que la solución de software emplee menos líneas de código generadas por los humanos.

4.- Terminar la unidad de prueba antes de la integración: Mientras que el diseño fluye de arriba hacia abajo, el proceso de pruebas fluye de abajo hacia arriba: las unidades más pequeñas se prueban completamente antes de enviarlo a la prueba de integración. Esta depuración constante es un intento de encontrar más fallas de programación (bugs) en este nivel, antes de pasar a la integración, en donde pueden causar un mayor desperdicio y re-trabajo.

5.- Mantener una capacidad detallada para trazar entre todos los artefactos: Para asegurarse que la consistencia y la completitud del programa se pueden mantener en cada etapa, los artefactos de los requerimientos necesitan trazarse para después poder diseñarlos y probarlos. Cuando se proponen o identifican cambios sobre la marcha, se proporciona una visión completa del impacto real o potencial del cambio.

6.- Documentar y mantener el diseño: El diseño sin documentación no es diseño. En las etapas iniciales, la documentación "es" el diseño. En las últimas etapas, como el código se vuelve el instrumento de ingeniería primario, los artefactos de diseño se deben actualizar para asegurar la consistencia y proveer una base para la toma de decisiones sobre los cambios.

7.- Determinar la calidad con un equipo independiente: Para mantener una cadena de reportes separada de los analistas, diseñadores e implementadores, el proyecto debe asignar a un equipo independiente la responsabilidad para asegurar el cumplimiento total de los estándares de calidad, tanto para los productos como para los procesos.

8.- Examinar todo: Examinar el diseño y el código a detalle es una manera de encontrar errores que a través de las pruebas. Asegurándose que los exámenes cubran todos los artefactos que provienen de los requerimientos, diseño, codificación y pruebas.

9.- Planear todo al inicio con un alto nivel de detalle: Un plan completo y preciso cae en la categoría de piedra angular, ya que presenta al perfecto detalle las actividades y artefactos con sus respectivas fechas de entrega, por lo tanto es necesario identificar las rutas críticas, manejar los riesgos, y evaluar los cambios de programación de actividades.

10.- Controlar rigurosamente las bases de código fuente: Una vez que los artefactos caen en la etapa de codificación, es necesaria una administración de la configuración para mantener un control central de las versiones formales en el proceso de pruebas y para que el producto evolucione a una etapa de cero defectos, favorable para su liberación.

2.5.2 Construcción de prototipos (pressman 2000)

Normalmente un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el programador puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma en que debe realizarse la interacción hombre-máquina. En estas y muchas otras situaciones, puede ser mejor método de ingeniería del software la construcción de un prototipo.

La construcción de prototipos es un proceso que facilita al programador la creación de un modelo del software a construir. El modelo tomará una de las tres formar siguientes:

1.- Un prototipo en papel o un modelo basado en PC que describa la interacción hombre-máquina, de forma que facilite al usuario la comprensión de cómo se producirá tal interacción.
2.- Un prototipo que implemente algunos subconjuntos de la función requerida deseada, o
3.- Un programa existente que ejecute parte o toda la función deseada, pero que tenga otras características que deban ser mejoradas en el nuevo trabajo de desarrollo.

La figura 1.8 muestra la secuencia de sucesos del paradigma de construcción de prototipos. Como en todos los métodos de desarrollo de software, la construcción y el cliente se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesario una mayor definición. Luego se produce un 'diseño rápido'. El diseño rápido se enfoca sobre la representación de los aspectos del software visibles al usuario, por ejemplo, métodos de entrada y formatos de salida. El diseño rápido conduce a la construcción de un prototipo. El prototipo es evaluado por el cliente usuario y se utiliza para definir los requisitos del software a desarrollar. Se produce un proceso interactivo en el que el prototipo es 'afinado' para que satisfaga las necesidades del cliente, al mismo tiempo que facilita al que no lo desarrolla una mejor comprensión de lo que hay que hacer.

Idealmente, el prototipo sirve como mecanismo para identificar los requisitos del software. Si se va a construir un prototipo que funcione, el realizador intenta hacer uso de fragmentos de programas existentes o aplica herramientas por ejemplo, generadores de informes, gestores de ventanas etc., que faciliten la rápida generación de programas que funciones.

Errores
El prototipo puede servir como primer sistema aquél que Brooks recomienda que tiremos. Pero esto puede ser una visión idealizada. Al igual que en el ciclo de vida clásico, la construcción de prototipos como paradigma para la ingeniería del software, puede ser problemática por las siguientes razones (Pressman 2000):

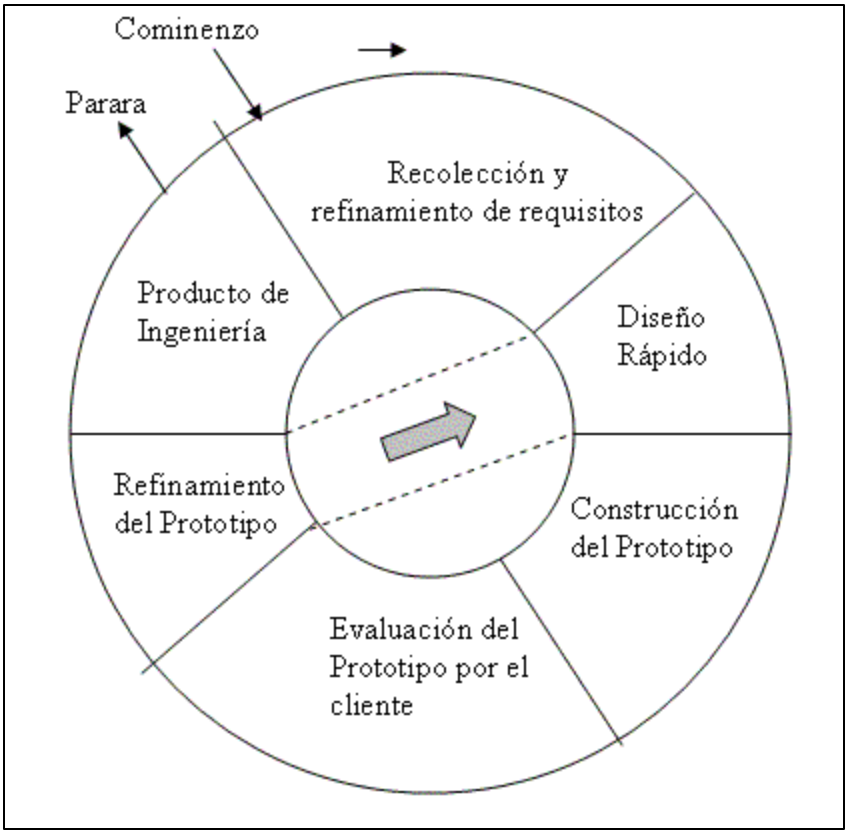


Figura 2 - Esquema de prototipos.

Fuente: Pressman 1998

1. El cliente ve funcionando lo que parece ser una primera versión del software, ignorando que el prototipo de ha hecho con plastilina y alambres, ignorando que, por las prisas es hacer que funcione, no hemos considerado los aspectos de calidad o de mantenimiento del software a largo plazo. Cuando se le informa de que el producto debe ser reconstruido, el cliente se vuelve loco y solicita que se apliquen cuantas mejoras sean necesarias para hacer el prototipo un producto final que funcione. El administrador del desarrollo del software cede demasiado a menudo.
2. El técnico de desarrollo, frecuentemente, impone ciertos compromisos de implementación con el fin de obtener un prototipo que funcione rápidamente. Puede que utilice un sistema operativo o un lenguaje de programación inapropiados, simplemente porque ya está disponible y es conocido; puede que implemente ineficientemente un algoritmo, sencillamente para demostrar su capacidad. Después de algún tiempo, el técnico puede haberse familiarizado con esas elecciones y haber olvidado las razones por las que eran inapropiadas. La elección menos ideal forma ahora parte integral del sistema.

2.5.3 El Modelo Incremental

En el modelo incremental el software es construido, no escrito, esto es, el software va siendo construido paso á paso, de la misma manera en que se construye un edificio. Mientras un producto de software esta en el proceso de ser desarrollado, cada paso adiciona algo sobre lo que se ha hecho anteriormente. Día a día los módulos del sistema son añadidos, el desarrollo del producto se procesa incrementalmente hasta la conclusión del producto.

Por supuesto no es muy cierto que el progreso se hará diariamente, así como un constructor que tiene que tirar una pared mal posicionada o remplazar una ventana que un pintor han quebrado, algunas veces es necesario re-especificar , re-diseñar , re-codificar o en el peor de los casos , eliminar lo que ya se había completado y empezar de nuevo . Pero el hecho que el producto sufre algunos ajustes y vuelve a empezar, no niega que la realidad básica es, que el software es producido pieza por pieza.

El hecho de que este software es construido incrementalmente, ha llevado a los desarrolladores del modelo que explotan ese aspecto del desarrollo de software, a llamarlo modelo incremental. El producto es diseñado implementado, integrado y probado como una serie de construcciones incrementales, donde la construcción consiste en piezas de código de varios módulos interactuando para proveer de una capacidad funcional específica.

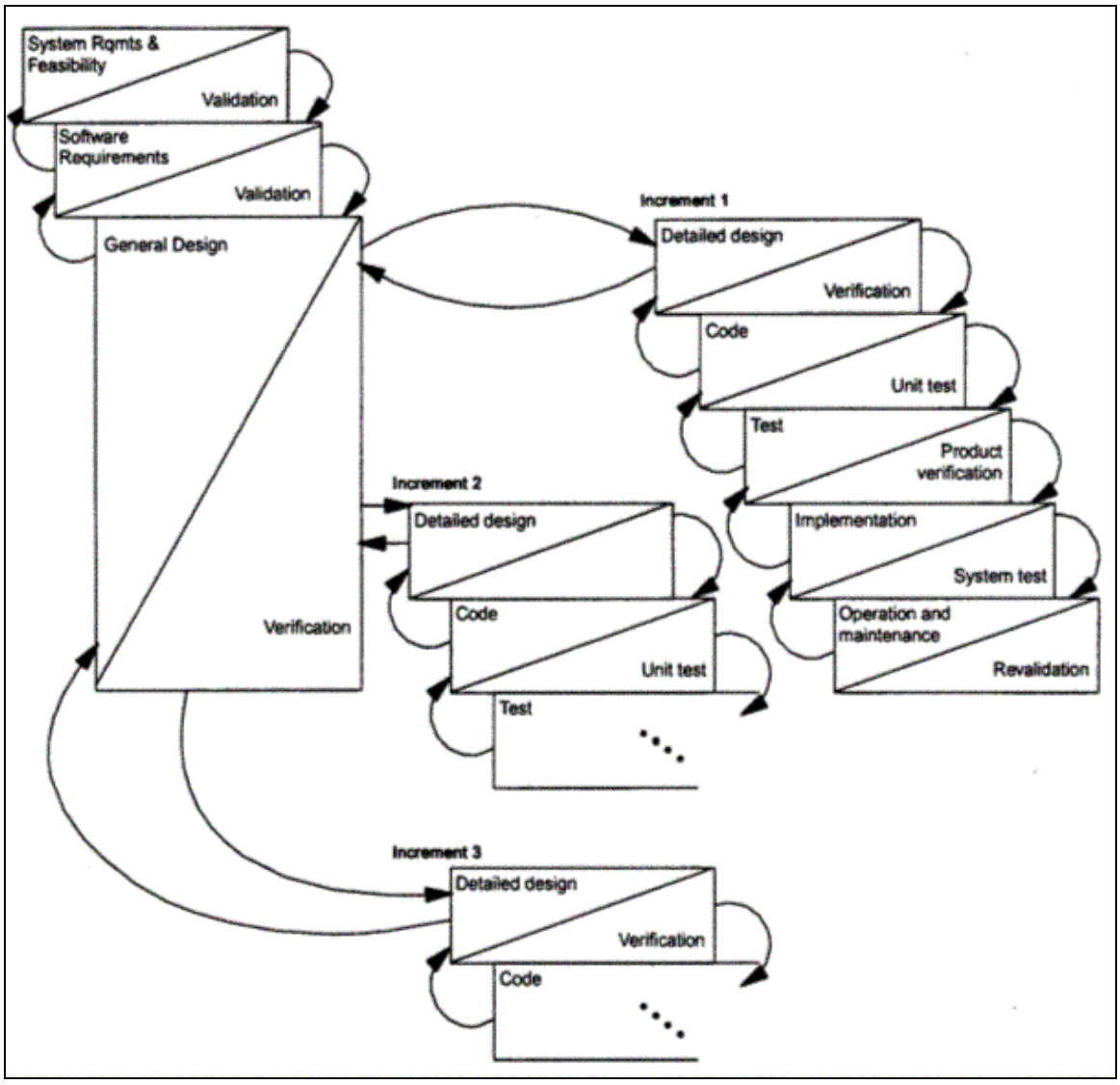


Figura 3 - Fases del Modelo Incremental.

Fuente: Van 2000

2.5.4 Modelo de Espiral

Por lo general un elemento de riesgo esta envuelto en el desarrollo de software, por ejemplo, personal clave puede renunciar antes que el producto haya sido adecuadamente documentado. También que el proveedor de hardware en el cual el producto es críticamente dependiente puede irse a la bancarrota. O la compañía puede investigar y desarrollar un sistema de administración de base de datos, pero antes que el producto pueda ser vendido, un paquete funcionalmente equivalente y a precio mas bajo es anunciado por la competencia. Los componentes de un producto construido usando el modelo incremental visto anteriormente pueden no ajustar juntos. Por obvias razones, los desarrolladores de software tratan de minimizar dichos riesgos como sea posible.

El modelo de espiral completo se muestra en la figura 4. La dimensión radial representa el costo acumulativo al día. La dimensión angular representa el progreso a través de la espiral. Cada ciclo de la espiral representa una fase, una fase empieza (en la parte izquierda superior del cuadrante) definiendo los objetivos de esa fase, alternativas para alcanzar esos objetivos, y obligaciones impuestos a esas alternativas. Este proceso resulta en una estrategia para alcanzar esos objetivos, esta estrategia es analizada desde el punto de vista de riesgo. Muchos intentos son hechos para resolver cada riesgo potencial, en algunos casos construyendo un prototipo. Si el riesgo no puede ser resuelto, el proyecto deberá terminar inmediatamente, bajo algunas circunstancias, una decisión podría ser continuar el proyecto pero a una escala menor, si todos los riesgos son resueltos exitosamente, el próximo paso en el desarrollo debe ser iniciado (cuadrante superior derecho). Este cuadrante del modelo de espiral es similar al modelo de cascada. Finalmente, los resultados de esta fase son evaluados y se planea la siguiente fase.

El modelo de espiral ha sido usado exitosamente para desarrollar una amplia variedad de productos. En un total de 25 proyectos en los cuales el modelo de espiral fue usado en conjunción con otros medios para incrementar la productividad, esta se incremento cada proyecto por lo menos en un 50 % sobre los niveles previos de productividad y en un 100% en muchos de los proyectos (Boehm, 1998)

Los errores

El paradigma del modelo en espiral para la ingeniería del software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala. Utiliza un enfoque evolutivo para la ingeniería del software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción del riesgo, pero lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución del producto. Mantiene el enfoque sistemático correspondiente a los pasos sugeridos por el ciclo de vida clásico, pero incorporándola dentro de un marco de trabajo interactivo que refleja de forma más realista el mundo real. El modelo espiral demanda una consideración de riesgos técnicos en todas las etapas del proyecto, si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemáticos.

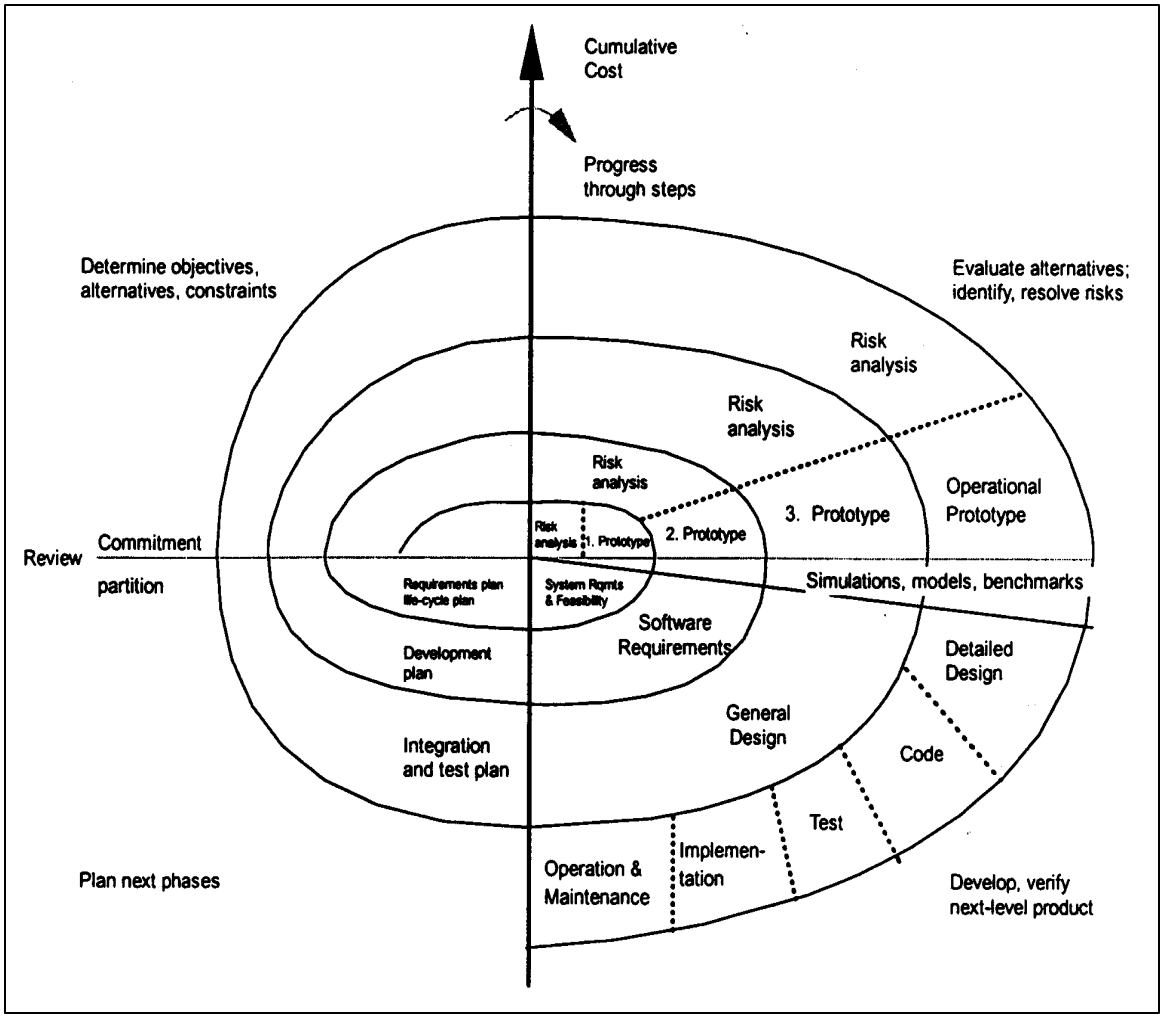


Figura 4 - Esquema de las fases del Modelo Incremental.

Fuente: Van 2000

Pero, al igual que otros paradigmas, el modelo en espiral no es la panacea. Puede ser difícil convencer a grandes clientes (particularmente en situaciones bajo control) de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la valoración del riesgo, y contar con esta habilidad para el éxito. Si no se descubre un riesgo importante, indudablemente surgirán problemas. Por último, el modelo es sí mismo es relativamente nuevo y no se ha usado tanto como el ciclo de vida o la creación de prototipos. Pasarán unos cuantos años antes de que se pueda determinar con absoluta certeza la eficacia de este importante nuevo paradigma.

2.5.5 Proceso Unificado (RUP)

La metodología del Proceso unificado de Racional (RUP) es un compendio de ideas y experiencias de los líderes de la industria, socios y literalmente cientos de proyectos reales cuidadosamente sintetizados dentro de un conjunto de mejores prácticas, flujos de trabajo y objetos para un desarrollo de software iterativo. El RUP se ha convertido rápidamente en el estándar de los 5 grandes integradores de sistemas en la industria de desarrollo. También 8 de los 10 bancos más grandes de los Estados Unidos y cientos de proyectos a nivel mundial confían en RUP. Cuando son usadas en combinación las mejores practicas promovidas por la metodología del RUP, las cuales incluyen desarrollo iterativo, administración de requerimientos, uso de arquitecturas de componentes, modelado visual, administración de cambios y continua verificación de calidad, golpea a la raíz de los problemas de desarrollo de software, ayudando a evitar caídas comunes con el apalancamiento de nuevas tecnologías y herramientas. Usando esta metodología probada y compartiendo un proceso compresivo, el equipo será capaz de comunicarse más efectivamente y trabajar más eficientemente. M&R (1998)

El Proceso Unificado guía a los equipos de proyecto en la administración el desarrollo iterativo de un modo controlado, mientras se balancean los requerimientos del negocio, el tiempo del mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y el establecimiento de una guía arquitectónica para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura. El proceso describe qué entregables producir, cómo desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

"El proceso Unificado es un proceso de desarrollo de software configurable que se adapta a proyectos que varían en tamaño y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología de objetos en el desarrollo de software de misión crítica en una variedad de industrias. Uno de los componentes clave es el UML" (M&R 1998).

El Proceso Unificado ha adoptado un enfoque que se caracteriza por:

- Interacción continua con el usuario desde un inicio.
- Mitigación de riesgos antes de que ocurran.
- Liberaciones frecuentes de ejecutables.
- Aseguramiento de la calidad.
- Participación del equipo en todas las decisiones del proyecto.
- Anticiparse al cambio de requerimientos.

Según el mismo autor, las características primordiales del Proceso Unificado son:

- Proceso Iterativo e incremental.
- Proceso Centrado en la arquitectura.
- Proceso Guiado por casos de uso.
- Confrontación de riesgos.

2.5.5.1 Las 6 mejores prácticas del desarrollo efectivo

1.- Desarrollar software de manera iterativa.
2.- Administración de requerimientos.
3.- Uso de arquitectura basada en componentes.
4.- Uso de software de modelación visual.
5.- Verificar la calidad de software.
6.- Control de cambios del software.

Desarrollar software de manera iterativa.- Dada la complejidad de los sistemas de software de hoy en día, no es posible desarrollarlo de una manera secuencial, definiendo el problema primero, diseñando la solución completa, construir el software y después probar en producto al final. Un enfoque iterativo es necesario que permita un entendimiento incremental del problema a través de refinamientos sucesivos, y de un crecimiento incremental de una solución efectiva, así como de múltiples iteraciones. Un enfoque iterativo ayuda a atacar el riego a través de procesos demostrables – liberación frecuente de ejecutables que permitan un envolvimiento y retroalimentación continua del usuario final. Porque cada iteración termina con la liberación de un ejecutable, el equipo de desarrollo esta enfocado en producir resultados, y el frecuente monitoreo del estatus ayuda a asegurarse que el proyecto se mantenga en el calendario. Un enfoque iterativo también hace más fácil el acomodar los cambios técnicos en los requerimientos, características o calendario.

Administración de requerimientos.- El RUP describe como elegir, organizar y documentar las funciones y restricciones requeridas. Rastrear y documentar las decisiones, capturar y comunicar fácilmente los requerimientos del negocio. La noción del casos de uso y escenarios preescritos en el procesos han sido probados ser un excelente manera de capturar los requerimientos funcionales y asegurarse que este maneje el diseño, implementación y prueba del software, haciendo mas fácil llenar los requerimientos del usuario final.

Uso de arquitectura basada en componentes.- El RUP soporta desarrollo de software basado en componentes. Los componentes son módulos no triviales, subsistemas que llenan una función limpia. El RUP provee de un enfoque sistemático para definir una arquitectura usando componentes nuevos y existentes. Estos son ensamblados en una arquitectura bien definida, incluso a la medida, o en una infraestructura de componentes como CORBA y COM para los cuales una industria de componentes reutilizables esta emergiendo.

Uso de software de modelación visual.- El proceso muestra como un software de modelación visual es utilizado para capturar la estructura y comportamiento de la arquitectura y los componentes. Esto permite esconder los detalles y escritura del código usando bloques de construcción gráficos, Abstracciones visuales ayudan a comunicarse en diferentes aspectos del software; ver como los elementos del sistema de acoplan conjuntamente. Se asegura que los bloques de construcción son consistentes con el código. Manteniendo consistencia entre el diseño y la implementación. El estándar de la

industria es el Lenguaje Unificado de modelado (UML) creado por Racional Software, el cual es el fundamento para un modelado visual exitoso.

Verificar la calidad de software.- El pobre desempeño y la poca confiabilidad de las aplicaciones son los factores comunes que pueden inhibir dramáticamente la aceptación de las aplicaciones de software de hoy en día. Por lo tanto la calidad debe de ser revisada con respecto a los requerimientos base, en la confiabilidad, funcionalidad, desempeño de la aplicación y del sistema. El RUP asiste en la planeación, diseño, implementación, ejecución y evaluación de este tipo de pruebas. La valoración de calidad es construida en el proceso en todas las actividades, envolviendo a todos los participantes, usando objetivos alcanzables y un criterio que permite que no sea tratada como una tarea o actividad aislada desarrollada por un equipo separado.

Control de cambios del software.- La habilidad de manejar los cambios – asegurarse que cada cambio es aceptado y rastreado- es esencial en un ambiente en el cual los cambios son inevitables. El proceso describe como controlar, rastrear y monitorear los cambios que permitan un desarrollo iterativo exitoso.

También guía en como establecer espacios de trabajo para cada desarrollador, proveyendo aislamiento de los cambios hechos en otros espacios de trabajo y controlando cambios de todos los artefactos del software (ejemplo: Modelos, Código, Documentos, etc.) y une al equipo a trabajar juntos como una sola unidad describiendo como hacer una integración y administración del desarrollo.

La estructura del RUP se muestra en dos dimensiones

El eje horizontal representa el tiempo y muestra los aspectos del ciclo de vida de procesos como su desarrollo.
El eje vertical representa disciplina, la cuál agrupa actividades lógicamente por naturaleza.

La primera dimensión representa el aspecto dinámico del proceso como es promulgado, y es expresado en términos de fases, iteraciones y pequeñas metas (Milestones). La segunda dimensión representa el aspecto estático del proceso: Cómo está descrita en términos de componentes de proceso, disciplina, actividades, flujos de trabajo, artefactos, y documentos.

El ciclo de vida del software esta compuesto de ciclos. Cada ciclo trabaja en una nueva generación de un producto. El RUP divide un ciclo de desarrolló en 4 fases consecutivas como se muestra en la figura 5:

- La Fase de Inicial.
- La fase de Elaboración.
- La fase de Construcción.
- La fase de Transición.

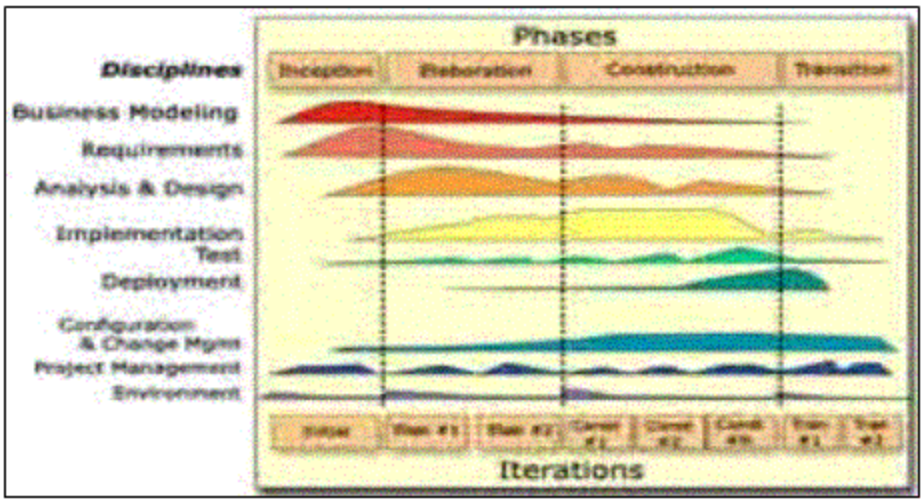


Figura 5 - Esquema de las fases del Proceso Unificado.

Fuente: Racional

Cada fase es concluida con una meta bien definida (milestone) – la cual es un punto en el tiempo y se centra en las decisiones criticas que se deben tomar y que metas clave deben ser alcanzadas.

La Fase de Principio.- Durante la fase de inicio se establece un caso de negocios del sistema y se delimita el alcance del proyecto. Para completar esto, se debe de identificar todas las entidades externas con las cuales el sistema interactuara (actores) y se debe definir la naturaleza de estas interacciones a un alto nivel. Esto involucra identificar todos los usos y describir un poco el significado de ellos. El caso de negocios incluye un criterio de éxito, valoración de riesgos, estimado de recursos necesarios y un plan de la fase mostrando las fechas principales de las metas.

El resultado de esta fase inicial es:
Una visión de documento: Una visión general de los requerimientos centrales del proyecto, las características claves y las restricciones principales.

- Un modelo de uso inicial.
- Un glosario inicial del proyecto.
- Un caso inicial de negocios.
- Una valoración inicial de riesgos.
- Un plan de proyecto, mostrando fases e iteraciones.
- Un modelo de negocios.
- Uno o varios prototipos.

Al término de la fase inicial se encuentra la primera meta del proyecto. La meta de objetivo de ciclo de vida.

Los criterios de evaluación de la fase inicial son:

- Los accionistas deben estar de acuerdo con el alcance de la definición de las estimaciones de costo y calendario.
- El entendimiento de los requerimientos como evidencia de fidelidad de los usos principales.
- Credibilidad de los estimados de costo y calendario, prioridades, riesgos y proceso de desarrollo.
- Gastos actuales contra gastos planeados.

La Fase de Elaboración.- el propósito de la fase de elaboración es analizar el dominio del problema, establecer una sólida arquitectura, desarrollar el plan del proyecto y eliminar los elementos de mayor riego del proyecto. Para lograr estos objetivos, se debe tener una amplia visión del sistema. Las decisiones sobre arquitectura deben ser hechas con un entendimiento de todo el sistema, sus alcances, su función principal y los requerimientos no funcionales, así como los requerimientos de desempeño.

Es fácil entender que la fase de elaboración es la fase mas critica de las 4 fases. Al término de esta fase la "ingeniería dura" es considerada completa y el proyecto experimenta su más importante día en los cálculos; la decisión sobre continuar o no con las fases de construcción o transición. Para muchos proyectos, esto también corresponde a la transición de una operación móvil, ligera, ágil, y de bajo riego una operación altamente costosa, de alto riesgo operación con sustancial inercia. Mientras el proceso debe siempre acomodarse a los cambios, las actividades de la fase de elaboración se aseguran que la arquitectura, los requerimientos y el plan sean lo suficientemente estables, y el riesgo sea lo suficientemente mitigado, así que se puede predecir el costo y el calendario para completar el desarrollo. Conceptualmente, este nivel de fidelidad corresponde al nivel necesario para que una organización contemple costo de la fase.

En la fase de elaboración un prototipo ejecutable de la arquitectura es construido en una o más iteraciones, dependiendo del alcance, el tamaño, el riego y la novedad del proyecto. Este esfuerzo por lo menos deberá manejar los usos críticos identificados en la fase de inicio, la cual típicamente expone la mayoría de los riesgos del proyecto. Mientras un prototipo evolucionado de un componente de producción de calidad es la meta, este no lo excluye del desarrollo de uno o más prototipos exploratorios, a través de prototipos que mitiguen los riesgos específicos como los de diseño, requerimientos intercambiados, estudios de factibilidad de componentes, o la demostración a los inversionistas, clientes y usuarios finales.

Los resultados de la fase de elaboración son:

- Un modelo de usuario (80% por lo menos)
- La captura de requerimientos suplementarios, los no funcionales y cualquier requerimiento que sea no asociado con un caso especifico de uso.
- Una descripción de la arquitectura del software.
- Un prototipo de la arquitectura ejecutable.
- Una lista de riesgos y un caso de negocios revisado.

- Un plan de desarrollo para el proyecto en general, incluyendo mostrando las iteraciones y la evaluación y los criterios de evaluación para cada iteración
- Un manual de usuario preliminar

Meta: Ciclo de vida de la arquitectura

Al final de la fase de elaboración se encuentra la segunda meta importante. La meta de arquitectura del ciclo de vida. En este punto se examinan a detalle los objetivos y alcance del sistema, la selección de la arquitectura y la resolución de los riesgos mayores.

El principal criterio de evaluación de la fase de elaboración envuelve las respuestas a las siguientes preguntas:

- ¿Es la visión del producto estable?
- ¿Es la arquitectura estable?
- ¿La demostración del ejecutable ha mostrado que los elementos de mayor riesgo han sido manejados y resueltos con credibilidad?
- ¿Es el plan de construcción de la fase lo suficientemente detallado y exacto? ¿Esta respaldado con un una base creíble de estimados?
- ¿Están los accionistas de acuerdo que la presente visión puede ser alcanzada si el plan es ejecutado para desarrollar el sistema completo en el contexto de la arquitectura actual?
- ¿Son aceptables los gastos en recursos actuales contra los gastos planeados?

La Fase de Construcción.- Durante la fase de construcción, todos los componentes restantes y las características de las aplicaciones son desarrollados e integrados al producto y todas las características son ampliamente probadas. La fase de construcción es en un sentido un proceso de manufactura, donde el énfasis esta colocado en la administración de recursos y el control de las operaciones para optimizar costos, calendarios y calidad. En este sentido, la administración mental experimenta una transición del desarrollo de la propiedad intelectual durante la fase de inicio y elaboración, al desarrollo de un producto destacable durante las fases de construcción y transición.

El resultado de la fase de construcción es un producto listo para ponerse en las manos del usuario final, y consisten en:

- El producto del software integrado en la adecuada plataforma.
- Los manuales de usuario.
- Una descripción de la revisión actual.

Meta: Capacidad operacional inicial.

Al termino de la fase de construcción se encuentra la tercera mayor meta del proyecto (la meta de capacidad operacional inicial) en este punto, se decidirá si el software, el sitio y los usuarios están listos para ser operacionales sin exponer al proyecto a riesgos mayores. Esta liberación es frecuentemente llamada versión "Beta".

El criterio de evaluación de la fase de construcción involucra la respuesta de las siguientes respuestas.

- ¿Es esta liberación de producto lo suficientemente estable y madura para ser destacable en la comunidad de usuarios?
- ¿Están todos los accionistas listos para la transición en la comunidad de usuarios?
- ¿Siguen siendo aceptables los gastos de recursos gastados actuales contra los gasto de recursos planeados?

La Fase de Transición.- El propósito de la fase de transición es la transición del producto de software a la comunidad de usuarios. Una vez que el producto ha sido dado a al usuario final, usualmente se requiere del desarrollo de nuevos entregables, corrigiendo algunos problemas o el terminar las características que fueron pospuestas.

La fase de transición es introducida cuando le producto es lo suficientemente maduro como para ser entregado al usuario final. Esto típicamente requiere que algunos subconjuntos del sistema hayan sido completados a un nivel aceptable de calidad y que la documentación del usuario esta disponible, así la transición del usuario proveerá de resultados positivos para todas las partes. Esto incluye:

- Una prueba beta que valide que el nuevo sistema cumple con las expectativas del usuario.
- Una operación paralela junto al sistema que se esta remplazando
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y administradores.
- Entrega del producto a los equipos de mercadotecnia, distribución y ventas.

La fase de transición se enfoca en las actividades requeridas para que le software se coloque en las manos de los usuarios. Típicamente esta fase incluye muchas iteraciones, incluyendo versiones beta, versiones con mejoras generales o con errores corregidos. Un esfuerzo considerable es hecho para desarrollar documentación orientada al usuario, entrenamiento de usuarios y reaccionar a la retroalimentación de usuario. En este punto del ciclo de vida del producto, la retroalimentación del usuario debe ser tomada en cuenta para afinar el producto, configurarlo e instalarlo.

Los objetivos primarios de la fase de transición incluyen:

- Alcanzar el soporte de usuario
- Lograr la concurrencia de los accionistas que el desarrollo esta completo y es consistente con el criterio de evaluación de la visión.
- Logra que los lineamientos de producto final tanto en efectividad de tiempo y costo como practica.

Al final de la fase de transición se encuentra la cuarta meta en importancia del proyecto. La meta de liberación de producto. En este punto, se decide si los objetivos fueron cumplidos o si se debe de empezar otro ciclo de desarrollo, en algunos casos, estas metas pueden coincidir con el fin de la fase de inicio del siguiente ciclo.

El principal criterio de evaluación de la fase de transición involucra la respuesta de las siguientes preguntas.

- ¿Esta el usuario satisfecho?
- ¿Son aceptables los gastos de recursos actuales contra los gastos planeados?

Beneficios de un enfoque iterativo.- comparado con el proceso tradicional de cascada, el proceso iterativo tienen las siguientes ventajas:

- Los riesgos son mitigados en las etapas tempranas.
- La administración de cambios es más fácil.
- Alto nivel de reutilización de código.
- El equipo del proyecto puede aprender a lo largo del camino.
- Mejor calidad general del proyecto.

2.5.6 Problemas de las metodologías convencionales de software

Royce (1998) haciendo referencia a una publicación de Barry Boehm titulada “La lista de las 10 métricas más importantes de la industria del software”, muestra una buena caracterización objetiva del estado del desarrollo de software. (Donde se muestra muy poca evidencia de los cambios significantes en la pasada década). A través de estas métricas describe algunas de las relaciones económicas que resultan de la administración convencional de software practicada durante los pasados 30 años:

1.- Encontrar y arreglar problemas después de la liberación cuesta 100 veces más que encontrar y arreglar los problemas en las etapas tempranas de diseño.

2.- Se puede contraer los calendarios del desarrollo de software en un 25% nominal, pero no más.

3.- Por cada dólar que se gasta en el desarrollo, se gastaran 2 dólares adicionales en el mantenimiento.

4.- El desarrollo de software y el costo del mantenimiento están en función del número de líneas de código fuente.

5.- Las variantes en la gente cuenta en las grandes diferencias en la productividad del software.

6.- El porcentaje global entre los costos de software y hardware sigue creciendo. En 1955 era de 15:85, y en 1985 era de 85:15.
7.-Solamente el 15 % de los esfuerzos de desarrollo están enfocados a la programación.

8.- Los sistemas de software y sus productos típicamente cuestan 3 veces más por línea de código fuente que un programa de software individual. Los productos sistemas de software cuestan 9 veces más.

9.- Las Pruebas de seguimiento cachan el 60% de los errores.

10.- El 80% de la contribución en la programación viene del 20% de los programadores.

Estas relaciones proveen buenas comparaciones que ayudan a evaluar las mejoras de procesos y tecnologías. Ellas representan reglas que objetivamente caracteriza el desempeño de la administración convencional de software y las tecnologías convencionales.

2.5.6 Características Económicas del Software

En los años 80's y los 90's la industria del software ha madurado y se ha transformado a una disciplina de ingeniería. Aunque muchos proyectos de software sigan siendo principalmente de una investigaron intensiva, dominada por la creatividad humana y las economías de escala.

La mayor parte del costo de administrar los proyectos puede estar en función de los 5 parámetros siguientes: (Royce1998)

1.- El Tamaño del producto final, el cual es típicamente cuantificado en términos del número de líneas en código fuente o el número de puntos funcionales que se requieren para desarrollar la funcionalidad requerida.

2.- Los Procesos usados para producir el producto final, en particular la habilidad de los procesos que eviten adicionar actividades sin valor (re-trabajo, retrazo, burocracia, sobre comunicación).

3.- Las Capacidades del personal de ingeniería de software, particularmente su experiencia con los asuntos de computación y asuntos de aplicaciones de dominio del proyecto.

4.- El Ambiente, en cual se desarrollaron están hechas las herramientas y técnicas disponibles para soportar eficientemente el desarrollo de software y para automatizar procesos.

5.-La Calidad requerida del producto, incluyendo características, desempeño, confiabilidad y la adaptabilidad.

En la Tabla 3 se muestra la manera como se han desarrollado tecnologías, en un esfuerzo por mejorar el desempeño de procesos y enfoques de administración encaminados a mejorar las características económicas del software (Royce 1998).

Parámetros de Software	Tendencia
Tamaño Abstracción Tecnologías de desarrollo basadas en componentes	Lenguajes de alto Orden (C++, Visual Basic, etc.) Orientado a Objetos (Análisis, diseño , programación) Re-uso Componentes Comerciales
Procesos Métodos y Técnicas	Desarrollo Iterativo Modelos de Madurez de procesos Arquitectura de Primer desarrollo Reformas de Adquisición
Personal Factores de la Gente	Entrenamiento y habilidades del personal de desarrollo
Ambiente Tecnologías y herramientas de Automatización	Herramientas Integradas (modelación visual, copelación, editores, administración de cambios, etc.) Sistemas Abiertos. Desempeño de las Plataformas de Hardware. Automatización de código, documentación , pruebas y análisis
Calidad Desempeño, Fiabilidad, Exactitud	Desempeño de las Plataformas de Hardware. Control de Calidad Estadística

Tabla 3 - Tecnologías que refuerzan el desempeño de los procesos.

Fuente: Royce (1998)

2.6. Administración de Proyectos

La administración de proyectos es un proceso que abarca el ciclo de vida completo de un proyecto, de inicio a fin. Los puntos principales son planear, ejecutar y controlar todos los recursos, tareas y actividades necesarios para completar el proyecto. La administración de proyectos no es una actividad aislada, son los esfuerzos como equipo de todo el equipo de desarrollo.

La administración de proyectos como lo comenta Williams (1996) es la aplicación del conocimiento, habilidades, herramientas y técnicas en las actividades del proyecto, con el fin de encontrar o superar las expectativas de los inversionistas y las expectativas del proyecto. Esto será posible solamente si se crea un balance entre la demanda de: metas, tiempo, costo y calidad.

En fin, la administración de proyectos es acerca de gente, procesos y como el trabajo esta siendo desempeñado. Las 4 p de la administración de proyectos son: gente desempeñando procesos perfectos (People, Performing Perfect Process)

2.6.1 El Valor de la Administración de proyectos

Implementar la cultura de administración de proyectos agrega un valor significante a las organizaciones. Esta conclusión es el resultado de una encuesta entre más de 100 administradores de proyectos a nivel gerencial, practicada por el centro de prácticas de negocios. En esta investigación de consultoría y entrenamiento organizacional, más del 94% de los entrevistados afirmaron que implementar los procesos de administración de proyectos agrega valor a sus organizaciones. Las organizaciones mencionan que ha habido significantes mejoras en mediciones financieras, mediciones de clientes, mediciones de proyectos y mediciones de crecimiento. Las organizaciones de todos los tamaños en todas las industrias reportan mejoras. (Valueofpm)

Aunque muchos piensen que un gerente de proyecto es alguien que se encarga de que se cumplan las fechas de las actividades con el fin de asegurar la entrega de sistema a tiempo, Los gerentes de proyecto de hoy en día, son responsabilidades de liderazgo del equipo de desarrollo.

Progresivamente, estas posiciones requieren una gran variedad de habilidades. Esto es especialmente cierto en la industria de tecnologías de Información, dónde la innovación y el cambio ocurren más rápido que en cualquier otra industria y hasta quizá que en cualquier otro tiempo en historia. En un ambiente rápidamente cambiante, en el cual las personas son esperadas para efectuar la entrega resulta más rápido, en el costo inferior, y con mejor calidad, el liderazgo juega un papel crítico en el desempeño de equipos motivadores y de optimización.

2.7 Principios de la Administración Moderna de Software

Los 10 Principales principios de la administración moderna de software (proceso Iterativo)

1.- Enfocar el proceso primeramente en la arquitectura.
2.- Atacar riesgos en etapas iniciales con un ciclo de vida interactivo.
3.- Enfatizar en el desarrollo basado en componentes.
4.- Establecer un cambio de administración del ambiente.
5.- Fomentar la libertad de cambio con herramientas para la ingeniería de viaje redondo.
6.- Usar rigurosamente, diseño basado en modelos.
7.- Instrumentar el proceso para un control de la calidad de los objetivos.
8.- Usar una valoración basada en demostraciones de artefactos intermedios.
9.- Planear entregas con envolvimiento de niveles de detalle.
10.- Establecer procesos escalables y configurables.

2.7.1 Practicas modernas de Software

Desde el año 2000 la filosofía del RUP esta arraigada en el uso de procesos predecibles y administrables, integrando ambientes de automatización, y su mayor parte (70%) serán componentes estandarizados. Quizás unos pocos (30%) tendrá la necesitaran ser construidos a la medida. Con los avances de la tecnología informática y ambientes integrados de producción, estos sistemas basados en componentes pueden ser producidos muy rápidamente. (Royce 1998)

Las tecnologías de ambiente de automatización, reducción de tamaño y mejoramiento de procesos no son independientes unas de otras. En cada una la clave crecimiento complementario en todas las tecnologías, por ejemplo los avances del proceso no pueden ser usados exitosamente sin nuevos componentes tecnológicos y ambientes de producción integrados. (Royce 1998)

El esquema de procesos de la administración de software
Estandarizar un proceso común es valioso para una organización del software, y hay una amplio espectro de implementaciones, es así como el esquema de proceso recomendado por Royce (1998) abarca un puñado de estándares específicos; fases del ciclo de vida, artefactos del ciclo de vida, flujos de trabajo del ciclo de vida, y puntos de chequeo del ciclo de vida. Estos elementos son factores para la transición de un enfoque convencional al enfoque iterativo, enfoque de línea de negocios.

2.7.2 Reducir el Tamaño del software

La manera más significativa de mejorar las utilidades y el retorno de la inversión, usualmente es producir un producto que alcance las metas del diseño con el monto mínimo de material humano generado.

El desarrollo basado en componentes, es introducido como la forma general para reducir el tamaño de código fuente generado por humanos necesario para alcanzar la solución del software. La reutilización, la tecnología orientada a objetos, la producción automática de

código, y uso de lenguajes del alto nivel están enfocados a alcanzar un sistema dado con menos líneas de código fuente generadas por humanos. Esta reducción de tamaño es la principal motivación detrás de las mejoras en lenguajes del alto nivel(C++, Ada 95, Java , Visual Basic y tros lenguajes de cuarta generación), generadores automáticos de código(herramientas CASE , Herramientas de modelación Visual, Constructores GUI) la reutilización de componentes comerciales(sistemas operativos, administradores de bases de datos, Middleware y redes) y las tecnologías Orientadas a Objetos (Lenguaje Unificado de Modelación, Herramientas de Modelación , Arquitectura).

Un cometario que hace Royce (1998) es "El código que no esta escrito, no fue necesario ser desarrollado y no puede marcar error."

Algo experimentado por muchas equipos de proyectos, es que las tecnologías maduras y confiables de reducción de tamaño son extremadamente poderosas en producir beneficios económicos en cambio las tecnologías inmaduras de reducción de tamaño, reducirá el tamaño de desarrollo pero se requerirá de mas inversión para alcanzar los niveles necesarios de calidad y desempeño por los impactos negativos en el desempeño global del proyecto.

2.7.2.1 Métodos Orientados a Objetos

Algunos estudios han concluido que los lenguajes de programación orientada a objetos aparece para beneficiar a ambos, la producción de software y la calidad del software(Jones 1994), pero un beneficio económico todavía no ha sido demostrado porque costoso paso de entrenar en los métodos de diseño orientados a objetos como El lenguaje Unificado de Modelación (UML).

Booch (1997) también describe las 5 características que poseen los proyectos exitosos que utilizan la metodología orientada a objetos.

- Un implacable enfoque en el desarrollo de un sistema que provea de una colección bien entendida de características esenciales mínimas.
- La existencia de una cultura que esta centrada en resultados, motiva comunicación y sin asustarse del fracaso.
- El uso efectivo del modelado orientado a objetos.
- La existencia de una visión fuerte de arquitectura.
- La aplicación de una bien manejada de ciclo de vida desarrollo iterativo e incremental.
- Herramientas de la Administración moderna de Software.

2.7.2.2 Modelación Visual

El lenguaje de modelado Unificado (UML) es el lenguaje estándar en la industria para especificar, visualizar y documentar los objetos en los sistemas de software. Este simplifica el proceso completo del diseño de software, haciendo un plano "Blue Print" de la construcción.

Para profesionales de base de datos, el UML puede ser usado para el diseño de la base de datos. Usar el UML para el diseño de la base de datos permite a los equipos de negocios y aplicaciones compartir un lenguaje común, y comunicarse con el equipo de base de datos.

El propósito es desarrollar el modelo para fortalecimiento en el sistema industrial de software y que se anticipe a la construcción de un plano para grandes proyectos. Los buenos modelos son esenciales para comunicar a través de los equipos del proyecto y para asegurar una firmeza arquitectónica. En la medida que la complejidad de los sistemas se incrementa, vemos la importancia de buenas técnicas de modelación. Hay muchos factores adicionales en el éxito de un proyecto, pero tener un riguroso estándar modelación es un factor esencial

2.7.2.3 Componentes Comerciales

Un enfoque común siendo perseguido hoy en día en muchos dominios es maximizar la integración de componentes comerciales. Mientras el uso de componentes comerciales esta ciertamente deseable como medio de reducción de desarrollo a la medida, no ha sido probado que sea francamente en la practica. En la tabla 4 se muestra las ventajas y desventajas del uso de componentes comerciales. El principal mensaje aquí es que la decisión de usar componentes comerciales debe ser hecha tempranamente en el ciclo de vida como parte del diseño de la arquitectura.

2.7.3 Mejorar los Procesos de Software

El objetivo de mejorar los procesos es maximizar la localización de los recursos, con el fin de minimizar el impacto de las actividades globales en los recursos como el personal, computadoras y el calendario (Royce 1996).

Las actividades productivas consisten en actividades productivas y de actividades globales. Las actividades productivas resultan de procesos intangibles a través del producto final. Para el software estas actividades incluyen prototipos, modelación, codificación y documentación del usuario. Las actividades globales que tiene un impacto en el producto final son requeridas en la planeación, documentación, monitoreo de progreso, valoración de riesgos, integración, pruebas, minimizar el desperdicio y el re-trabajo, administración de personal, capacitación de personal, administración de negocios y otras tareas.

Enfoque	Ventajas	Desventajas
Componentes Comerciales	Costo de licencias predecible Ampliamente Usadas, tecnología madura Disponible inmediatamente Organización con Soporte dedicado Independencia	Actualizaciones frecuentes Tarifas de mantenimiento recurrentes Dependencia del vendedor Sacrificios de eficiencia en tiempo de ejecución Características innecesarias que consumen recursos extras Incompatibilidades de múltiples vendedores Frecuentemente inadecuada estabilidad y confiabilidad No se tiene control sobre actualizaciones y mantenimientos La integración no es siempre trivial
Desarrollo a la medida	Libertad de Cambio Completa Pequeñas, frecuentemente implementación simple Ofrecen mejor desempeño Control de desarrollo y Realce	Desarrollo Costoso e impredecible Fecha de disponibilidad Impredecible Modelo de mantenimiento no definido Frecuentemente inmaduros y frágiles Dependencia de una sola plataforma Basado en las recursos de los expertos

Tabla 4 - Ventajas y desventajas de usas Componentes Comerciales

Fuente: Royce (1998)

La calidad del proceso de software afecta fuertemente los esfuerzos requeridos y el calendario para producir el producto de software deseado. En la práctica la diferencia entre un buen proceso y uno malo, afectara los costos estimados globales en un 50% a un 100%, y la reducción en esfuerzos mejorara el calendario global. Mas aun, un mejor proceso puede tener un mejor efecto en reducir el tiempo que tomara al equipo alcanzar la visión de producto con la calidad requerida.

2.7.3.1 El Modelo de Capacidades de Madurez (CMM)

El Capability Maturity Model (CMM) fue desarrollado por el instituto de ingeniería de software (SEI) en la universidad Carnegie Mellon en Pittsburg (Paulk, 1995). Define las áreas del proceso del software organizados en niveles de madurez (5 niveles de madurez) a través de progresar en etapas. El CMM esta basado en las mejores practicas del gobierno, industrias y militares de los Estados Unidos de América para el desarrollo de software, tomando los anos de experiencia de estos departamentos. Es un modelo

genérico que provee el marco de trabajo para la mejora de procesos en el momento de producir software por empresas dedicadas a la construcción de sistemas computacionales.

Los cinco niveles de madurez del CMM son los siguientes:

1.- El primer nivel de madurez es ad hoc e informal. Hay un medio ambiente de desarrollo y mantenimiento del software inestable.

2.- El segundo nivel, repetible, introduce las prácticas de la administración como un proyecto de planeación, administración de requerimientos y administración de la configuración. Las áreas claves de procesos en el nivel dos proveen una base para realizar el nivel tres.

3.- Definido comienza con la incorporación de prácticas técnicas con las prácticas de administración. Estas practicas son compartidas entre la organización y empiezan a ser institucionalizadas.

4.- Administrado, introduce mas administración de los procesos cuantitativamente (habilidad para planificar, encontrar problemas etc.) por lo que los procesos deben ser medidos.

5.-Optimización, Introduce la detección y prevención de problemas. El proceso es auto optimizado basándose en la retroalimentación de la medición de los procesos.

Aplicar el modelo de CMM en una empresa tarda de 12 meses a 3 años dependiendo del caso pero se obtienen beneficios como:

Mejorar la predicción de entrega y del presupuesto, así como el tiempo de los ciclos para llegar al mercado, con la flexibilidad y estabilidad.

Aumentar a productividad y la calidad de la medida por los errores y defectos
Incrementar la satisfacción del cliente, la moral del empleado, además de incrementar el retorno de inversión bajando costos.

El uso de este modelo se ha hecho fundamental para las industrias, como en la india, donde se abaten en varias decenas de veces los costos por línea de código , empezando por el tiempo dedicado al pasar de una línea de código por hora-hombre a menos de 20 minutos-hombre para la misma línea.

2.7.4 Mejorar la Efectividad del Personal

Royce (1998) también comenta que es muy conocido que las diferencias en el personal influye es un factor importante en los cambios de la productividad.

El trabajo de equipo es mucho más importante que la suma de los individuos. Con los equipos de software, el administrador de proyecto necesita configurar un balance de talentos sólidos con gente con altas habilidades en la posiciones de apalancamiento.

2.7.5 Alcanzar la Calidad Requerida

Royce (1998) comenta que muchas de las mejores prácticas de software que hoy en día son aceptadas esta derivadas del desarrollo de procesos y tecnologías sumariadas en este capitulo. Estas prácticas tienen un impacto adicional, para mejorar el costo beneficio. A continuación se describen algunas de esas mejoras de calidad.

Enfocarse en manejar los requerimientos y casos críticos en las etapas iniciales del ciclo de vida. Enfocándose en la atención de plenitud de los requisitos y el seguimiento en las ultimas etapas del ciclo. Y enfocarse a lo largo de las etapas del ciclo de vida en el balance la evolución de requerimientos, la evolución del diseño y la evolución de la planeación.

Usar métricas e indicadores para medir el progreso y la calidad de una arquitectura como si esta envolviera un prototipo de alto nivel dentro de un producto complaciente.

Proveer ambiente del ciclo de vida integrado que soporte una temprana y continua configuración del control, administración de cambios, métodos de diseño riguroso, automatización de documentos y automatización de pruebas.

Usar modelación visual y lenguajes de alto nivel que soporten el control de la arquitectura, la abstracción, programación confiable, el re-uso, y auto documentación.
Continuo entendimiento dentro del tema de desempeño a través de evaluaciones basadas en demostraciones.

2.8 .La Estructura de la Metodología del RUP

En esta parte se describirá la metodología de ciclo de vida que se esta analizando para poder familiarizarse con la metodología, así mismo se describen los beneficios que se obtienen al implantar cada uno de los procesos o actividades de esta metodología, con el objetivo de tener un punto de comparación entre las metodologías existentes y esta nueva forma de desarrollar y administrar proyectos de software de una manera exitosa...

Royce (1998) en su libro Administración de proyectos de software un enfoque unificado, nos describe una metodología de administración de proyectos, siguiendo los mejores estándares de la industria que a continuación se enumeran:

- Las fases del ciclo de vida:
- Los artefactos del proceso.
- Arquitecturas basadas en modelos.
- Flujo de trabajo del proceso.
- Puntos de chequeo del proceso.

Estos estándares se detallaran a continuación….

2.8.1 Las Fases del ciclo de vida de un Proyecto de Software

Como ya se ha observado en los capítulos anteriores las fases son las siguientes:

- La Fase Inicial
- La Fase de Elaboración
- La Fase de Construcción
- La Fase de Transición

2.8.2 Artefactos del proceso

Los proyectos convencionales están enfocados en el desarrollo de artefactos de software de manera secuencial: construcción de requerimientos, construcción de un modelo detectable de los requerimientos, construir una implementación acorde al diseño del modelo y por ultimo compilar y probar la implementación para el desarrollo.

Este modelo puede funcionar para proyectos a escala pequeña, si embargo este modelo no funciona muy bien para la mayoría de los sistemas de hoy en día, en los cuales la complejidad de los sistemas resultan numerosos riesgos y una serie de relaciones en las que no pueden usarse una transformación secuencial simple de manera efectiva.

En un enfoque iterativo los artefactos del proceso están organizados en cinco conjuntos: Administración, Requerimientos, Diseño Implementación y Desarrollo.
Los artefactos de administración capturan la información necesaria para sincronizar las expectativas de todas las personas involucradas en el proceso (stakeholders).
Los artefactos de requerimientos, diseño, implementación y desarrollo son capturados en notaciones rigurosas que soportan análisis y navegación automatizada.

Los artefactos están organizados en conjuntos, cada conjunto abarca artefactos relacionados que son similares y cuentan con un formato de representación uniforme (C++, visual Basic, java, plantilla de documento estándar, un modelo de UML).

Los artefactos del ciclo de vida del software están organizados en cinco conjuntos que están plenamente identificados por el lenguaje subyacente del conjunto: administración, requerimientos, (texto organizado y modelos del problema de espacio), diseño (modelos de solución de espacio) implementación (lenguajes de programación entendidos por el humano y archivos fuentes asociados) y el desarrollo (lenguajes de procesados por maquinas y archivos asociados).

El surgimiento de rigurosas y más poderosas notaciones de ingeniería para artefactos de requerimientos y diseño que soporten desarrollo de primera arquitectura ha sido un gran avance en la tecnología. En particular el lenguaje unificado de modelado (UML) a envuelto un formato de representaciones, llamados modelos visuales con una sintaxis muy bien especificada para los artefactos de requerimientos y de diseño. El lenguaje de modelación visual usando UML es una notación primitiva para los artefactos iniciales del ciclo de vida. Los conjuntos de artefactos son mostrados en la figura 6.

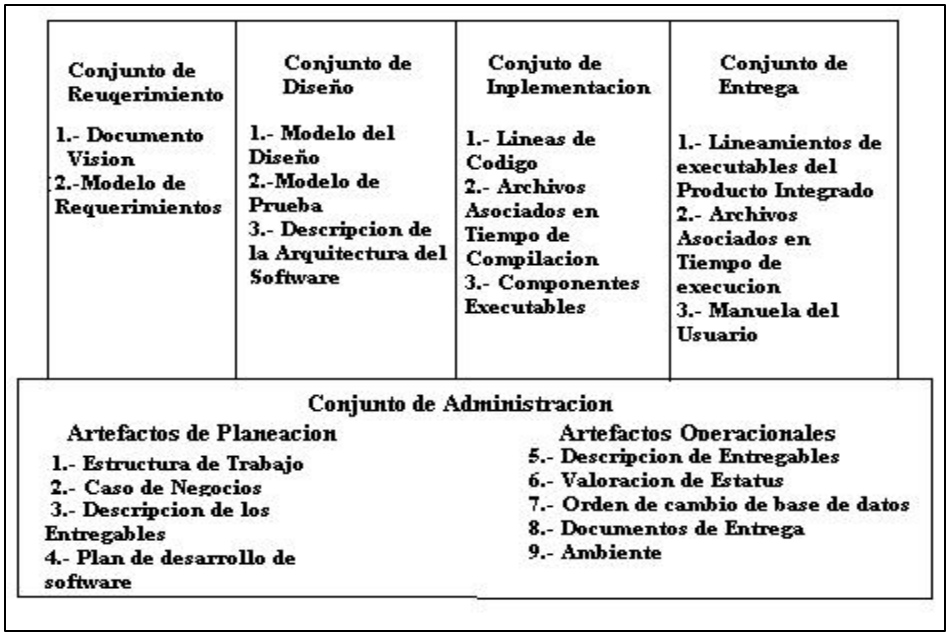


Figura 6 - Conjunto de Artefactos.

Fuente (Royce, 1998)

La figura 7 muestra como cada conjunto de artefactos esta enfocado a una fase del ciclo de vida, donde en esa fase tiene un desarrollo predominante y en las demás fases toma un rol de chequeo y balance.

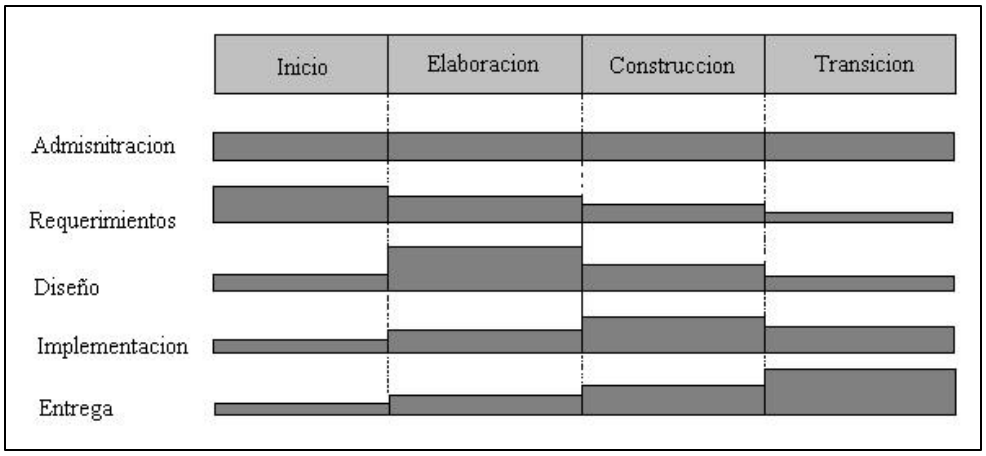


Figura 7 - Ciclo de vida enfocado en el conjunto de artefactos.

Fuente (Royce, 1998)

Artefactos Administrativos

El conjunto de artefactos de administración captura los artefactos asociados con el proceso de planeación y ejecución. Estos artefactos usan un proceso a la medida, incluyendo texto y gráficos o cualquier representación que sea requerida para capturar los contratos del personal del proyecto(administrador del proyecto, arquitectos, usuarios,

desarrolladores, probadores, administradores) a través de los accionistas (fondo, autoridad, usuario, administrador del proyecto, administrador de la organización y la agencia reguladora) y entre el personal del proyecto y los accionistas, los artefactos específicos de este conjunto son el WBS(activity break down structure) , un caso de negocios (costo, calendario, expectativas de ganancias), las especificaciones de los entregables(alcance, plan , objetivos para los entregables bases), el plan de desarrollo de software (instancia del proceso del proyecto), descripción de los resultados (resultados los entregables base),estatus de valoración (valoración periódica del progreso del proyecto).

Las ordenes de cambio del software (descripción de cambio de manera discreta), los documentos del desarrollo (cutover plan, cursos de entrenamiento, kit de ventas), y el ambiente (herramientas de hardware y software, automatización de procesos, documentación, entrenamiento colateral necesario para soportar la ejecución del proceso descrito en el plan de desarrollo de software y la producción de artefactos de ingeniería.

2.8.2.1 WBS

Un WBS es el vehículo para presupuestar y colectar los costos .para monitorear y controlar el desempeño financiero del proyecto, el administrador del proyecto debe adentrarse en los costos del proyecto y como son gastados. Las lecciones aprendidas numerosos proyectos menos exitosos han demostrado que si el WBS es estructurado de una manera inapropiada, este puede manejar el diseño y la estructura del producto en una dirección incorrecta.

2.8.2.2 Caso de negocios

El artefacto de plan de negocios provee toda la información necesaria para determinar si el proyecto es una mala invasión, este detalla las ganancias esperadas, el costo esperado, los planes técnicos y administrativos, le respaldo de datos necesarios para demostrar el riesgo y el realismo de los planes.
El pan de negocios debe ser implementado en una propuesta a gran escala con múltiples volúmenes información. En una esfuerzo a escala pequeña para productos comerciales, debe ser implementado en plan escrito con un una hoja de balance. El propósito principal es transformar las visión en términos económicos, así que la organización pueda ser una exacta valoración de
ROI. El pronostico financiero es revolucionario, actualizado con mas pronósticos como en la ciclo de vida del proceso.

2.8.2.3 Descripción de Entregables

El documento de la descripción de entregables describe los resultados de cada entregable, incluyendo el desempeño a través de cada criterio de evaluación en la correspondiente especificación de entregable.

2.8.2.4 Especificaciones de los Entregables

Los criterios de evaluación del alcance, plan y los objetivos para cada lineamiento entregable estas derivados de el enunciado de la visón, así como otros fuentes (análisis de compra-venta, administración de riesgos, consideraciones arquitectura, restricciones de implementación.

2.8.2.5 Plan de Desarrollo de Software

El plan de desarrollo de software elabora el enfoque del proceso con un plan completamente detallado: este es el documento de definición para el preso del proyecto. este debe acceder, cumplir con los estándares de la organización, desarrollarse largamente con el diseño y los requerimientos. Los dos indicadores de un útil SPD son la actualización periódica (si esta destacado) y el entendimiento de la aceptación de los administradores y calidad de proyecto)

2.8.2.6 Valoración de Estatus

La valoración de estatus provee retratos periódicos de la salud y el estatus del proyecto, incluyendo la valoración de riesgos por parte del administrador de proyecto. Indicadores de calidad e indicadores administrativos. La valoración de estatus periódica provee un mecanismo crítico para administrar las expectativas de todos a través del ciclo de vida, para dirigir, comunicar y resolver los asuntos administrativos .los asuntos técnicos, los riesgos del proyecto.
La valoración típica de estas debe incluir una revisión de los recursos, datos financieros personales (costos y ganancias), los 10 principales riesgos, los procesos técnicos, los principales retos y sus resultados, y por ultimo el alcance total del proyecto.

2.8.2.7 Administración de la Configuración y Cambios

Administración de cambios es uno de los principios fundamentales de un proceso de desarrollo iterativo. Con una gran libertad de cambio, un proyecto puede interactuar más productividad, esta flexibilidad incrementa el contenido, la calidad y el número de iteraciones que el proyecto puede alcanzar con el calendario dado.

La administración de cambios trae beneficios como (Rojas 2003):

• Permitir, controlar y monitorear cambios para habilitar un desarrollo iterativo.
• Controlar todos los artefactos de software - modelos, código, documentos, etc.
• Administrar todas las versiones, con integración automática a los cambios realizados al software.
• Establecer espacios de trabajo seguro y aislado para cada desarrollador.
• Contar con métricas de estado y avance.

En pocas palabras,"Saber qué está pasando en el equipo y en el proyecto". (Rojas 2003)

2.8.2.8 Desarrollo

El documento del desarrollo puede ser llevado a cabo en varias formas. Dependiendo del proyecto, este puede incluir varios subconjuntos de documentos para hacer una transición de un producto a un estatus operacional. En grandes esfuerzos contractuales de los cuales el sistema es liberado para un organización mantenida separadamente, los artefactos de de desarrollo incluirán los manuales de operaciones de los sistemas de computo, los manuales de instalación, los planes y procedimientos para quitar el sistema heredado, el examinar los centros de computo y así sucesivamente. Para los productos de software comerciales, los artefactos de desarrollo deben de incluir planes de mercadeo, conjuntos de ventas y cursos de entrenamiento.

2.8.2.9 Administración de secuencias de artefactos

En cada fase del ciclo de vida, nuevos artefactos son producidos y artefactos previamente desarrollados son actualizados para incorporar las lecciones aprendidas y para capturar soluciones con mas profundidad y anchura.

Artefactos de ingeniería

La mayoría de los artefactos son capturados en notaciones rigurosamente de ingeniería como el UML, los lenguajes de programación o códigos ejecutables maquina.

2.8.2.10 Documento de visión

El documento de Visión provee una visión completa del sistema de software bajo desarrollo y soporta el contacto entre el fondeo, la autoridad y el desarrollo de organización.

Cada proyecto necesita la base para capturar las expectaciones de los accionistas. La visión del proyecto

Este documento de visión es escrito desde la perspectiva del usuario, enfocándose en las características esenciales del sistema y a unos niveles aceptables de calidad. El documento visión debe contener al menos dos apéndices. El primer apéndice debe contener el concepto operacional usando caso de uso (una modelo visual y un artefacto separado) el segundo apéndice debe describir el riesgo de cambio inherente en el enunciado de visión. Para guiar a unos esfuerzos efectivos.

2.8.2.11 Descripción de la Arquitectura

La descripción de la arquitectura provee una vista organizada de la arquitectura del software bajo desarrollo. Esto es extraído del diseño del modelo e incluye vistas del diseño, implementación y el despliegue suficientes para entender como el concepto operacional de los requerimientos que serán alcanzados.

2.8.2.12 Manual del Usuario el Software

El manual del software provee al usuario la documentación referente necesaria para soportar la entrega del software.

El manual de usuario debe incluir los procesos de instalación, las guías y procedimientos de uso, las restricciones operacionales y una descripción de la interfase del usuario por lo menos, para productos de software con interfase de usuario, este manual de ser desarrollado tempranamente en el ciclo de vida porque esto es un mecanismo de comunicación y establece un conjunto importante de requerimientos. El manual del usuario debe ser escrito por miembros del equipo de pruebas, quienes entienden más la perspectiva del usuario.

2.8.3 Arquitecturas de Software basada en modelos

La arquitectura del software es el diseño centra del problema de un sistema de software complejo. En el mismo sentido esta arquitectura es el diseño central del problema para la edificación de un sistema complejo. Para una efectiva administración del proyecto es son necesarias administrar varias disciplinas las cuales son: planeación, organización, automatización y control del proyecto.

La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Los casos de uso guían el desarrollo de la arquitectura y la arquitectura se realimenta en los casos de uso, los dos juntos permiten definir, gestionar y desarrollar adecuadamente el software.

El producto técnico más crítico de un proyecto de software es la arquitectura: la infraestructura, el control y las interfases de los datos que permiten a los componentes del software a cooperar como un sistema y los ingenieros de software a cooperar eficientemente como un equipo. Estableciendo una acertada y precisa comunicación a través de las personas del equipo es el eterno problema en cualquier organización.

Desde una perspectiva administrativa, hay tres aspectos diferentes de una arquitectura.

1.- Una arquitectura (el concepto de diseño intangible) es el diseño del sistema de software, como opuesto al diseño de componentes, esto incluye toda la ingeniería necesaria para especificar una cuenta completa de materiales.

2.- Los linimientos de la arquitectura (los artefactos tangibles) es el diseño de la pieza de información a través de los artefactos de ingeniería suficientes de satisfacer a los accionistas que su visión puede ser alcanzada con los parámetros de el caso de negocios (costo, ganancia, tiempo tecnología y gente).

3.- Una descripción de la arquitectura (una representación entendible por le humeo de una arquitectura, la cual es una de los componentes de los lineamientos de la arquitectura) es el conjunto organizado de información extraída del modelo de diseño, esto incluye:

La importancia de la arquitectura del software representa una liga cercana a las metas del proceso de desarrollo moderno de software, la cual es descrita a continuación

Alcanzar una arquitecta estable representa un meta significativa del proyecto al cual la decisiones de compra/hacer tiene que ser resueltas.

La arquitectura y los procesos encapsulan mucha de la importancia (el pago y alto riesgo en comunicaciones a través de los individuos, equipos, organizaciones y accionistas).
Una pobre arquitectura y proceso inmaduro son las razones dadas frecuentemente como motivo del fracaso del proyecto.

El desarrollo de la arquitectura y la definición de procesos son pasos intelectuales a la solución sin la violación de las restricciones, ellos requieren innovaciones humanas y que no puedan ser automatizadas.

Cuando la arquitectura se diseña y construye en base a componentes, se obtienen los siguientes beneficios (Rojas 2003):

- Reutilización de componentes.
- Flexibilidad al cambio.
- Facilidad para desarrollar elementos en paralelo.

2.8.4 Flujos de trabajo del proceso

Las actividades del proceso están organizadas en siete flujos de trabajo principales: administración, ambiente, requerimientos, diseño implementación valoración y despliegue.

En etapas anteriores se introdujo el ciclo de vida del proceso y los conjuntos. El macro proceso comprende fases e iteraciones discretas.

El próximo nivel de descripción de procesos son lo micro procesos o flujos de trabajo, que producen los artefactos. El flujo de trabajo está dirigido a los artefactos del producto descritos anteriormente. Hay siete flujos de trabajo principales.

Flujo de trabajo de administración: Controlar los procesos y asegurarse que se satisfagan las condiciones para todos los accionistas.

Ambiente de flujo de trabajo: Automatización de procesos y envolver el ambiente de mantenimiento.

Requerimientos de flujos de trabajo: Analizar el problema de espacio y envolver los artefactos de requerimientos.

Diseño de flujos de trabajo: Modelación de la solución que envuelva a la arquitectura y el diseño de artefactos.

Implementación de flujo de trabajo: Programación de componentes y envolver la implementación y el despliegue de artefactos.

Valoración del flujo de trabajo: Valoración de la tendencia en procesos y calidad del producto.

Implementación de flujo de trabajo: Programación de componentes y envolver la implementación y el despliegue de artefactos.

Valoración del flujo de trabajo: Valoración de la tendencia en procesos y calidad del producto.

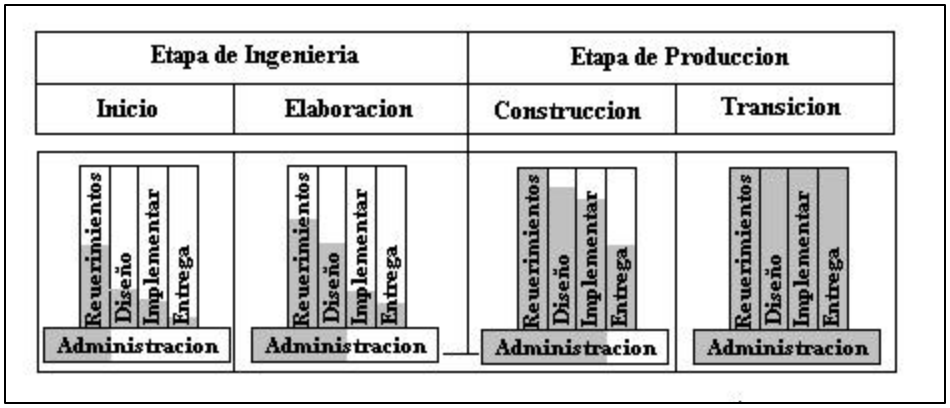


Figura 8 - Niveles de Actividad entre las Fases del ciclo de vida.

Fuente (Royce, 1998)

2.8.5 Puntos de chequeo del proceso

Esto es siempre importante, el tener metas visibles en el ciclo de vida donde varios socios de negocios conozcan, confronten y discutan, el progreso los planes.

El propósito de estos eventos es normalmente demostrar como el proyecto es bien desempeñado, pero también alcanzar los siguientes:

- Sincronizar las expectativas de los socios de negocios y alcanzar una concurrencia en tres perspectivas envolventes: los requerimientos, el diseño y el plan.
- Sincronizar los artefactos relacionados a un estado balanceado y consistente.
- Identificar los riesgos importantes, activos y condiciones fuera de control
- Desempeñar una valoración global para el ciclo de vida completo, no solamente la situación actual de una perspectiva individual inmediata del proyecto.

Las metas deben tener expectativas bien definidas y proveer resultados tangibles.

Los tres tipos de metas son:

1.-Metas mayores: Estos eventos de vida sistémica están manejados al final de cada fase de desarrollo. Ellos proveen una visión de los valores sistémicos, sincronizan la administración y las expectativas de ingeniería y verifican que las metas de la fase han sido alcanzadas.
2.- Metas menores: Estos eventos enfocados en iteraciones son conducidos para revisar el contenido de una iteración a detalle y para autorizar un trabajo continuo.

3.- Valoración de estatus: Estos eventos periódicos proveen administración con una revisión regular y frecuente en el progreso que esta siendo realizado.

Cada una de las cuatro fases - Inicio, Elaboración, Construcción y Transición - consiste en una o más iteraciones y concluye con una meta mayor cuando una capacidad técnica planeada es producida en una forma demostrable.

La figura 9 muestra la típica secuencia de los puntos de chequeo de los proyectos para proyectos grandes relativamente.

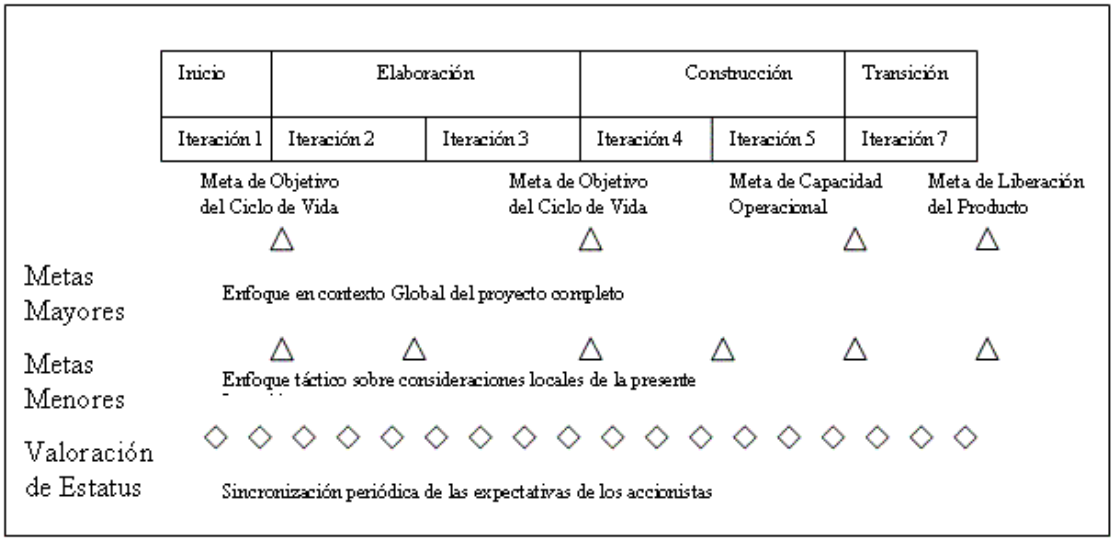


Figura 9 - Típica secuencia de puntos de chequeo en el ciclo de vida.
Fuente (Royce, 1998)

Etapas mayores: Son llevadas a cabo al final de cada fase de desarrollo, estas sincronizan las perspectivas administrativas con las de ingeniería, aseguran que los planes del ciclo de vida, las funciones, y la calidad sean desarrollados en un balanceado nivel de detalle, aseguran la consistencia entre los distintos artefactos y verifican que sean alcanzados los objetivos de cada fase.

Etapas menores: Estas son llevadas a cabo para revisar a detalle el contenido de una iteración y para autorizar el trabajo continuo.

Evaluación del estatus: Estos eventos proporcionan a la administración de una idea regular y frecuente (mensualmente, quincenalmente) del progreso realizado, son llevadas a cabo para dirigir los indicadores de progreso y calidad, para asegurar una atención continua a la dinámica del proyecto y para mantener una comunicación abierta entre todos los involucrados del proyecto.

En este apartado se he descrito la metodología propuesta por Walter Royce para el ciclo de vida de desarrollo de software, que nos plantea que estantes seguir con el objetivo de lograr que los procesos sean más estándares y se tenga un acercamiento a los posibles riesgos que se puedan tener. A continuación se describirá. Las disciplinas de administración de proyectos también recomendada por Walter Royce (1998) donde comenta los puntos a considerar para que la administración de proyectos sea exitosa, esto es que se terminen con la calidad, tiempo y costo que fueron planteados al inicio del proyecto.

Es de suma importancia es estar en constante monitoreo de la calidad del software, pues como comenta Jacobson (2001) referenciado por Rojas (2003), en donde comenta que no solamente es depurar errores al final del proyecto, sino durante toda la etapa de desarrollo.

A partir de lo expuesto, Luzuriaga (2002) resume que los beneficios comprobados al aplicar puntos de son:

- Reducción de los defectos en el uso del software.
- Reducción de los recursos de desarrollo, sobre todo en las etapas de codificación y prueba.
- Reducción en los costos de mantenimiento correctivo.

2.9 Disciplinas de Administración de Desarrollo del Software

En el apartado anterior se describió la metodología propuesta por Royce para incrementar las posibilidades de buen desarrollo de proyectos de software, lo que sirve de referencia para crear un proceso maduro de desarrollo de software. A continuación se verán las disciplinas que se deben de tomar en cuenta en la administración de proyectos de software también propuestas por Walker Royce (1998), para entender que en una metodología unificada es importante hacer un vinculo entre el ciclo de desarrollo de software y la administración de proyectos, viendo desde una perspectiva global, se tendrá una mayor posibilidad de éxito en el proyecto.

También Royce plantea los pasos a seguir a para lograr un administración de los recursos de un proceso de desarrollo de software. Dicho proceso abarca los siguientes puntos:

- Planeación de proceso iterativo
- Organización del proyecto y responsabilidades
- Automatización de procesos
- Control de proyectos e instrumentación del proceso
- Ajustar el proceso

La planeación la parte crucial de una administración. El reto es desarrollar un plan que balancee de la mejor manera los recursos disponibles para proveer de óptimas condiciones hacia todos los accionistas. La disciplina de organización de proyectos consiste por si misma con administración de personal- el equipo de la organización en términos de responsabilidades para operaciones eficientes. Automatización del proceso de desarrollo en un repositorio electrónico para proveer un fundamento de artefactos para una instrumentación efectiva.

Actividades de control del proyecto actúan como el sentido del proyecto. Estas actividades son usadas para evaluar la salud del plan de desarrollo, la calidad de los artefactos y la necesidad de cambios a cualquier conjunto de artefactos administrativos que definan las expectativas a través de los accionistas.

2.9.1 Planeación del proceso Iterativo

En cualquier proyecto informático, empezando con buen pie es crítico. NO solo es realizar esfuerzos tempraneros para ajustar el tono del proyecto entero, también se requiere de identificar los altos el riesgo y desafiar áreas del sistema. Probablemente más que la mitad de todos los proyectos estén condenados a fracasar dentro su primer mes, debido a los factores que incluyen:

- Una relación débil con los clientes.
- Presupuestos inadecuado.
- Pobre administración (incluyendo la inhabilidad de administrar y priorizar riesgos y una pobre meta administrativa).
- Habilidades y experiencias de ingeniería inadecuada.

- Calendario no realista.

El Proceso unificado de racional (RUP) trata de reducir el número de factores que contribuyen a proyectar fracaso, mejorando la eficiencia de un equipo y guiando al equipo de una manera madura. Un mejor control de los datos por parte del administrador del proyecto, mejores herramientas que soporten el equipo de ingeniería y mejores procesos ayudaran al producto informático a evolucionar en una forma previsible. (Franklin 2003)

Así como el desarrollo de software, la planeación el proyecto requiere un proceso iterativo como se hace con le software, el plan es una pieza intangible de propiedad intelectual a las cuales los mismos conceptos deben ser aplicados.

Los planes tienen una etapa de ingeniería, durante la cual el plan es desarrollado y una etapa de producción, cuando el plan es ejecutado. Los planes deben incluir como el entendimiento del problema envuelve el espacio del problema y el espacio de la solución.

Los errores de la planeación son solamente errores del producto. Mientras mas temprano en el ciclo de vida son resueltos, menos impacto tendrán en el éxito del proyecto,
Esta parte como lo comenta Royce (1998) no es un modelo de cocina es mas bien un modelo preliminar sencillo con algunas dimensiones, que servirán como punto de inicio para desarrollar un plan.

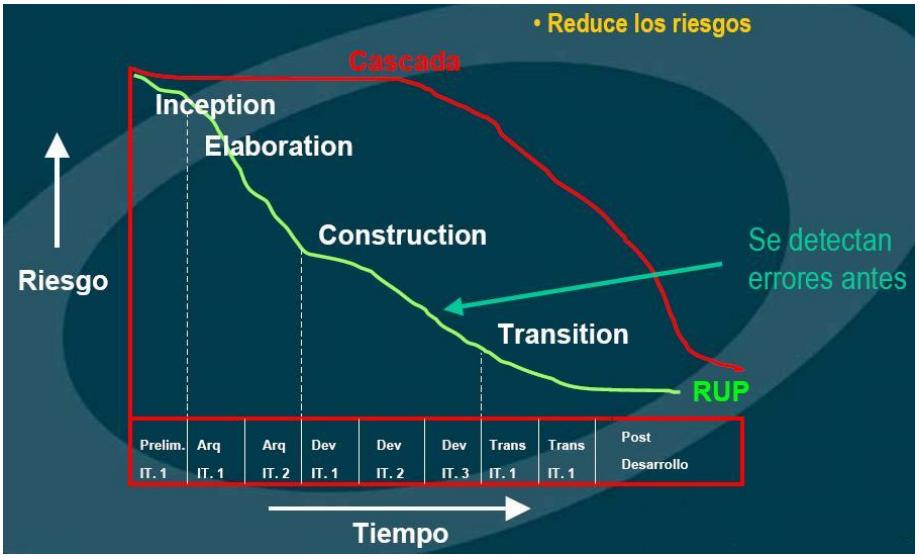


Figura 10 - Desempeño de un proyecto usando desarrollo iterativo

Fuente (Rojas 2003, 1998)

Es así como Rojas(2003) muestra en la figura 10 el desempeño en el tiempo de un proyecto de desarrollo de software utilizando el proceso iterativo contrastándolo con un proyecto utilizando el modelo tradicional o de cascada, en la figura se puede observar como con el proceso iterativo el riesgo es controlado y disminuye proporcionalmente

conforme transcurre el tiempo , mientras que en el proceso tradicional el riesgo no es controlado sino hasta ya muy avanzado el proyecto provocando que el proyecto se retrace y no se cumpla con los objetivos inicialmente planteados.

Beneficios

- Deducción de riesgos
- Se detectan errores antes
- Control estricto sobre variables de desempeño

2.9.1.1 Estructuras de Desglosamiento de trabajo

Una buena Estructura de Desglosamiento de Trabajo (WBS) es esto es una sincronización con el enfoque de esquema son factores críticos en un proyecto de software exitoso.

Las estructuras de desglosamiento de trabajo (WBS) son una simple jerarquía de elementos que descomponen el plan del proyecto en tareas de trabajo discretas. Un WBS proporciona la siguiente estructura de Información:

- El diseño de todo el trabajo significativo.
- Una descomposición clara de tareas para poder asignar las responsabilidades.
- Una estructura para la programación de tiempos, presupuestos, y registro de expediciones.

Muchos parámetros pueden manejar la descomposición del trabajo en tareas discretas; subsistemas de producto, componentes, unidades organizacionales, ciclo de vida del proceso, incluso geográficamente.

Las estructuras convencionales de WBS frecuentemente sufren de tres errores fundamentales.

1- Están prematuramente estructurado alrededor del diseño del producto.
2.- Están prematuramente descompuesto, planeados y presupuse todas en tanto mucho detalle.
3.- Están por proyecto específico y las comparaciones de proyecto cruzado son usualmente difíciles o imposibles.

Una típica estructura de una WBS es mostrada en la figura 11

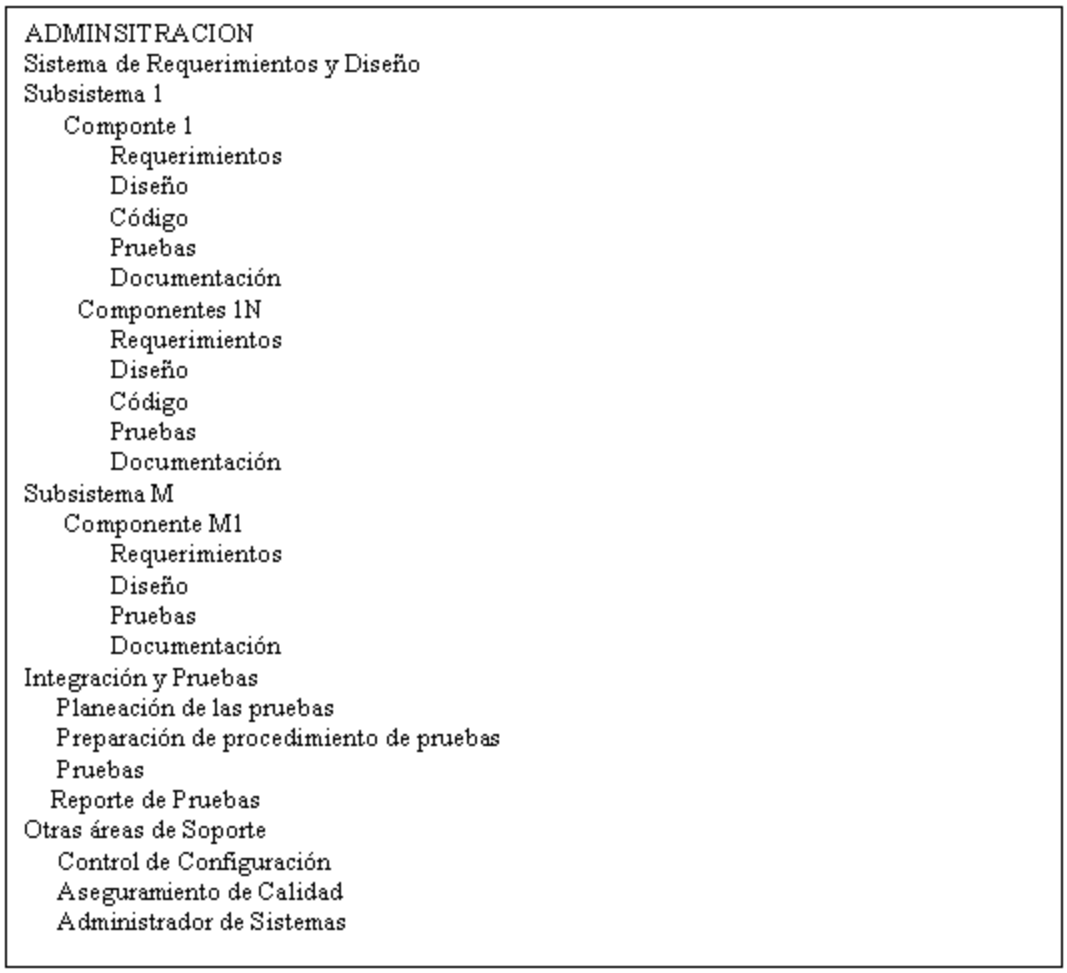


Figura 11 - Esquema estándar del WBS

Fuente: (Royce 1998)

Los problemas que sufren una WBS convencional son:

1.-Las Estructuras de desglosamiento del trabajo tradicionales están prematuramente estructuradas alrededor del diseño del producto.
2.- Las Estructuras de desglosamiento del trabajo tradicionales son prematuramente descompuestas, planeadas y fondeadas, en partes muy pequeñas o demasiado detalladas.
3.- Las Estructuras de desglosamiento del trabajo tradicionales son para un proyecto específico y una comparación entre proyectos es normalmente difícil o imposible.

La evolución de las WBS

Una WBS evolucionada debe organizar la planeación alrededor de la estructura del proceso en vez de la estructura del producto. La recomendación básica para realizar el WBS es organizando la estructura jerárquica como sigue:

1.- El primer nivel de elementos del WBS son los flujos de trabajo (ambiente, administración, requerimientos, diseño, implementación, valoración y desarrollo) estos elementos están usualmente colocados en un equipo simple y constituyen la anatomía el proyecto para los propósitos de planeación y comparación con otros proyectos.

2.- El segundo nivel de elementos del WBS son definidos por cada fase del ciclo de vida (Inicio, Elaboración, Construcción y Transición).Estos elementos permiten una fidelidad del plan para envolver mas naturalmente con el nivel de entendimiento de los requerimientos y arquitectura y los riesgos.

3.- El tercer nivel de elementos del WBS están definidos para enfocarse en las actividades que producen los artefactos de cada fase. Estos elementos deben ser al nivel mas bajo en la jerarquía que pueda colectar el costo de un artefacto discreto para una fase dado ellos deben ser compuestos adicionalmente a niveles de actividades bajas que, juntos tomen lo que produzca un artefacto.

Un WBS de estándar consiste a un esquema de procesos (fases, flujos de trabajo, y artefactos) como la mostrada en la figura 12. Esta estructura recomendada provee un ejemplo de cómo los elementos del esquema de procesos pueden ser integrados en un plan. Esto provee un esquema de estimación de costos y calendarios para cada elemento, colocándolos a través de la organización del proyecto.

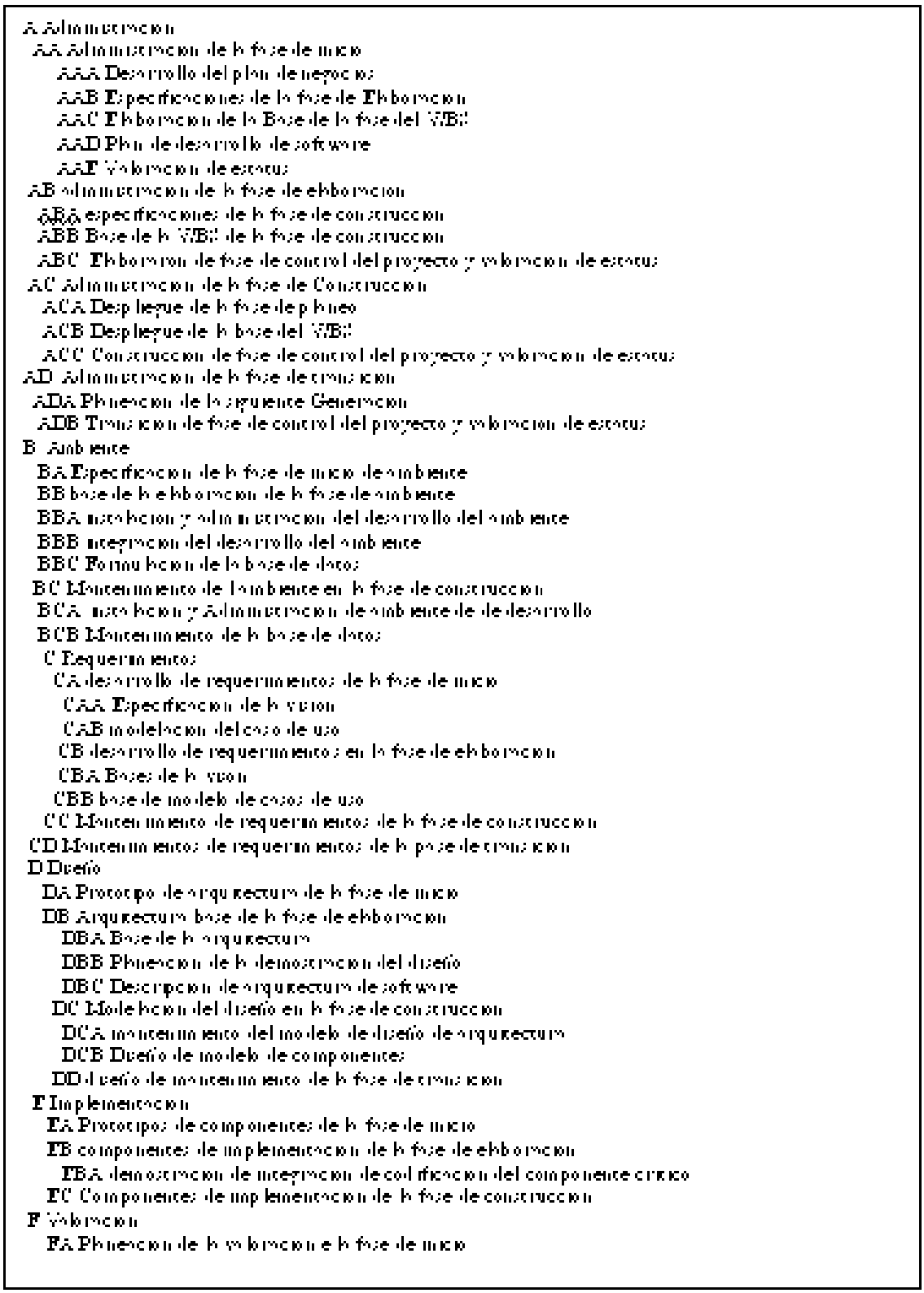


Figura 12 - Esquema estándar de un WBS

(Royce 1998)

Beneficios

- Estimación de costos acertada.
- Estimación de tiempos acertada.

2.9.1.2 Guía de Planeación

Los proyectos de software se extienden a lo largo de un amplio rango de dominios. Es valido pero riesgoso el hacer unas recomendaciones para plan especifico independientemente de contexto.

Dos simples guías de planeación deben ser consideradas cuando el plan de un proyecto comienza a ser desarrollado. La primera guía es descrita en la tabla 5, que prescribe una colocación de los costos a través del primer nivel de los elementos del WBS. La segunda guía detallada en la tabla 6 describe la colocación de los esfuerzos y fechas a través del ciclo las fases del ciclo de vida.

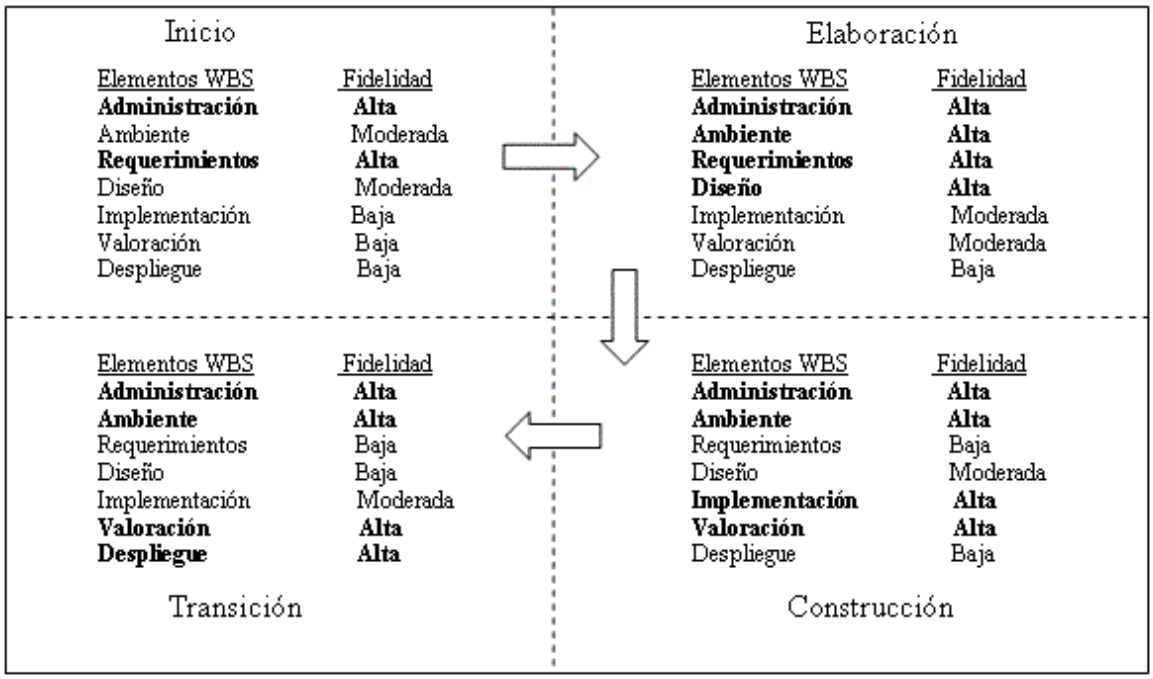


Figura 13 - Evolución de un plan de fidelidad en la WBS sobre le ciclo de vida.

(Royce 1998)

Elementos del WBS de primer nivel	**Presupuesto Default**
Administración	**10 %**
Ambiente	**10 %**
Requerimientos	**10 %**
Diseño	**15 %**
Implementación	**25 %**
Valoración	**25 %**
Despliegue	**5 %**
Total	**100 %**

Tabla 5 - Prosupuestos estándar para un WBS.

(Royce 1998)

Dominio	Inicio	Elaboración	Construcción	**Transición**
Esfuerzo	5 %	20 %	65 %	**10 %**
Tiempo	**10 %**	**30 %**	**50 %**	**10 %**

Tabla 6 - Distribución estándar de los esfuerzos y tiempos ordenados por fase.

(Royce 1998)

Beneficios

- Distribución de recursos adecuada.
- Conocimiento y control sobre disponibilidad de recursos.
- Disminución de riesgos.

2.9.1.3 Estimación de Tiempos y Costos

El proceso de planeación del proceso necesita estar derivado desde dos perspectivas. La primera es el enfoque de Top-Down. El cual comienza con el entendimiento general de los requerimientos, y luego descomponerlo en elementos de más bajo nivel, desde esta perspectiva la secuencia es:

1. El administrador del proyecto caracteriza en forma general el tamaño, ambiente, personas y la calidad requerida por el proyecto.
2. El administrador del proyecto de software realiza una estimación a nivel macro del esfuerzo y los tiempos usando un modelo de estimación de costos.
3. El administrador del proyecto realiza la Distribución del esfuerzo en los primeros niveles de la jerarquía del WBS.
4. El sub.-administrador del proyecto descompone cada elemento del WBS en niveles más bajos.

La segunda perspectiva es el enfoque de bottom-up, el cual comienza con una mentalidad en el fin, analiza los tiempos y presupuestos de nivel micro, y suma todos los elementos en presupuestos de alto nivel y etapas intermedias. Desde esta perspectiva, la secuencia es:

1. El administrador responsable de los elementos del WBS, elabora en tareas detalladas los elementos de más bajo nivel del WBS.
2. Las estimaciones son combinadas e integradas en elementos de más alto nivel del WBS.
3. Se realiza una comparación con los presupuestos y tiempos del top-down. Las diferencias más grandes son evaluadas y se realizan los ajustes con el fin de lograr una convergencia entre las estimaciones de top-down y bottom-up.

Los presupuestos y la estimación de costos utilizando top-down tienden a ser en general optimista, mientras que el enfoque bottom-up tiende a ser pesimista. Estos dos enfoques de planeación deben ser usados conjuntamente a través del ciclo de vida del proyecto. En la figura 14 se muestra el balance que debe existir entre la utilización de la perspectiva top-down y bottom-up para la planeación del proyecto. Durante la etapa de ingeniería la perspectiva top-down debe dominar, mientras que la perspectiva bottom-up deberá ser la que domine durante la etapa de producción.

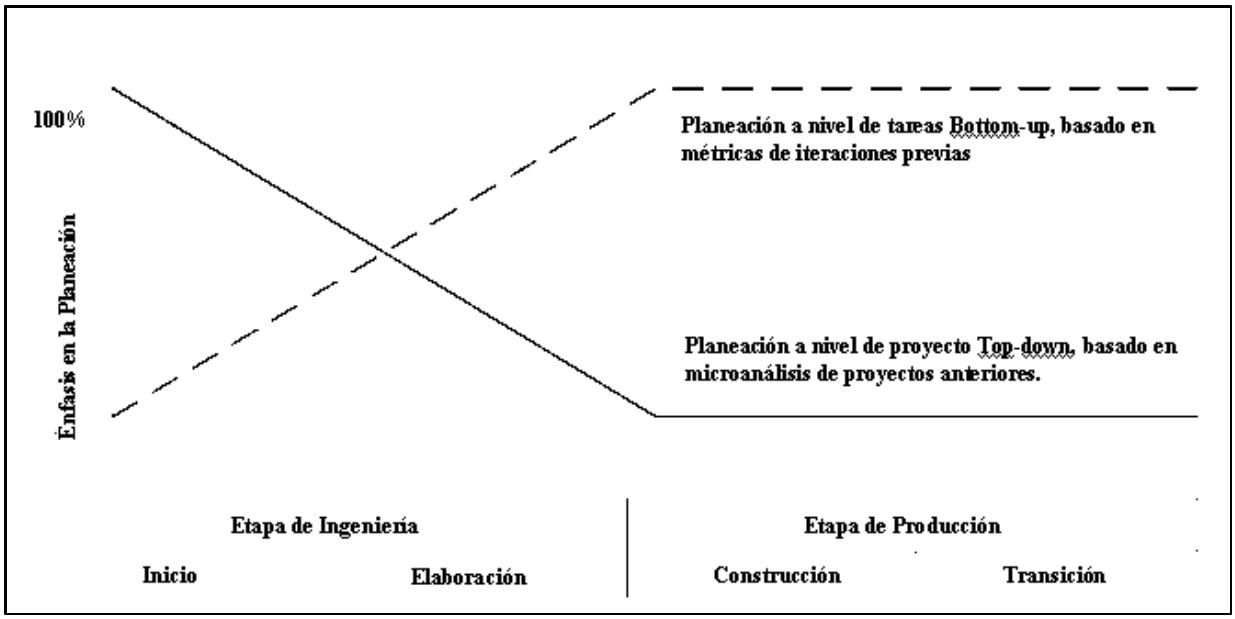


Figura 14 - Balance de la planeación a través del ciclo de vida.

Fuente (Royce, 1998)

La planeación es extremadamente importante para el éxito del proyecto, esta provee una estructura para tomar decisiones y transforma estructuras de proceso genéricas y subjetivas en procesos objetivos. El plan del proyecto es la definición de cómo los requerimientos del proyecto serán transformados en un producto con las características del negocio, este debe ser realista, actual, debe ser producto de un equipo, ser entendido por todos los involucrados del proyecto, y debe ser usado.

2.9.2 Planeación del Proceso Iterativo

Otra dimensión de la planeación que es concerniente a calendario de las metas mayores y sus iteraciones es probablemente lo más tangible del plan de la administración de riesgo total.
Un programa genérico que se construye y guías generales en un número de iteraciones en cada fase son descritos a continuación:

Iteración de inicio.- en las actividades de prototipos tempranas integran los componentes fundaméntales de una arquitectura candidata y provee un ejecutable esquema para la elaboración de un caso de uso critico del sistema , este esquema incluye los componentes existentes , componentes comerciales y suficientes prototipos para demostrar la arquitectura candidata y suficiente entendimiento de los requerimientos para establecer un caso de negocios creíbles, visión, y plan de desarrollo de software. El desarrollo de gran escala necesitara de 2 iteraciones para alcanzar un prototipo aceptable.

Iteración de elaboración.- Estas iteraciones son el resultado de una arquitectura, incluyendo un esquema completo y una infraestructura para la ejecución. Antes de completar la arquitectura de iteración, unos casos de uso crítico deben de ser demostrados: inicializar la arquitectura, inyectar un escenario para manejar el peor caso de flujo de procesamiento de datos a través del sistema. Desarrollar un escenario para manejar el peor caso de flujo de control a través del sistema.
Muchos proyectos deben de planear en dos iteraciones para alcanzar una guía de de arquitectura aceptable. Arquitecturas sin precedentes podrán requerir iteraciones adicionales, considerando proyectos con una arquitectura bien establecida probablemente se obtiene en una iteración.

Iteración de Construcción Muchos proyectos requieren por lo menos dos iteraciones mayores en construcción: una versión alfa y una versión beta del entregable. Un entregable alfa incluirá el la capacidad de ejecutar todos los casos de uso críticos. Usualmente representa solamente el 70% del total del producto y desempaña unos niveles de calidad por debajo de los esperados en el producto final. Un entregable beta provee típicamente el 95% de la capacidad total del producto y alcanza importantes niveles de calidad. A través de muchos proyectos se necesitaran al menos uno o dos mas iteraciones de construcción, pues hay muchas razones para sumar una o dos iteraciones con el objetivo de administrar riesgos u optimizar el gasto de recursos.

Transición de iteración- muchos proyectos usan una sola iteración en la transición de la versión beta al producto final.

La guía es que la mayoría de los proyectos usaran entre cuatro y nueva iteraciones. El típico proyecto tendrá las siguientes 6 iteraciones:

- Una iteración en inicio: un prototipo de arquitectura.
- Dos iteraciones en elaboración: un prototipo de arquitectura y una base de arquitectura.
- Dos iteraciones en construcción: una versión alfa y una versión beta.
- Una iteración en transición: la entrega del producto.

Proyectos altamente predecibles con una arquitectura predefinida o proyectos de escala pequeña, podrían tener una sola iteración en fases combinadas como inicio y elaboración y pueden producir un producto eficiente con un total de cuatro iteraciones. En cambio un proyecto muy grande e impredecible con muchos accionistas, requerirá tal vez de un total de 9 iteraciones. La administración total resultante debe ser bien hecha para costear la administración de los riesgos y la sincronización de los accionistas.

9.2.1 Planeación Pragmática

Incluso en una buena planeación, el proceso iterativo es mas dinámico, cuando se ejecuta una iteración N de cualquier fase, el administrador del proyecto debe estar monitoreado y controlado a través del plan que fue inicializado en la Iteración N-1 y debe estar planeando la iteración N+1. El arte de una buena administración de proyectos es negociar el plan en la presente iteración y planear la siguiente iteración basado en los resultados deseados en la presente iteración y las pasadas iteraciones.

Consecuentemente el éxito de cada proyecto exitoso puede ser atribuido a una buena planeación. Es por eso que hacemos especial énfasis en planeación, requerimientos y arquitectura.

Un plan de proyecto es la definición de cómo los requerimientos del proyecto serán transformados en un producto con los requerimientos del negocio. Este debe ser realista, debe ser actualizado, debe ser un producto de equipo, debe ser entendido por todos los participantes y más importante, debe ser usado.

La planeación no es solamente para los administradores, entre más abierto y visible serán los procesos de planeación y los resultados, la propiedad mostraran los miembros del equipo que lo ejecutaran. Un mal plan puede causar desgaste, en cambio un buen plan, puede formar una cultura y animar el trabajo en equipo.

2.9.3 Organización del proyecto y responsabilidades

Las líneas de negocios del software y equipo de proyectos tienen diferentes motivaciones. Las líneas de software de negocios están motivadas por el retorno de inversión, nuevos negocios discriminadores, diversificaron de mercados y ganancias. Los equipos del

proyecto están motivados por el costo, el calendario y la calidad de los entregables específicos.

Los profesionistas del software en ambos tipos de organizaciones están motivados por crecimiento profesional, satisfacción en el trabajo y oportunidad de hacer la diferencia.

En el pasado la mayoría de las organizaciones estaban enfocadas en el proyecto, cuyo nivel es donde el software es desarrollado entregado. Los proyectos tienen sus intereses propios y raramente invertirán en tecnologías o servicios que no tengan impacto directo en el costo, tiempo o calidad de los entregables. Esta parte describirá como se debe manejar el equipo del proyecto.

Línea de negocios de las organizaciones
La figura 15 muestra los roles y las responsabilidades de una organización de línea de negocios estándar esta estructura puede ser modificada a la medida de acuerdo circunstancias específicas:

Las características principales de las organizaciones son las siguientes:

1.- Responsable de la definición de procesos y el mantenimiento es especifico para una línea de negocios cohesivo, donde el procesos tiene sentido. Ejemplo el proceso de desarrollo de software para aviones es diferente que para desarrollar aplicaciones de oficina.

2.- La responsabilidad de la automatización de procesos es un rol organizacional y es igual de importante que el rol de definición de procesos.

3.- Los roles organizacionales deben ser llenados individualmente o por diferentes equipos, dependiendo de la escala de la organización. Un proyecto de 20 personas probablemente requerirá solo una persona para llenar los roles, mientras una compañía de telecomunicaciones probablemente requerirá cientos de personas para alcanzar una organización efectiva del software.

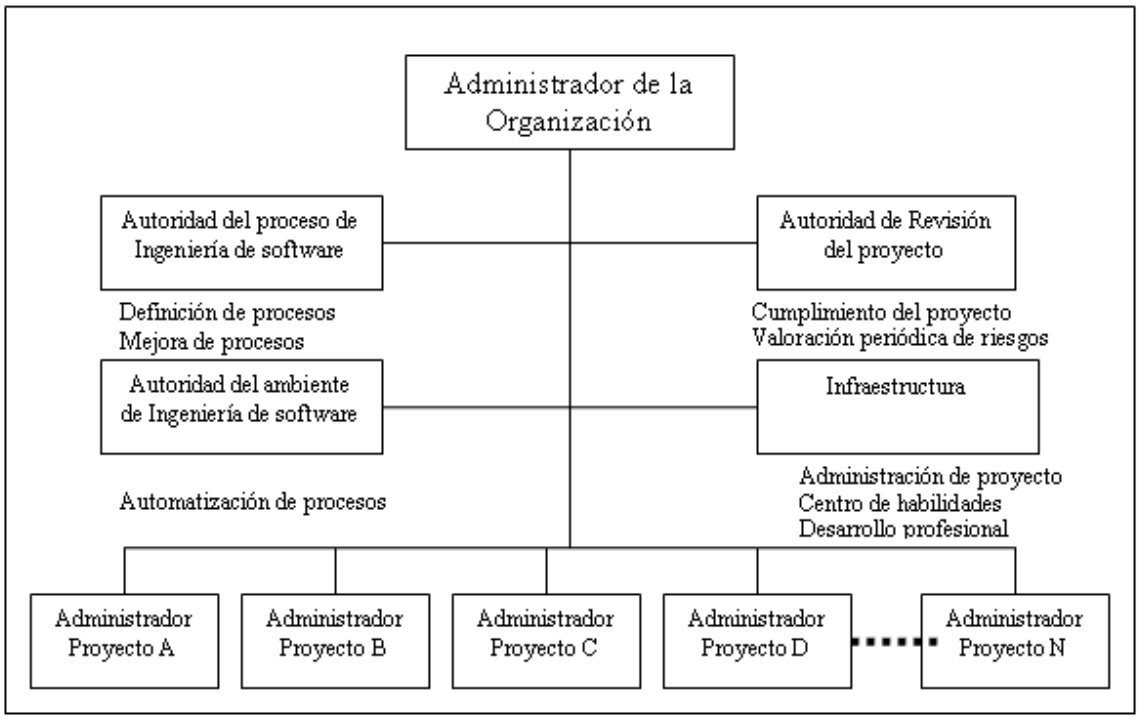


Figura 15 - Roles estándares de las organizaciones de líneas de negocios.

Fuente (Royce, 1998)

-**Autoridad del proceso de ingeniera de software:** facilita el intercambio de información y la un de procesos desde y ara los participantes del proyecto. Esta rol le rinde cuentas al administrador general de la organización de mantener valoración de la madures de los procesos y su planeación para futuras mejoras. El SEPA es un rol necesario en una organización. Este toma la responsabilidad de contabilizar la definición d e procesos y su mantenimiento (modificación, mejora, y inserción de tecnología).

- **Autoridad de revisión de proyecto:** es un individuo responsable de asegurarse que el software cumpla con todas las políticas, prácticas y estándares organizacionales y del negocio.

- **Autoridad de ambiente de ingeniería de software:** Es el responsable por automatizar los procesos del la organización, mantener un ambiente estándar en la organización, proyectos de entrenamiento para usar el ambiente y mantener los activos de la organizaron usables. Este rol necesita alcanzar un nivel de retorno de inversión para un proceso común. Herramientas, técnicas y enredamiento puede ser amortizado efectivamente en la organización a través de múltiples proyectos si alguien en la organización es responsable de soportar, administrar y estandarizar el ambiente.

- **Infraestructura:** La infraestructura de la organización provee soporte de recursos humanos, investigación y desarrollo de independiente del proyecto y otros activos de capitales de ingeniería de software. La infraestructura para cualquier línea de negocios de software puede estar desde lo trivial hasta altamente burocráticas. Los componentes típicos de la infraestructura son los siguientes:

1. Administrador del proyecto.
2. Centro de habilidades de ingeniería.
3. Desarrollo profesional.

Organización del proyecto

La figura 16 muestra la organización de un proyecto y dibuja los roles y responsabilidades a nivel proyecto a nivel proyecto. Esta estrictita puede adecuarse al tamaño y circunstancias de una organización de proyecto.

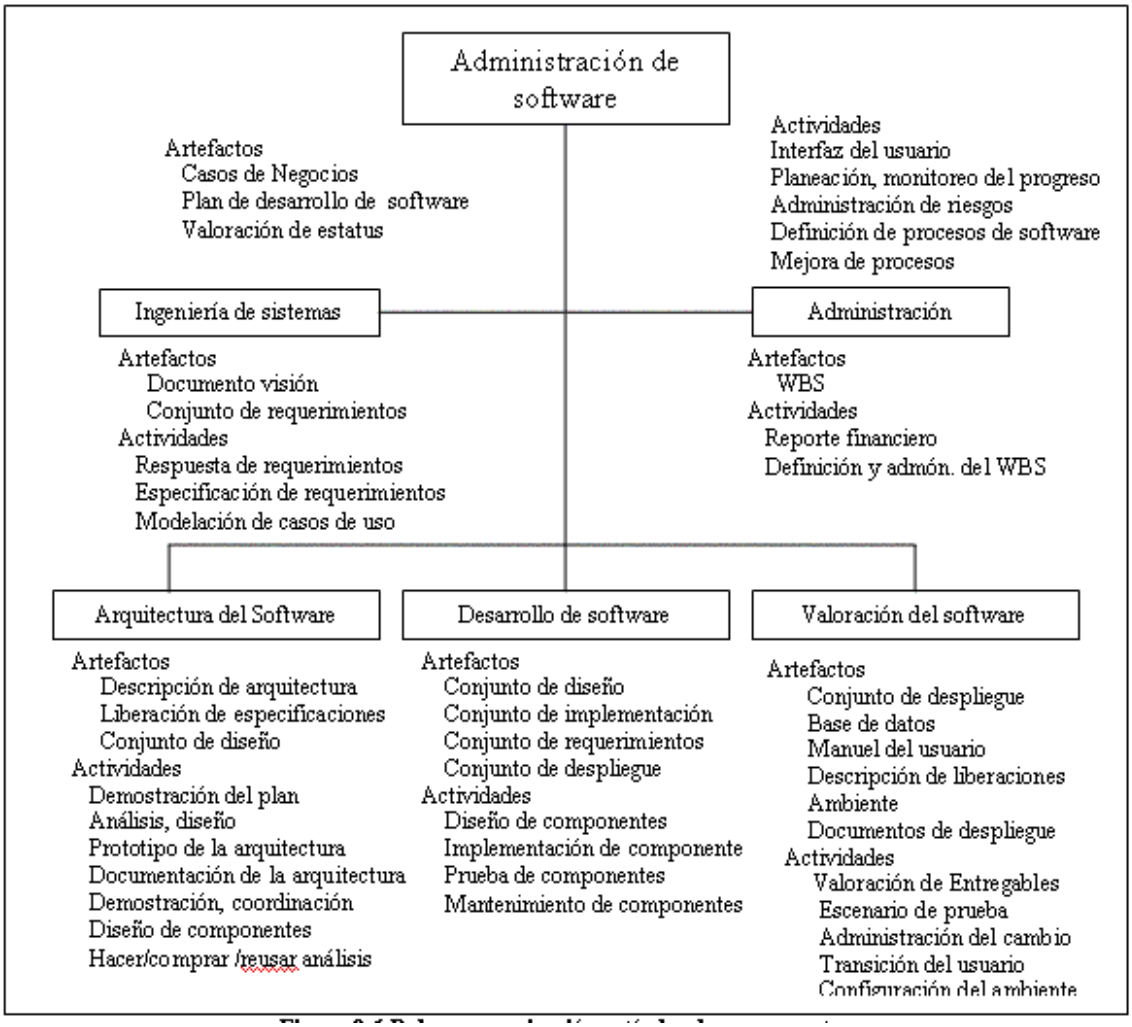


Figura 16 - Roles y organización estándar de un proyecto

Fuente (Royce, 1998)

Las características principales de una organización estándar de proyecto son las siguientes:

- El administrador del proyecto es el participante activo, responsable de producir así como administrar.
- El equipo de arquitectura es el responsable de los artefactos reales y por la integración de componentes, no solamente funciones staff.
- El equipo de desarrollo posee los componentes de las actividades de construcción y mantenimiento. El equipo de evaluación es separado del de desarrollo. Esta estructura fomenta una perspectiva de calidad independiente y se enfoca en el equipo de pruebas y de evaluación del producto.
- La calidad es el trabajo de todos, integrándola es todas las actividades y puntos de chequeo. Cada equipo se hace responsable de una perspectiva de calidad diferente.

Equipo de administración del software
El equipo de administración del software es el encargado de proveer condiciones ganadoras para todos los involucrados (accionistas) es este sentido el administrador de proyecto gasta todo el día un trabajo de balance.

El equipó administrador es el responsable de la planeación de esfuerzos, conducción del plan, y adaptar el plan a los cambios en el entendimiento de bs requerimientos o del diseño. Además toma el control de los recursos administrativos y del alcance del proyecto y opone prioridades operacionales a través del ciclo de vida.

El equipo de administración de software toma la propiedad de todos los aspectos de calidad. En particular, es responsable de lograr y mantener un balance entre esos aspectos así que la solución global es adecuada para los accionistas y optimas como sea posible.

Equipo de arquitectura del software
El equipo de arquitectura es el responsable de la arquitectura, esta responsabilidad comprende la ingeniería necesaria para especificar un conjunto completo de materiales para el desarrollo del software y ingeniería necesaria para hacer las tratos así que los componentes estándares estén elaborados para asegurar que los costos de construcción sean predecibles.

Equipo de desarrollo del software
El equipo de desarrollo de software es el grupo de más aplicaciones específicas. En general, el equipo de desarrollo comprende varios sub-equipos dedicados a grupos e componentes que requieren habilidades comunas. Las habilidades típicas son las siguientes:

- Componentes comerciales
- Bases de datos
- Interfase gráfica del usuario
- Sistemas operativos y redes
- Aplicaciones de dominio

El equipo de desarrollo del software es el responsable de la calidad de los componentes individuales, incluyendo todos los componentes de desarrollo, pruebas, y mantenimiento. Los componentes de prueba deben de ser construidos como auto documentados, el software repetible esta es tratado como otro componente operacional así que se puede mantener naturalmente y disponible para automatizar. El equipo desarrollo decide como es resuelto cualquier diseño o asunto de implementación local para un solo componente.

Equipo de evaluación del software
Hay dos razones para usar equipos independientes para la evaluación de software. La primera tiene que ver con los aseguramientos de una perspectiva de calidad independiente. Esta frecuentemente tiene sus ventajas (la cual asegura que la autoridad de los perjuicios de desarrolladores no contamina la evaluación de calidad) y sus desventajas (cual como relevar al equipo de desarrollo de software de promediad de calidad, a algo extendido).
Una razón mas importante para usar equipo de pruebas independientes es las concurrencias de actividades, Calendarios pueden ser acelerados desarrollando software y preparando para pruebas en paralelo con actividades de desarrollo. Administración de cambio, prueba de planes y desarrollo de prueba de escenarios pueden ser mejorados en paralelo usando diseño y desarrollo.

2.9.4 Automatización del Proyecto

La sección2.9.1 introduce los tres niveles de procesos. Cada nivel requerir cierto nivel de automatización para el procesos correspondiente para ser acarreado ala eficiencia.

1.- Metaprocesos: las políticas de la organización, procedimientos y prácticas para administrar una intensiva línea de negocios. El soporte de automatización para este nivel es llamado infraestructura. Una infraestructura es un inventario de las herramientas preferidas, plantillas de artefactos, guisa de micro procesos, guías de macro procesó, repositorio de desempeño del proyecto, base de datos con el conjunto de habilidades organizacionales, y una librería de ejemplos de planes de proyectos pasados y resultados.

2.-Macro procesos: son las políticas del proyecto, prácticas para producir un producto completo de software con certidumbre en el tiempo, costo y calidad. El soporte de automatización de este proceso del proyecto es llamado ambiente. Un ambiente es la colección específica de herramientas para producir un conjunto especifico de artefactos que para gobernad el plan de un proyecto especifico.

3.- Micropocesos: son las políticas del equipo, procedimientos y prácticas para alcanzar un artefacto del proceso del software. El soporte de automatización para generar un artefacto es generalmente llamado herramienta. Las herramientas típicas incluyen administración de requerimientos, modelación visual, compiladores editores, debuggers, administración del cambio, automatización de métricas automatización de documentos, automatización de pruebas, estimación de costos y automatización de flujos de trabajo.

Beneficios

- Reducción de tiempo de programación
- Reducción de errores al tener automatizado
- Estimación de costo y tiempo más acertada
- Necesidad de gerente no tan especializada
- No depende tanto de la habilidad del personal

2.9.4.1 Herramientas: Construcción Automática de Bloques

Muchas herramientas están disponibles para automatizar el procesos de desarrollo de software, en esta sección se enfoca a proveer un ambiente global necesario para soportar el esquema de trabajo. Introduce algunas de las herramientas importantes que son necesarias universalmente a través de están correlacionados e un esquema de proceso. La mayoría de las herramientas clave de desarrollo de proyectos son mostradas en la figura 17 en uno de los flujos de trabajo del proceso.

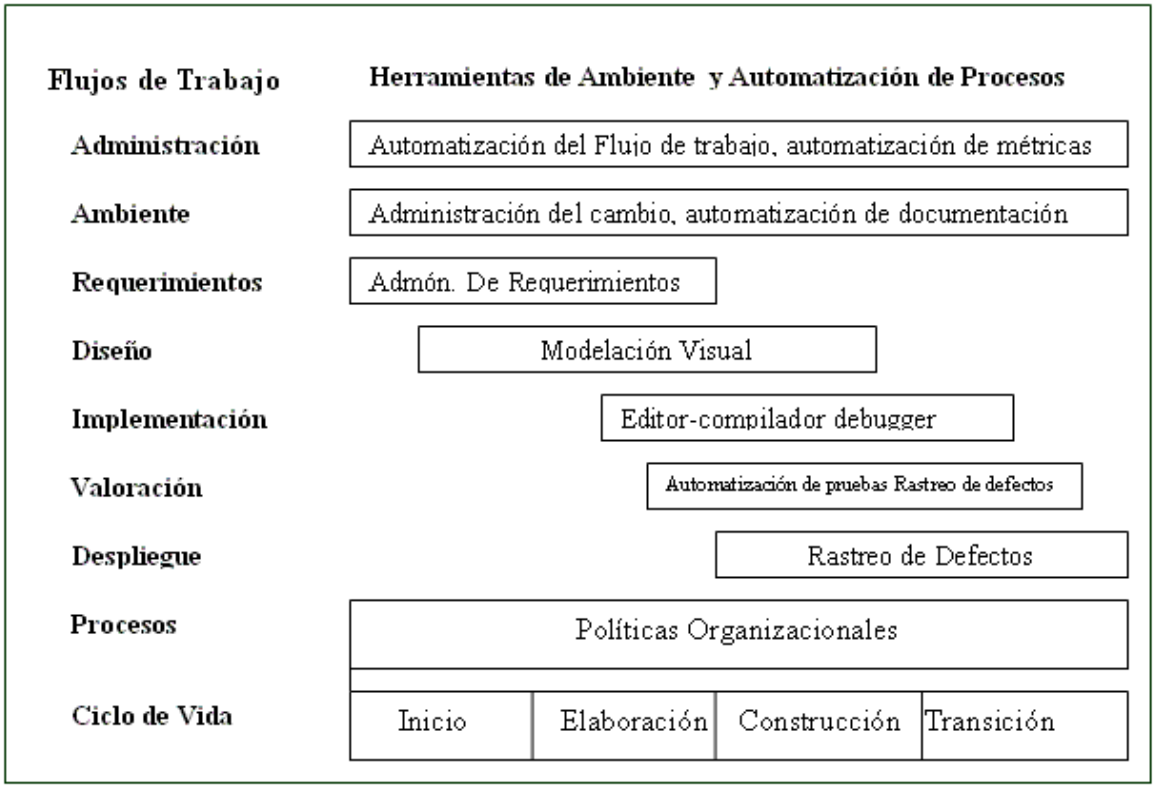


Figura 17 - Típica herramientas de automatización que soporta los flujos de proceso

(Royce 1998)

Cada uno de los flujos de procesos tiene una necesidad distinta para soporte de automatización. En algunos casos es necesario generar un artefacto, algunos artefactos críticos asociados con cada flujo de procesos son descritos a continuación.

Administración: Hay muchas oportunidades para la automatización de la planeación de proyecto y e control de las actividades de administración de Flujo de trabajo. Herramientas de estimación de costos de software y herramientas de WBS son útiles para generar la planeación de artefactos. Para administrar a través del plan, herramientas de administración de flujo de trabajo. Y control de proyecto, Esta automatización soporta considerablemente el mejoramiento de la colección de métricas.

Ambiente: Administración de configuración y control de versiones es esencial en un proceso de desarrollo iterativo.

Requerimientos: En un proceso moderno, el sistema de requerimientos están capturados en el enunciado de visión, los modelos iterativos permiten al usuario y el desarrollador con versiones tangibles del sistema.

Diseño: las Herramientas que soporten a lo requerimientos, diseño, implementación y valoración son comúnmente usadas juntos. De hecho mientras menos separadas estén, es mejor. La principal herramienta requerida para el diseño del flujo de trabajo, es la modelación visual, la cual es usada para capturar y diseñar modelos, presentando un formato entendible por el humano.

Valoración y despliegue: La valoración de flujo de trabajo utiliza todas las métricas usadas así como las capacidades de soportar la automatización y administración de pruebas, para incrementar la libertad de cambio, probar y documentar la producción debe ser mayormente automatizada. El rastreo de defectos es otra herramienta importante que soporta la valoración.

Beneficios

- Reducción de tiempo de programación
- Reducción de errores al tener automatizado
- Estimación de costo y tiempo más acertada
- Necesidad de gerente no tan especializada
- No depende tanto de la habilidad del personal

2.9.5 Control e Instrumentación de los procesos del proyecto

2.9.5.1 Las siete métricas clave

Indicadores Administrativos
Trabajo y progreso (ejecución del trabajo a través del tiempo).
Los presupuestos de costos y gastos (costos incurridos a través del tiempo).
Colocación de personal y dinámica de equipos (cambios en el personal a través del tiempo).

Indicadores de calidad
Cambios en el tráfico y estabilidad (cambio del tráfico a través del tiempo).
Rotura y modularidad (promedio de roturas por cambio a través del tiempo).

Re-trabajo y adaptabilidad (promedio de re-trabajo por cambio a través del tiempo).
Tiempo promedio entre fallas (MTBF por sus siglas en ingles) y madurez (taza de defectos a través del tiempo).

Beneficios

- Constante seguimiento de las variables
- Datos para toma de decisiones
- Minimizar riesgos
- Aseguramiento de calidad

2.9.6 Adecuar el proceso

El esquema de los procesos debe ser configurado a las características específicas del proyecto la escala del proyecto. En particular el tamaño del equipo maneja la configuración más que ningún otro factor.
Otros factores claves incluyen las relaciones con los accionistas, flexibilidad de procesos madurez de procesos, riesgo de la arquitectura y experiencia de dominio. Mientras la implementación de procesos específicos puede variar el espíritu del equipo será el mismo.

En la adecuación de procesos administrativo para un proyecto especifico, hay dos dimensiones de factores a tomar en cuenta la complejidad técnica y la complejidad administrativa. La figura 18 muestra las prioridades que de tendrán en cada uno de las dos dimensiones dependiendo de la cada uno de las complejidades.

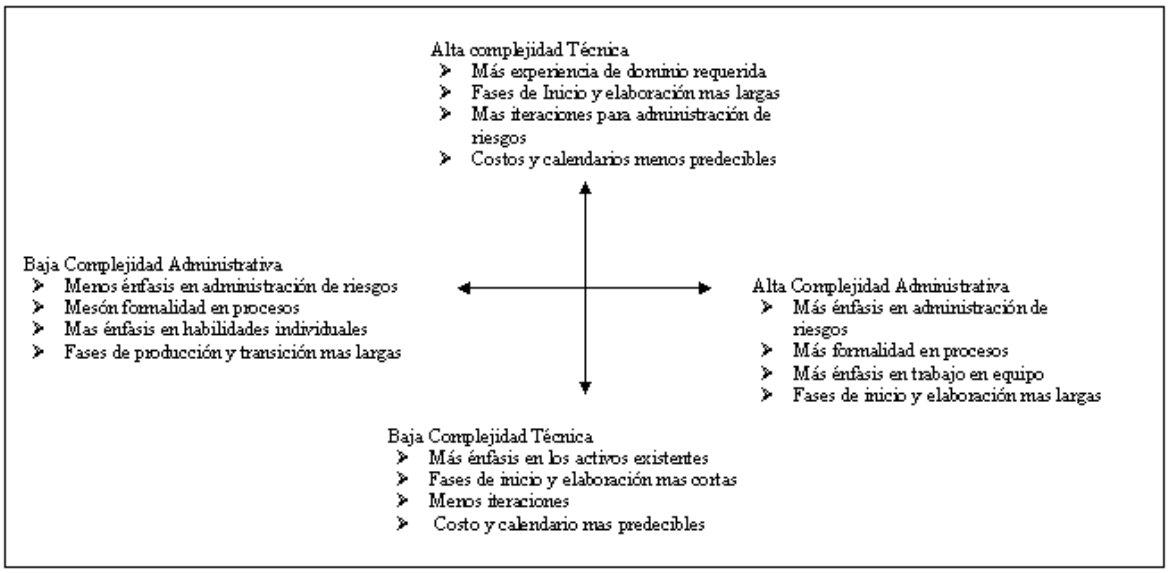


Figura 18 - Prioridades de adecuación del esquema del proceso

Royce (1998)

Una estimación de costos pragmática

Un problema crítico en la estimación de costos, es la falta de casos de estudio bien documentados de proyectos que han usaron un enfoque de desarrollo iterativo. A través de los modelos de costos los vendedores comentan que sus herramientas son convenientes para estimar proyectos de desarrollo iterativo, unos pocos más están basados en bases de datos empíricas con historias de éxito de desarrollos iterativos modernos.

Cerca del 50% de los herramientas de estimación de costos, datos y servicios compiten con la industria del software (pag26) hay algunos modelos de estimación de costos muy populares como son *Cocomo, check point, knolewgeplan, price-s proQMs seer, slip, softcost and sqpr/20*.

Beneficios

- Optimización de recursos
- Optimización de personal
- A la medida

Como se ha podido observar en el presente capitulo, es necesario contra con metodologías que unifique una metodologías de proceso iterativo, con las mejores disciplinas de administración de proyectos de desarrollo de software.

Como punto final comenta Ivar Jacobson:

“Un procesos sin herramientas integrales, es solamente una idea académica”

El RUP Materializa las Mejores Prácticas y asegura que todos entiendan claramente y puedan seguir un proceso práctico al desarrollar software.

Capitulo III. Modelo particular

El modelo particular que se describe a continuación, es una representación esquemática de cómo se desarrolla la presente investigación, la cual tiene como finalidad comprobar que "Con ayuda de una metodología de desarrollo de proyectos de software que se base en un enfoque unificado, se podrán realizar proyectos de manera exitosa", la cual corresponde a la hipótesis planteada al inicio de la presente investigación.

A continuación se describirá cada uno de los elementos que integran este modelo en este modelo en particular:

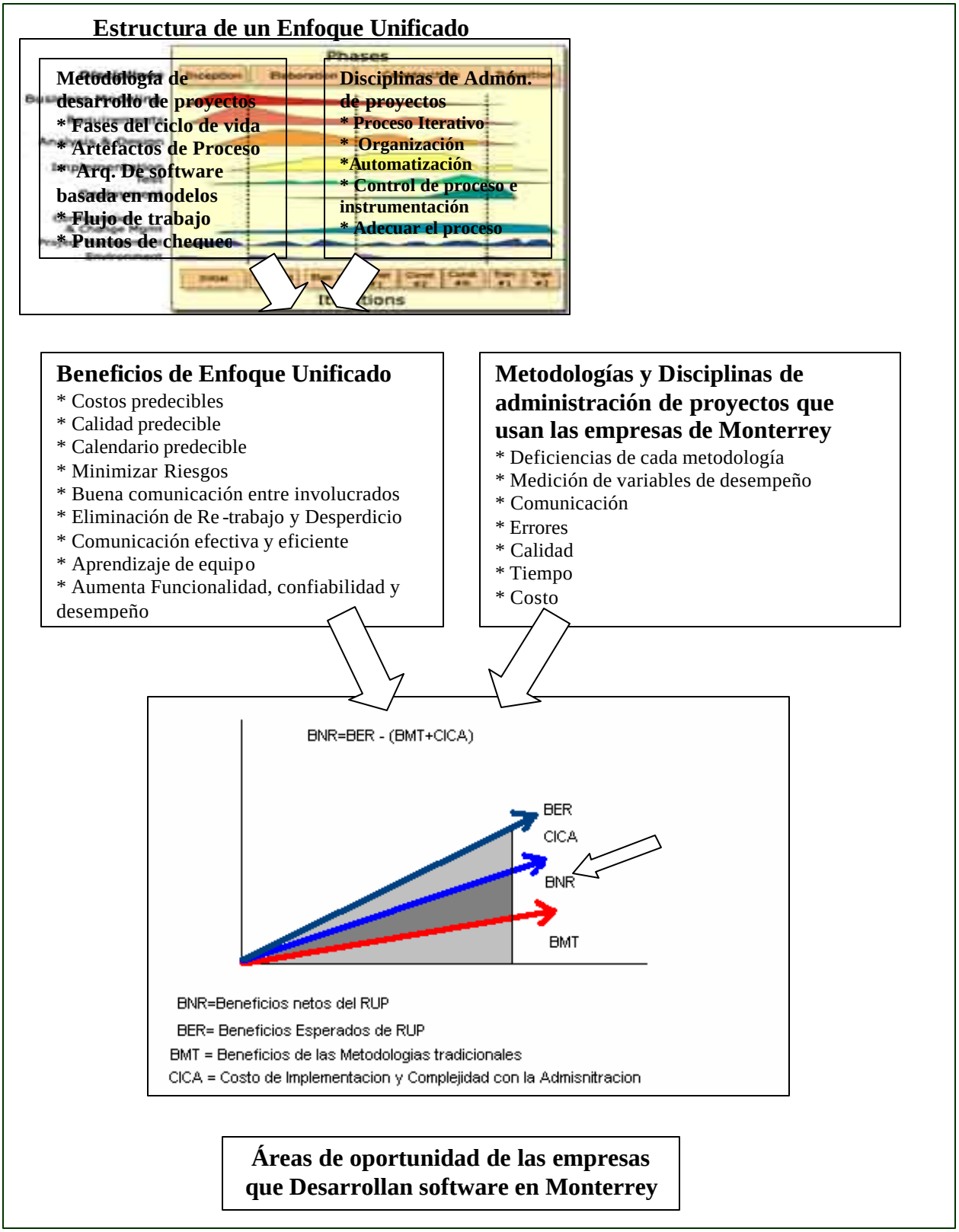


Figura 19 - Modelo Particular

3.1 Descripción de las etapas del modelo particular.

El modelo particular que se presenta en la figura 19, pretende describir la manera en que la presente investigación será llevada a cabo, se describe cada una de las partes que componen este modelo: El cual tiene como objetivo verificar que la hipótesis que se planteo al inicio de la presente investigación sea verdadera.

La figura 20, muestra la representación del proceso unificado, el cual es un compendio de las mejores prácticas de la industria del software, que unifica en un enfoque de desarrollo de software elementos como: ciclos de vida de software, artefactos de proceso, flujos de trabajo, arquitectura de software basada en modelos, puntos de chequeo, entre otros. Y los une a las disciplinas de administración de proyectos que comprenden: el proceso iterativo, organización, automatización, control de los procesos y por ultimo la adecuación del proceso.

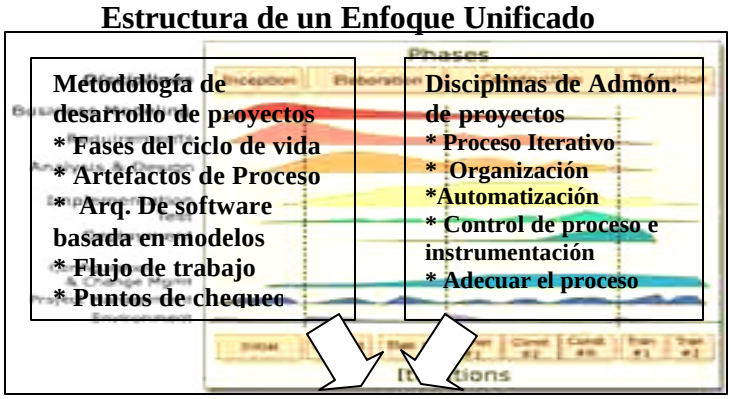


Figura 20 - Estructura de un Enfoque Unificado

El enfoque de proceso unificado conjunta las mejores prácticas del desarrollo de software que involucran:

- Ciclo de vida de desarrollo de software
- Artefactos del proceso
- Arquitectura de software basado en modelos
- Flujos de trabajo de los procesos
- Puntos de chequeo del proceso

Y los une con las disciplinas de administración de proyectos de desarrollo de software, que ayudaran a mantener un estrecho control de los recursos, requerimientos, métricas y personal que interviene en el desarrollo del proyecto.

A continuación se mencionan las disciplinas que intervienen en este enfoque:

- Planeación de un proceso iterativo
- Organización del proyecto y las responsabilidades
- Automatización de procesos
- Control del proyecto e instrumentación del proceso
- Adecuar el proceso

En la figura 21 se representan los beneficios que son esperados al aplicar las mejores prácticas del desarrollo de software, en este caso la implantación de una metodología basada en un enfoque iterativo propuesto por Royce.

Al implantar la metodología de desarrollo unificado propuesta por se obtendrán beneficios como:

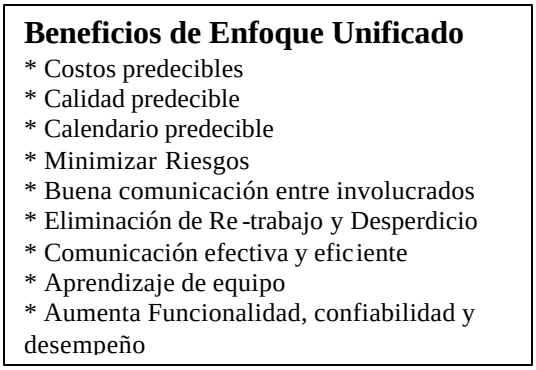


Figura 21 - Beneficios del Enfoque Unificado

- **Costos predecibles:** Tener una estimación de costos muy acertada, ayuda a que los proyectos sean rentables, al saber con exactitud el costo real del proyecto en todo momento. El desarrollo de software es una inversión, por que debe cuidar que exista un costo bajo (deseable) y un rendimiento alto.
- **Calendario predecible:** Tener un calendario predecible ayudara al administrador a maximizar el uso de sus recursos, tanto humanos como materiales.
- **Calidad predecible:** Teniendo una calidad predecible, significa que siempre se tendrá la seguridad de que no habrá fallas, y siempre se tendrá la seguridad de la funcionalidad del software.
- **Minimizar riesgos:** Al minimizar riesgos se tiene la certeza de cual será el desempeño del desarrollo, al igual que siempre se sabrá con exactitud el tiempo y costo del proyecto. En pocas palabras te tendrá un estricto control del desarrollo del proyecto**.**
- **Mejorar comunicación entre involucrados:** Al mejorar la comunicación de los involucrados, estos estarán al tanto de los avances, requerimientos, riesgos, calendario y puntos críticos, por lo que se podrán tomar medidas para mejorar el desempeño del proyecto, sin contar que se tendrá motivado y participando al todo el equipo.

- **Eliminación de re-trabajo y desperdicio:** Al no desperdiciar trabajo, se mejora la calidad y la funcionalidad del software. A la ves que el equipo no se desgastará por trabajo innecesario. Solo se aplicara el trabajo en las partes donde es necesaria.
- **Comunicación efectiva y eficiente:** Al tener una comunicación eficiente todos los elementos del equipo de desarrollo se sentirán pertenecientes a un equipo integrado, con lo que se incrementara la motivación y se minimizaran riesgos.
- **Aprendizaje de equipo:** El equipo aprenderá de cada experiencia y cada nuevo proyecto, se podrá desarrollar mas rápido y con mejor calidad, lo que conllevará una especialización del desarrollo, y tratara de llegar a la excelencia.
- **Incremento en funcionalidad, confiabilidad y desempeño del software:** al estar libre de errores el software es más confiable, al incrementar la funcionalidad se podrá eliminar el re-trabajo y el desperdicio.

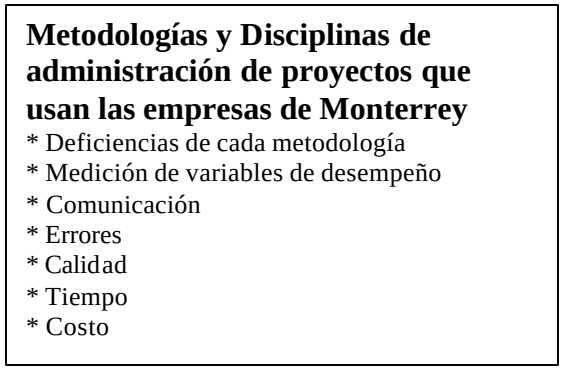


Figura 22 - Medición de las Variables de Desempeño

Al conocer los beneficios que se podrán obtener de la implementación de las mejores prácticas de proceso unificado, el siguiente paso consistirá en detectar la situación actual (metodología usada) de las empresas que desarrollan software en México, especialmente en la ciudad de Monterrey, Nuevo León.

Se desarrollara un estudio exploratorio (encuesta)orientado a los administradores de proyecto o gerentes del departamento de desarrollote software, con el objetivo de conocer el desempeño actual en el desarrollo de software, se pretende identificar las metodologías utilizadas, herramientas automatizadas que posee y la percepción de éxito en los proyecto de desarrollo de software.

El cuestionario intentara evaluar el desempeño de la metodología utilizada, así como de las diversas métricas de desempeño de desarrollo eficiente de software, las cuales se muestran en la figura 22, las cuales a juicio de algunos autores, son críticas para el éxito de los proyectos de software.

Diseño del cuestionario:
Este cuestionario se divide en 3 tipos de preguntas

1.-Se refiere a la metodología que utiliza para el desarrollo de software.
2.-Se refiere al grado de madurez de sus procesos (empleo de herramientas automatizadas), que algunos autores recomiendan como son:

- Costo
- Tiempo
- Calidad
- Comunicación
- Errores
- Requerimientos
- Las deficiencia de cada metodología

3.- La percepción del grado de éxito que tiene la empresa en los mismos procesos que se mencionaron en el párrafo anterior.

Para lograr dicho objetivo se aplicara la entrevista en empresas medianas y grandes de monterrey que desarrollan software (ya sea para uso interno o para comercializarlo), con el objetivo de identificar el grado de madurez de la metodología que utilizan

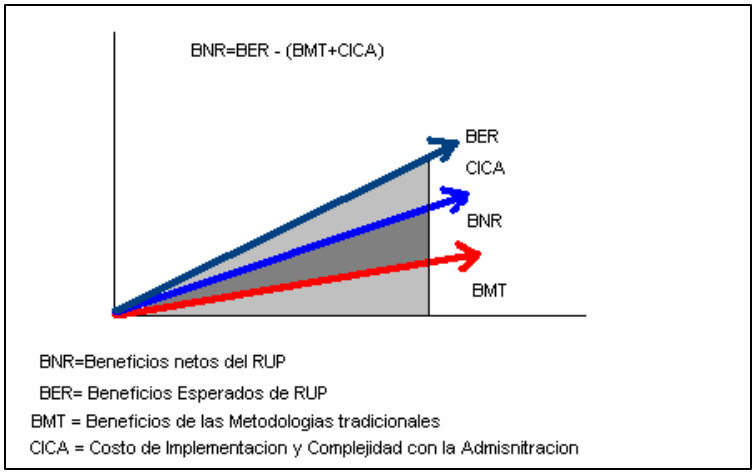


Figura 23 - Diferencias o Áreas de Oportunidad Existentes en las Empresas

Teniendo los resultados de las encuestas, se procederá a contrastarlo con los beneficios esperados por RUP, para obtener las áreas de oportunidad o beneficios de implementar una metodología de desarrollo iterativo.

Cabe mencionar que esta metodología conlleva algunos inconvenientes, los cuales se mencionan a continuación:

Costo de Implantación Alto: Al ser necesario desarrollar un ambiente adecuado para el desarrollo de software, será necesario comprar herramientas que permitan automatizar todo el ciclo de desarrollo. Dichas herramientas tiene un costo, el cual en algunos casos resulta elevado.

Administración Compleja: Al contar con un ambiente estructurado y complejo, también será necesario contar con el equipo necesario para administrar cada una de las áreas que integran la metodología, pues se considera que la metodología tiene una administración compleja.

Por lo tanto los beneficios netos de la metodología, están dados por que las empresas que implanten la metodología dispongan de recursos financieros y humanos, para absorber un costo de implantación alto y una administración compleja. Por lo regular, estas empresas serán las grandes o medianas empresas que desarrollan grandes proyectos de desarrollo de software.

Áreas de oportunidad (definición)

Las áreas de oportunidad son aquellas áreas del conocimiento en las que se propende a la solución de problemas pre-identificados de interés social y económico.

Capitulo IV. Diseño de la investigación

En este capitulo se hablara acerca del diseño de la investigación, donde de describirá el tipo de investigación que se realizara, la población a la que se dirige, las variables que se pretenden medir, los métodos y herramientas que se utilizaran y una descripción de la recolección de datos que se realizara.

4.1 Tipo de estudio

El tipo de estudio que se realizará es una investigación no experimental, este tipo de investigación según Hernández, Fernández y Baptista (2003) se define como una investigación que se realiza sin manipular deliberadamente variables. Es decir, una investigación en la que no se hace variar de forma intencional las variables independientes. En éste tipo de investigación únicamente se observa el fenómeno tal y como se da en su contexto natural, para después analizarlo.

El alcance de la investigación será descriptiva, según la clasificación de Danhke (1998) referenciado por Hernández, Fernández y Baptista (2003) define que los estudios descriptivos buscan especificar propiedades, características y perfiles importantes de personas, grupos, comunidades o cualquier otro enfoque que se someta a un análisis.

Se han adoptado diversas clasificaciones para la investigación no Experimental, por su dimensión temporal o el número de momentos o puntos en el tiempo en los cuales se recolectan datos, se clasifican en transaccionales o longitudinales. Según los mismos autores citados anteriormente es transaccional cuando la investigación se centra en:

a) Analizar cual es el nivel, estado o presencia de una o diversas variables en un momento dado;
b) Evaluar una situación, grupo, fenómeno, situación o punto en el tiempo y
c) Determinar o ubicar cuál es la relación entre un conjunto de variables en un momento.

Por lo tanto, se define que la presente investigación corresponderá a una investigación No experimental –transeccional- descriptiva, debido que se buscara identificar y describir las metodologías con que cuentan las empresas que desarrollan software en la ciudad de Monterrey, con el objetivo de identificar los errores y las posibles áreas de oportunidad que existen en la actualidad , y que puedan ayudar a recomendar una metodología que provea un proceso maduro de administración y desarrollo de proyectos de software.

4.2 Población

Para un enfoque cuantitativo, una población es el conjunto de todos los casos que concuerdan con la serie de especificaciones (seltziz 1980) es preferible para un enfoque cualitativo, establecer con claridad las características de la población, con la finalidad de delimitar cuales serán los parámetros maestrales.

De acuerdo a lo anterior, se tomo como población las empresas medianas y grandes de la cuidad de Monterrey que tengan como una de sus funciones desarrollar software, ya sea de manera interna o con fines de negocio.

4.3 Selección de la muestra.

Debido a que no existe un registro de empresas de desarrollo de software en monterrey; la muestra para la presente investigación esta determinada por la oportunidad de encontrar empresas con las características requeridas para la investigación.

Las características de las empresas

- Empresas medianas o grandes.
- Radiquen en la ciudad de Monterrey.
- Tengan un departamento de Desarrollo de software (ya sea para uso interno o para comercializar).
- Cuenten con los recursos necesarios para comprar herramientas.
- Dispongan de la infraestructura necesaria (personal capacitado) para administrar y manejar una administración compleja de la metodología.

4.4 Definición de variables

Con el objetivo de medir las áreas de oportunidad, las variables que se utilizaron para medir el grado de madurez de los procesos que debe tener un proceso de desarrollo maduro, son los siguientes:

Procesos de Estimación de costos: Es el proceso mediante el cual se estima el dinero que se gastara en la realización del proceso.

Aseguramiento de Calidad: Es el procesos que se refiere a proporcionar las herramientas que permitan al equipo cumplir con la funcionalidad del software, sea confiable y tenga un buen desempeño.

Estimación de Tiempo: Es el proceso mediante el cual el tiempo de desarrollo del proyecto es estimado y es cumplido.

Eliminación de errores: se refiere al proceso mediante el cual se crea un ambiente de pruebas, para tratar de eliminar la mayor cantidad de fallas durante el desarrollo del software.

Asegurar la Comunicación: Es el proceso mediante el cual se asegura que las personas involucradas en el desarrollo del software, estén al tanto de todo lo que involucra el desarrollo.

Minimización de Riesgos: Es el proceso mediante el cual se trata de minimizar la probabilidad de que una falla ocurra y afecte el desempeño del proyecto o sistema.

Éxito en proyectos: Se refiere al grado en que el proyecto sea completado en el tiempo, presupuesto y la calidad estimada inicialmente.

Aprendizaje del equipo: Es la acción de guardar los conocimientos, con el propósito de utilizarlos con provecho en el futuro.

Re-trabajo: la acción de volver a realizar una actividad de manera repetitiva.

4.5 Métodos y Herramientas

Se realizara un cuestionario el cual se divide en 4 secciones, las cuales recabaran los distintos tipos de información, cada sección se describe a continuación:

La **primera parte** se refiere a los datos personales de quien responde la encuesta, entre los datos que se solicitan están: nombre, empresa en la que labora, puesto que desempeña, teléfono y una dirección de correo electrónica para contacto.

La S**egunda parte** básicamente se busca conocer la metodología que utiliza la empresa actualmente (Si utiliza una), si conoce las deficiencias y beneficios de dicha metodología, si tienen acciones para controlar dichas deficiencias y explotar los beneficios. Estaría dispuesto a invertir en una nueva metodología que le provea de mejores beneficios para el desarrollo exitosos de software.

La **Tercera parte** Comprende una serie de preguntas cerradas para detectar el nivel de automatización de los procesos que siguen para administrar riesgos, costos, tiempos, requerimientos así como la comunicación.

Y la **última parte** son preguntas también cerradas que tienen como objetivo evaluar el nivel de éxito que tienen las empresas en la medición de métricas de desempeño, como son: riesgos, costos, tiempos, requerimientos, y comunicación.

Esta encuesta recolectara la información que se analizara para de determinar el grado de madurez que tienen de los procesos de desarrollo de software, con le objetivo de asociarlo con el grado de éxito que tienen, para encontrar las áreas de oportunidad que cuentan las empresas de Monterrey que desarrollan software.

4.6 Estrategia de recolección de datos.

Se realizara una búsqueda en el directorio telefónico, con el objetivo de identificar y establecer contacto con la empresas que desarrollan software en la cuidad de monterrey Nuevo León.

La estrategia de recolección de datos se realizara mediante un cuestionario, la cual se desarrollara como página Web en un servidor de Internet como lo muestra la figura 24, para facilitar el acceso a los entrevistados.

Con los datos de las empresas a las que se invitaría a realizar las encuestas, se procederá a enviarles un correo electrónico, donde se les explicara el objetivo y el propósito de la presente investigación, así mismo se les extenderá una invitación para que participen en dicho ejercicio. El total de empresas que contestaron la encuesta fue 20 .

En la siguiente sección se analizara a detalle la información que se obtuvo.

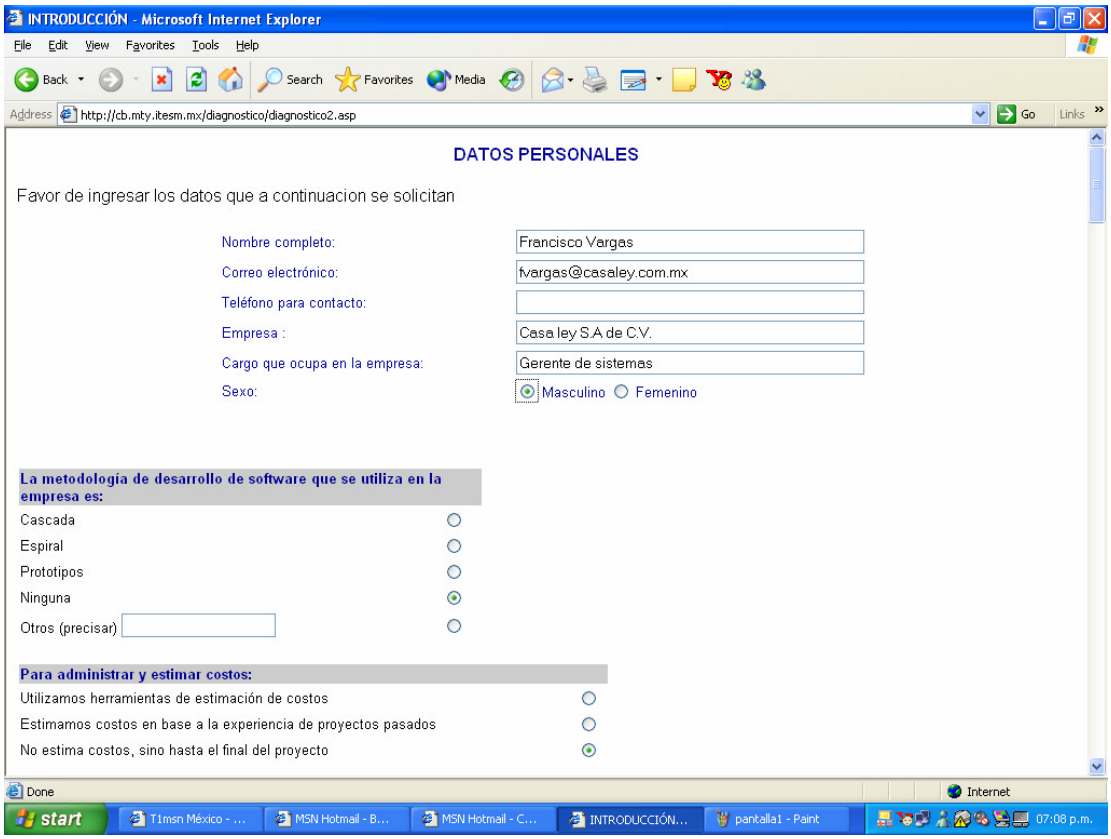


Figura 24 - Pagina Web de la Encuesta.

Capitulo V. Análisis de Datos

Una vez que las encuestas fueron contestadas, es el turno de hacer el análisis de los datos obtenidos, en esta sección se explicara a detalle el análisis que se realizo de los datos.

En la tabla 7 se puede observar el condensado de respuesta de las 20 empresas que fueron entrevistadas, Para el análisis de los datos se procederá a analizar los resultados por secciones las cuales se describen a continuación:

	Pregunta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	***Metodología de desarrollo***	*1*	*3*	*0*	*5*	*3*	*5*	*-1*	*5*	*5*	*1*	*0*	*1*	*0*	*3*	*5*	*1*	*5*	*3*	*5*	*1*
2	***Administrar y estimar costos***	*3*	*5*	*1*	*5*	*3*	*1*	*3*	*1*	*3*	*3*	*1*	*1*	*1*	*3*	*1*	*3*	*3*	*1*	*1*	*3*
3	***Administrar y estimar tiempos***	*5*	*3*	*1*	*5*	*3*	*3*	*5*	*5*	*3*	*3*	*3*	*3*	*3*	*3*	*3*	*5*	*3*	*3*	*3*	*3*
4	***Calidad de software***	*3*	*3*	*3*	*5*	*3*	*5*	*5*	*5*	*3*	*5*	*3*	*3*	*3*	*3*	*1*	*5*	*3*	*3*	*1*	*5*
5	***Minimizar riesgos***	*3*	*5*	*3*	*3*	*1*	*5*	*5*	*5*	*3*	*5*	*3*	*1*	*3*	*3*	*3*	*5*	*3*	*3*	*1*	*5*
6	***Administrar cambios***	*1*	*5*	*3*	*3*	*1*	*1*	*5*	*5*	*5*	*5*	*3*	*3*	*3*	*3*	*1*	*3*	*5*	*1*	*1*	*5*
7	***Comunicación entre involucrados***	*5*	*3*	*3*	*5*	*3*	*5*	*5*	*5*	*5*	*5*	*3*	*3*	*3*	*3*	*5*	*5*	*5*	*5*	*1*	*5*
8	***Administrar requerimientos***	*5*	*5*	*1*	*5*	*1*	*3*	*5*	*5*	*5*	*5*	*3*	*5*	*3*	*3*	*5*	*1*	*5*	*5*	*3*	*5*
9	***Deficiencias de metodología***	*1*	*5*	*5*	*3*	*1*	*5*	*5*	*5*	*5*	*5*	*3*	*5*	*5*	*5*	*3*	*5*	*5*	*5*	*3*	*5*
10	***Oportunidad de metodología***	*3*	*3*	*5*	*5*	*5*	*5*	*5*	*5*	*5*	*5*	*3*	*5*	*3*	*5*	*3*	*3*	*3*	*5*	*5*	*3*
11	***R Estimaron de Costos***	*2*	*4*	*4*	*4*	*2*	*3*	*4*	*3*	*4*	*4*	*6*	*2*	*3*	*4*	*1*	*7*	*4*	*7*	*2*	*4*
12	***R Estimaron de Tiempos***	*1*	*3*	*3*	*4*	*1*	*4*	*4*	*5*	*5*	*3*	*5*	*4*	*3*	*3*	*4*	*2*	*4*	*4*	*2*	*4*
13	***R Asegurar Calidad***	*4*	*3*	*4*	*3*	*1*	*2*	*5*	*5*	*5*	*5*	*5*	*4*	*6*	*4*	*2*	*4*	*4*	*4*	*2*	*5*
14	***R Minimizar Riesgos***	*3*	*3*	*4*	*4*	*1*	*3*	*5*	*5*	*5*	*4*	*1*	*2*	*6*	*3*	*1*	*4*	*4*	*3*	*1*	*5*
15	***R Administrar Cambios***	*1*	*3*	*4*	*5*	*2*	*2*	*5*	*3*	*5*	*5*	*1*	*3*	*3*	*5*	*1*	*2*	*2*	*4*	*2*	*3*
16	***R Asegurar Comunicación***	*4*	*3*	*4*	*4*	*1*	*4*	*5*	*5*	*4*	*4*	*4*	*4*	*6*	*4*	*3*	*4*	*5*	*5*	*3*	*5*
17	***R Administrar Requerimientos***	*1*	*3*	*1*	*5*	*2*	*2*	*5*	*5*	*6*	*4*	*5*	*3*	*3*	*3*	*2*	*4*	*4*	*3*	*2*	*5*

Tabla 7 - Condensado de los datos de las encuestas.

Análisis de Preguntas de la metodología: las preguntas 1, 9, y 10 están enfocadas a identificar la metodología que utiliza la empresa, también intentara identificar si la empresa conoce las ventajas y desventajas de la metodología que utiliza.

Análisis de datos de Grado de automatización de procesos: las preguntas 2, 3,4, 5, 6,7 y 8 de la encuesta están elaboradas para conocer el grado de automatización de procesos de desarrollo de software.

Análisis de las preguntas de Éxito del desarrollo: en las preguntas 11, 12, 13, 14, 15 y 16 están elaboradas para conocer el grado de éxito que tienen la empresa en cuestión de los procesos que los expertos son críticos para un buen desarrollo de proyectos de software.

A continuación se describen los resultados obtenidos en cada una de las preguntas de la encuesta aplicada. (Ver Anexo 1), así como una breve interpretación del resultado.

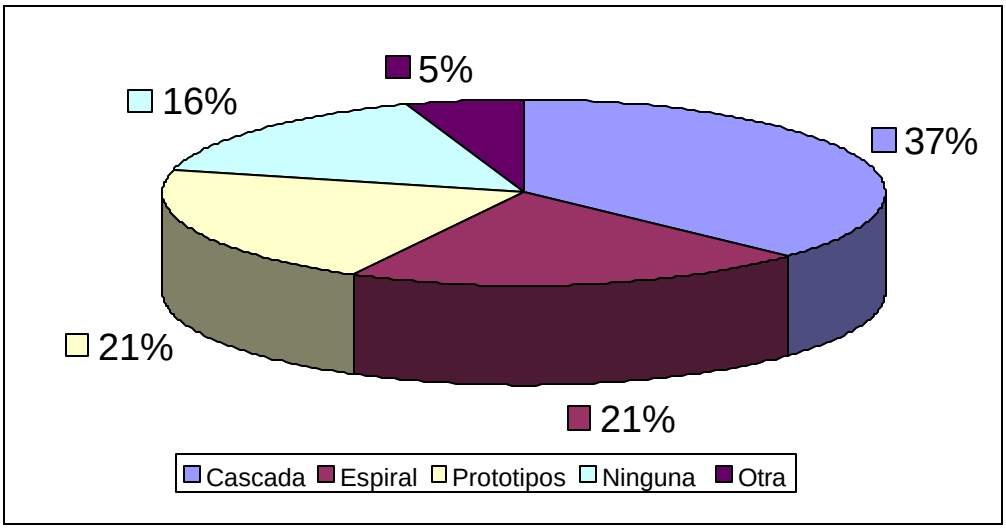
Análisis de Preguntas de la metodología:

1.- ¿Que metodología de desarrollo de software se utiliza en la empresa?

1	***Metodología de desarrollo***	*1*	*3*	*0*	*5*	*3*	*5*	*-1*	*5*	*5*	*1*	*0*	*1*	*0*	*3*	*5*	*1*	*5*	*3*	*5*	*1*

Tabla 8 - Detalle de la pregunta 1.

Cascada : 7
Espiral : 4
Prototipos : 5
Ninguna : 3
Otra : 1
Total : 20



Grafica 1 - Resultados Pregunta 1.

La grafica 1 representa las respuestas a la pregunta 1, se puede observar que un 80 % de las empresas entrevistadas utilizan una metodología clásica de desarrollo (Cascada, Espiral o Prototipos), las cuales como ya se ha visto en la presente investigación presentan algunos errores como: estimación de costos no acertada, mala estimación de tiempos y una pobre calidad en la funcionalidad del software. También se observa que el 15% de las empresas no cuentan con una metodología para el desarrollo eficiente de software.

Se puede deducir que existe una gran oportunidad para implementar una metodología que provea un proceso de desarrollo lo suficientemente maduro como para asegurar la calidad del software y atacar los errores que implican las metodologías clásicas

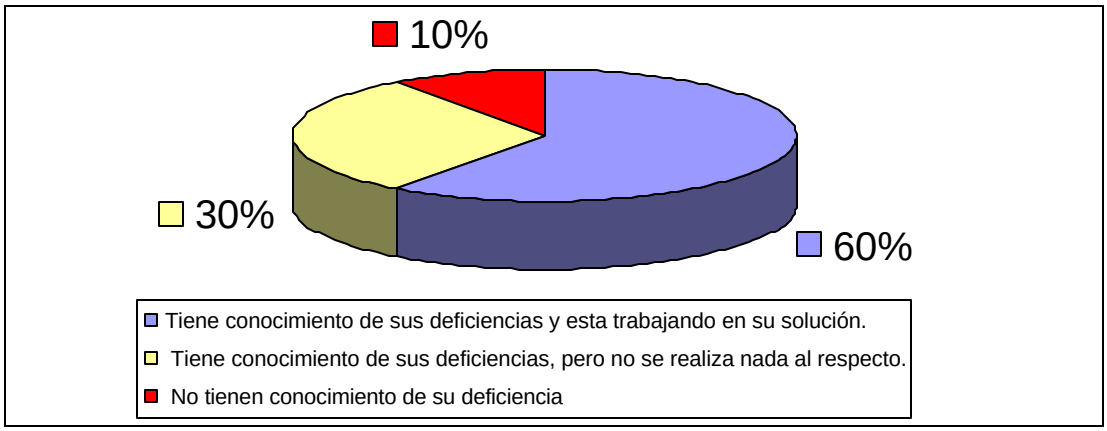
.

9.- Con respecto a las deficiencias de la metodología de desarrollo de software que utiliza

Tiene conocimiento de sus deficiencias y esta trabajando en su solución.	12
Tiene conocimiento de sus deficiencias, pero no se realiza nada al respecto.	6
No tienen conocimiento de su deficiencia	2
Total	20

9	***Deficiencias de metodología***	***1***	***5***	***5***	***3***	***1***	***5***	***5***	***5***	***5***	***5***	***3***	***5***	***5***	***5***	***3***	***5***	***5***	***5***	***3***	***5***

Tabla 9 - Detalle de la pregunta 9



Grafica 2 - Resultados Pregunta 9

Como se puede observar en la grafica correspondiente a la pregunta numero 2, el 60% de las empresas tienen conocimiento de las deficiencias que tiene la metodología que utilizan, y esta trabajando en un plan para solucionar o atacar esos errores. Se puede deducir que estas empresas están concientes de que existen muchas posibilidades de desarrollar software de manera eficiente.

Otro 30% de las empresas contestaron que tiene conocimiento de las deficiencias que tiene su metodología, pero no tienen un plan de acciones para contrarrestarlas, en estas empresas existe la posibilidad de hacerles ver la oportunidad que existe en aplicar metodologías que proveen un marco procesos maduro de desarrollo de software.

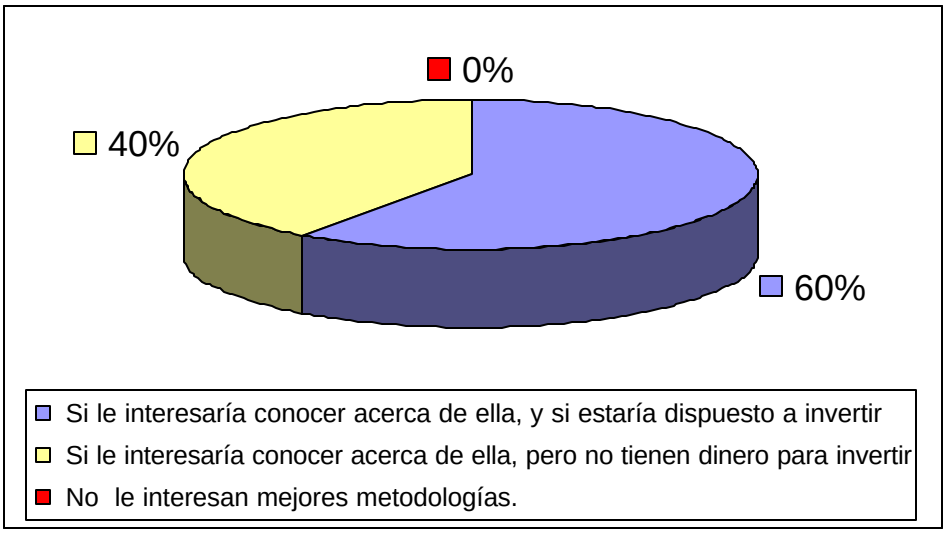
Por ultimo el 10% restante de las empresa comenta que no tienen conocimiento de las deficiencias de su metodología por lo que las posibilidades de mejora puedes ser bastantes, pues estas ultimas ni siquiera tienen detectados posibles fallas que pudieran estar teniendo.

10.- En caso de que exista una metodología que ayude a mejorar las métricas de desempeño e incremento el grado de éxito de sus proyectos de desarrollo de software

Si le interesaría conocer acerca de ella, y si estaría dispuesto a invertir	12
Si le interesaría conocer acerca de ella, pero no tienen dinero para invertir	8
No le interesan mejores metodologías.	0
Total	16

10	***Oportunidad de metodología***	*3*	*3*	*5*	*5*	*5*	*5*	*5*	*5*	*5*	*5*	*3*	*5*	*3*	*5*	*3*	*3*	*3*	*5*	*5*	*3*

Tabla 10 - Detalle de la pregunta 10



Grafica 3 - Resultados Pregunta 10

En la grafica 3 se representan los resultados de la pregunta numero 10, en la que se puede observar que del total de las empresas encuestadas 60% si les interesa conocer acerca de nuevas metodologías que ayuden a incrementar el éxito de los proyectos y también estaría dispuesto a invertir en ellas, mientras que el 40% de las empresa comentan que si están interesados en mejores metodologías, pero que no cuentan con el capital económico para invertir en ellas.

Se puede deducir que las empresas están conscientes de las oportunidades existentes al adquirir herramientas para proveer un ambiente de desarrollo de software efectivo.

Análisis de los Datos de Automatización de herramientas

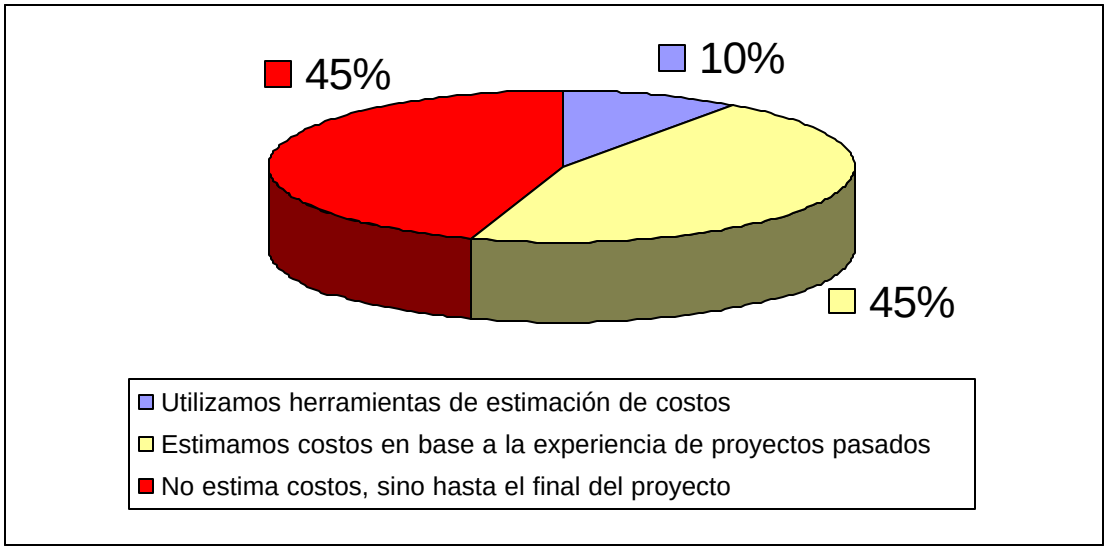
2.- Para administrar y estimar costos:

2	***Administrar y estimar costos***	*3*	*5*	*1*	*5*	*3*	*1*	*3*	*1*	*3*	*3*	*1*	*1*	*1*	*3*	*1*	*3*	*3*	*1*	*1*	*3*

Tabla 11 - Detalle de la pregunta 2.

Las respuestas fueron:

Utilizamos herramientas de estimación de costos	2
Estimamos costos en base a la experiencia de proyectos pasados	9
No estima costos, sino hasta el final del proyecto	9
Total	20



Grafica 2 - Resultados Pregunta 2.

En la grafica 2 se representa las respuestas a la pregunta número 2, en la que se observa que solo un 10% hace uso de herramientas de estimación de costos. Pero se puede observar que un 45% estima costos en base a su experiencia es proyectos pasados, este es un grave error pues se recordara que ningún proyecto es igual a otro y cada proyecto es distinto en diversos aspectos como tiempo costo y funcionalidad.

También se puede observar que un 45% estiman costos hacia el final del proyecto. Este es un grave error debido a que un proyecto se debe manejar como una inversión, y es necesario estimar costos para verificar la rentabilidad del proyecto. Al no contar con una estimación de costos el proyecto se puede resultar muy costoso.

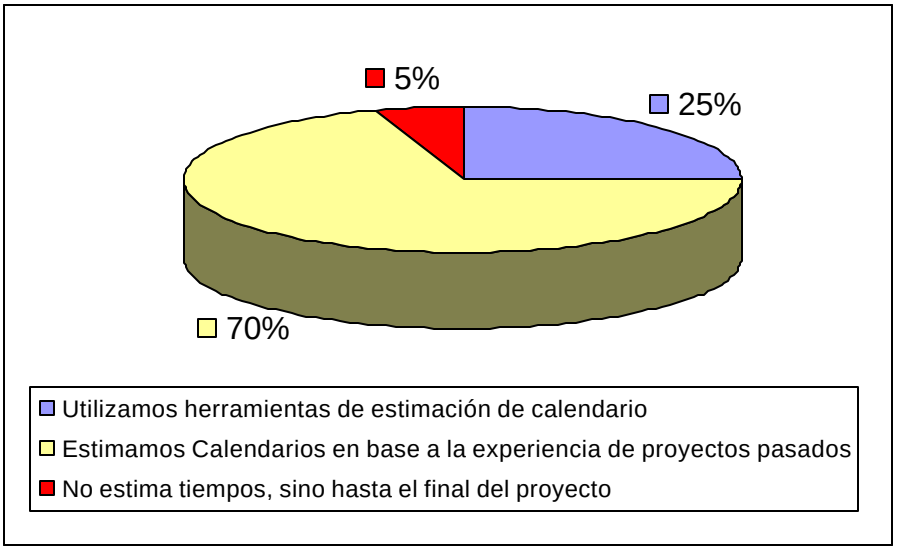
Se puede concluir que el 55% de las empresas presentan la oportunidad de instrumentar el proceso de estimación de costos, lo que les ayudara a tener un mayor control sobre los recursos tanto financieros como humanos.

3.- Para Administrar y estimar tiempos:

3	***Administrar y estimar tiempos***	*5*	*3*	*1*	*5*	*3*	*3*	*5*	*5*	*3*	*3*	*3*	*3*	*3*	*3*	*3*	*5*	*3*	*3*	*3*	*3*

Tabla 12 - Detalle de la pregunta 3.

Utilizamos herramientas de estimación de calendario	5
Estimamos Calendarios en base a la experiencia de proyectos pasados	14
No estima tiempos, sino hasta el final del proyecto	1
Total	20



Grafica 3 - Resultados Pregunta 3.

En la grafica 3 se representa las respuestas a la pregunta número 3. Se puede observar que solo una cuarta parte de las empresas utilizan una herramienta de estimación y administración de calendario.

Las empresas que estiman un calendario en base a sus experiencias pasadas que son el 70% de las entrevistadas tiene la oportunidad de implementar un instrumento automatizado para mejorar el desempeño de la estimación de calendario. Con ayuda de un instrumento de estimación d e calendario la tarea de hará mas sencilla de implementar y administrar.

Solo un 5% de las empresas no estiman el calendario, lo cual se trata de un grave error pues se tiene el peligro que los proyectos se atrasen y se alarguen en su tiempo de desarrollo, lo cual repercutiría en: desgaste del equipo, mayores gastos, frustración del equipo, desgaste del equipo y otras cosas.

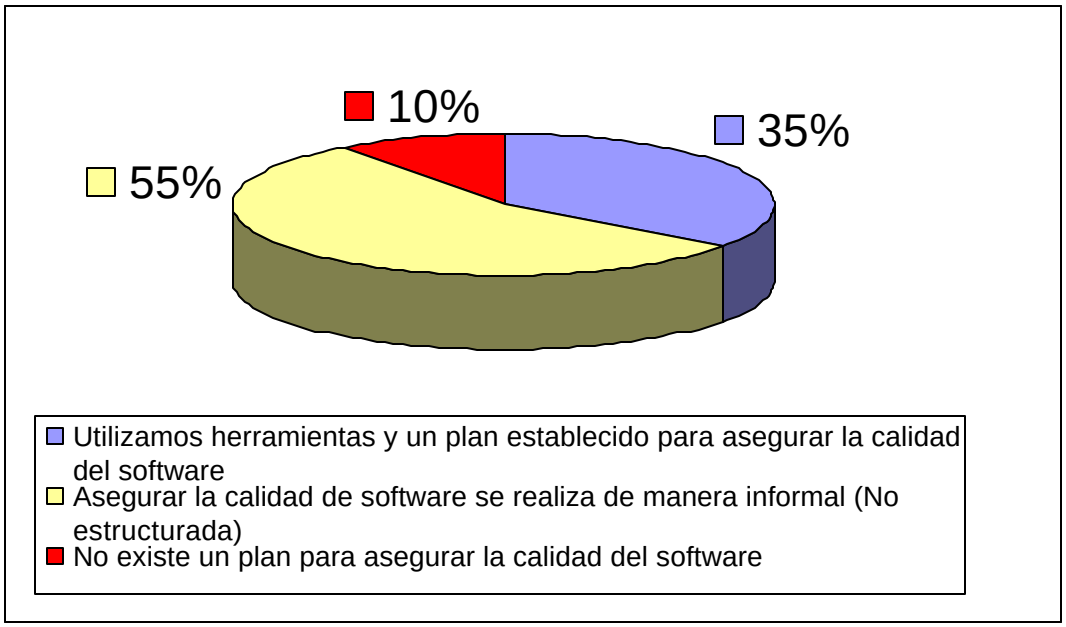
Se puede deducir que en el 75% de las empresas tiene una excelente oportunidad al implantar una estimación de costos utilizando una herramienta automatizada. Lo cual les ayudara a tener un control más estricto del calendario y de los recursos de la empresa.

4.- Para asegurar la calidad del software

Utilizamos herramientas y un plan establecido para asegurar la calidad del software	7
Asegurar la calidad de software se realiza de manera informal (No estructurada)	11
No existe un plan para asegurar la calidad del software	2
Total	20

4	*Calidad de software*	*3*	*3*	*3*	*5*	*3*	*5*	*5*	*5*	*3*	*5*	*3*	*3*	*3*	*3*	*1*	*5*	*3*	*3*	*1*	*5*

Tabla 13 - Detalle de la pregunta 4



Grafica 4 - Resultados Pregunta 4

En la grafica 4 se representa las respuestas a la pregunta numero 4: En esta grafica se observa que solo un 35% de las empresas hacen uso de herramientas automatizadas para asegurar la calidad del software.

Pero también se detecta que un 55% de dichas empresas cuentan con un proceso no automatizado para asegurar la calidad, lo cual da la pauta para hacer uso de herramientas automatizadas para asegurar la calidad. Esto tendrá como resultado un mejor desempeño por parte del equipo de desarrollo.

Un 10% de las empresas no cuentan con un plan establecido para asegurar la calidad del software, en este tipo de empresas la oportunidad es muy buena para mejorar el desempeño.

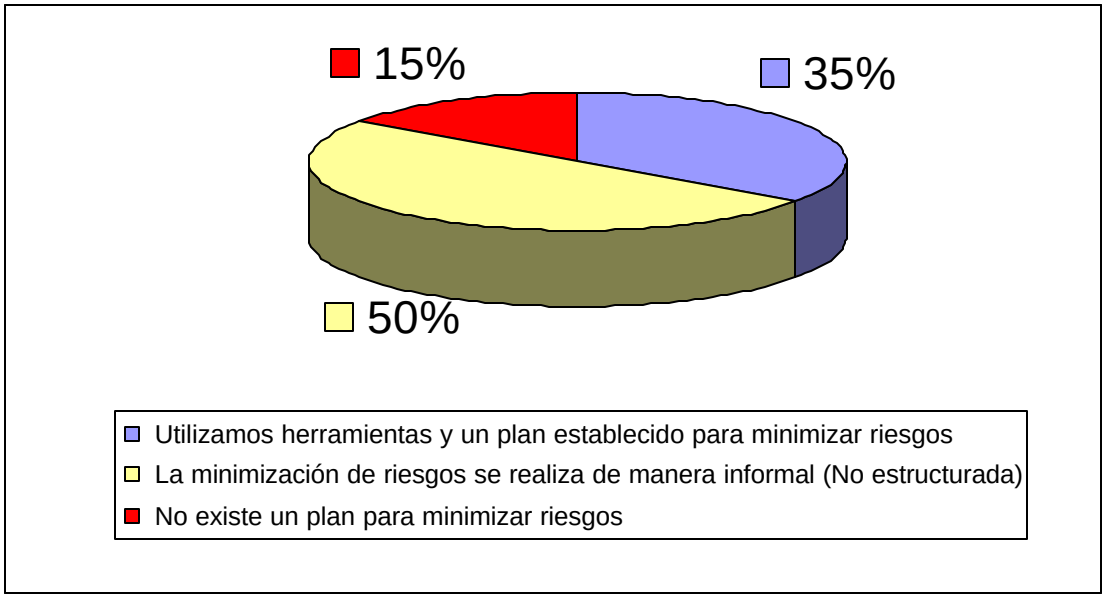
En este sentido las empresas tienen una basta oportunidad de instrumentar el aseguramiento de calidad, pues les traerá beneficios como: eliminación de desperdicio. Mejor desempeñó por parte de los desarrolladores. Menores gastos, cumplimiento con tiempos y costos y lo mas importante cumplir con la funcionalidad del software desarrollado.

5.- Para minimizar riesgos:

Utilizamos herramientas y un plan establecido para minimizar riesgos	7
La minimización de riesgos se realiza de manera informal (No estructurada)	10
No existe un plan para minimizar riesgos	3
Total	20

5	***Minimizar riesgos***	*3*	*5*	*3*	*3*	*1*	*5*	*5*	*5*	*3*	*5*	*3*	*1*	*3*	*3*	*3*	*5*	*3*	*3*	*1*	*5*

Tabla 14 - Detalle de la pregunta 5



Grafica 5 - Resultados Pregunta 5

En la grafica 5 se observa que solo un 35 % de las empresas que participaron en este ejercicio utilizan una herramienta y un plan establecido para asegurar que los riesgos sean minimizados al máximo. Mientras que el 50% de las empresas entrevistadas comenta que existe un plan para administrar riesgos de manera informal por lo que existe una oportunidad de mejorar al implementar una o varias herramientas automatizadas que lograran que el proyecto tenga bajos incertidumbre y pocas amenazas.
Y por ultimo el 15% restante de las empresas no cuentan con plan para minimizar y contrarrestar riesgos a lo cual se hace obvio el área de oportunidad con que cuentan dichas empresas y estar expuestas al no contar con un plan para administrar riesgos.

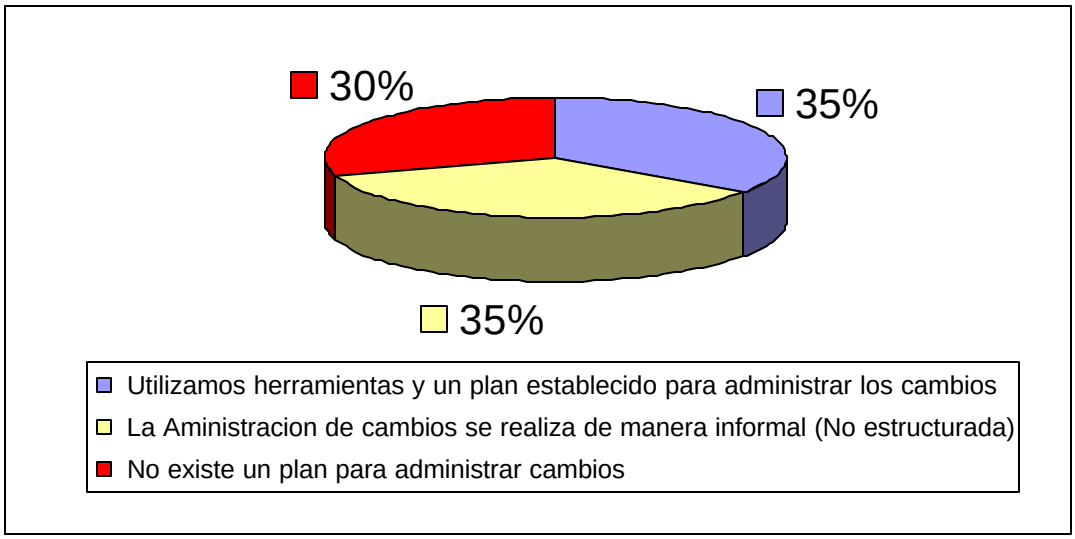
Se puede concretar que existe una excelente oportunidad para las empresas en instrumentar un programa para minimizar riesgos. El cual tendrá beneficios como: un control cercano con los recursos, calendario siempre predecible, costos predecibles, y un desempeño favorable por parte del equipo de desarrollo.

6.- Para administrar los cambios:

Utilizamos herramientas y un plan establecido para administrar los cambios	7
La minimización de riesgos se realiza de manera informal (No estructurada)	7
No existe un plan para administrar cambios	6
Total	20

6	*Administrar cambios*	*1*	*5*	*3*	*3*	*1*	*1*	*5*	*5*	*5*	*5*	*3*	*3*	*3*	*3*	*1*	*3*	*5*	*1*	*1*	*5*

Tabla 15 - Detalle de la pregunta 6



Grafica 6 - Resultados Pregunta 6

En 6 se observar que el 35% de las empresas entrevistadas cuentan con un plan instrumentado para la administración de cambios. Estas empresas presentan la oportunidad de incrementar el desempeño al administrar cambios. Un 35% de las empresas comenta que cuentan con un plan para administración de cambios de manera informal, la oportunidad para estas empresas estaría en instrumentar el proceso con el objetivo de hacerlo mas eficiente

Por ultimo hay un 35% de las empresas que no cuentan con un plan para administrar cambios, estas empresas presentan el peligro latente de perder cambios, realizar trabajos doble, no liberar cambios en nuevas versiones entre otros. Estas empresas cuentan con una gran oportunidad de mejorar significativamente el desempeño de administración de cambios al instrumentar un plan para la administración de cambios.

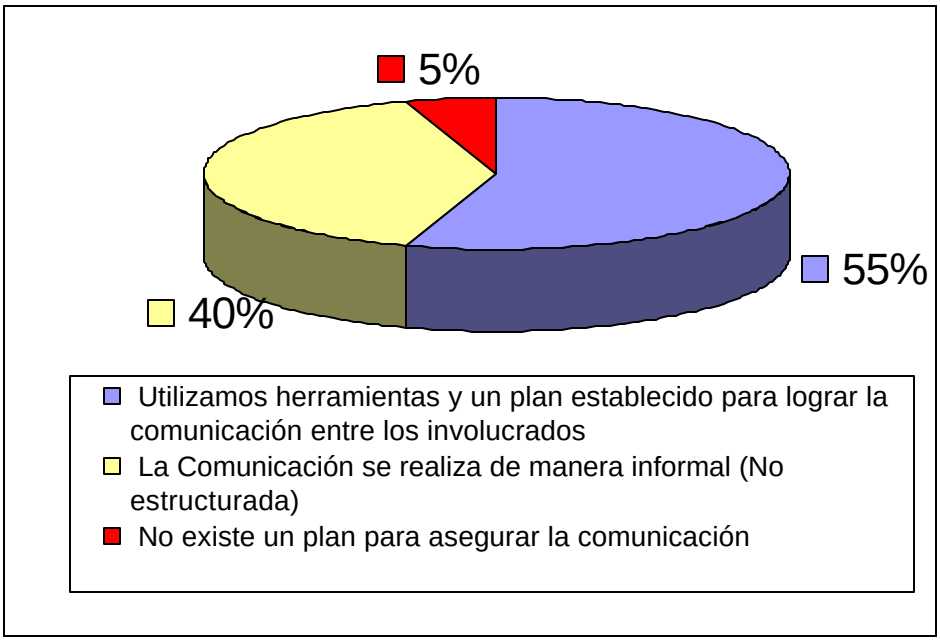
Los beneficios de tener una procesos automatizado de control de cambios son: Implementación de todos los cambios, documentación de cambios, seguimiento de cambios, establecimiento de prioridades, y todo esto se hace de manera automática con lo cual se ahorra en una administración compleja por parte de administrador del proyecto, también ayuda en el sentido que todos trabajan con un mismo enfoque y mejora la comunicación

7.- Para asegurar la comunicación entre los involucrados:

Utilizamos herramientas y un plan establecido para lograr la comunicación entre los involucrados	11
La Comunicación se realiza de manera informal (No estructurada)	8
No existe un plan para asegurar la comunicación	1
Total	20

7	***Comunicación entre involucrados***	*5*	*3*	*3*	*5*	*3*	*5*	*5*	*5*	*5*	*5*	*3*	*3*	*3*	*3*	*5*	*5*	*5*	*5*	*1*	*5*

Tabla 16 - Detalle de la pregunta 7



Grafica 7 - Resultados Pregunta 7

Lo que se puede observar en la grafica 7, es que un 55% de las empresas poseen una herramienta automatizada para asegurar la comunicación entre los involucrados. Un 40% los entrevistados comento que ellos cuentan con un plan informal para la comunicación entre involucrados, en este caso existiría una oportunidad de que estas empresas puedan mejorar el proceso mediante al adquisición e implantación de una proceso automatizado para que le comunicación entre involucrados pueda fluir con regularidad

Sin lugar a dudas la mayor oportunidad estará en el 10% restante de las empresas que no cuentan con un plan específico para asegurar la comunicación entre involucrados.

Al implantar una herramienta automatizada para asegurar la comunicación traerá beneficios como: Todos los involucrados estarán al tanto de los avances, cambios, requerimientos, en pocas palabras todos se involucraran mas en el proyecto al sentirse pertenecientes al grupo de desarrollo.

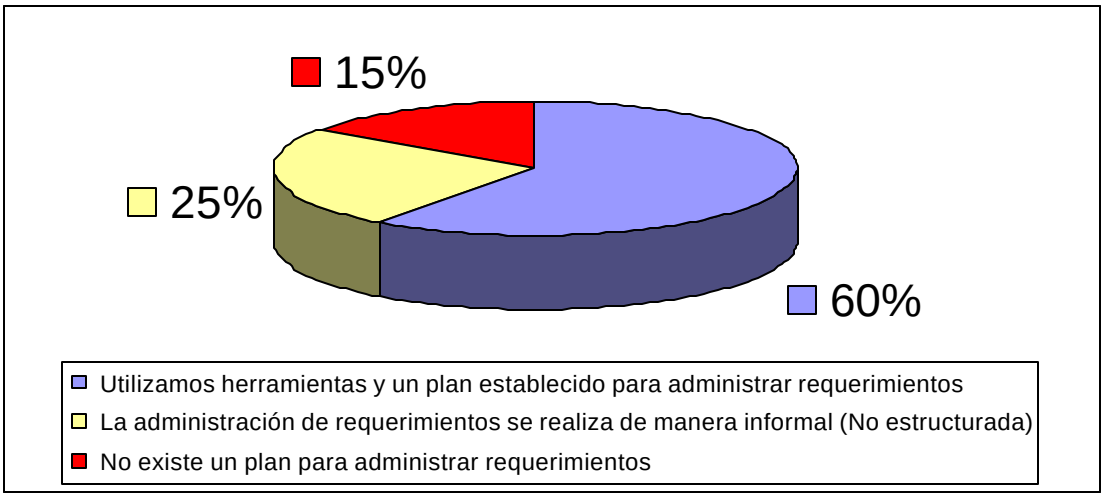
Sin duda alguna las existe una oportunidad para estas empresas el instrumentar un proceso de desarrollo de software eficiente.

8.- Para administrar requerimientos:

Utilizamos herramientas y un plan establecido para administrar requerimientos	12
La administración de requerimientos se realiza de manera informal (No estructurada)	5
No existe un plan para administrar requerimientos	3
Total	20

8	***Administrar requerimientos***	*5*	*5*	*1*	*5*	*1*	*3*	*5*	*5*	*5*	*5*	*3*	*5*	*3*	*3*	*5*	*1*	*5*	*5*	*3*	*5*

Tabla 17 - Detalle de la pregunta 8



Grafica 8 - Resultados Pregunta 8

En la grafica8 se presentan los resultados en cuanto a la automatización de administración de requerimientos, un 60% de las empresas comentan que ya disponen de una herramienta automatizada para administrar este proceso, el cual trae beneficios como: identificación y administración de requerimientos, minimizar riesgos de no saber algún requerimiento y seguimiento a cada requerimiento hecho por el usuario. En este sentido estas empresas ya cuentan con dichos beneficios, pero siempre existe la posibilidad de incrementar el desempeño y posterior mejora del proceso, para las empresas.

El 25% que solo cuentan con un plan informal presentan la oportunidad de instrumentar el procesos con el objetivo de volverlo mas automático y que no recaiga en la experiencia del administrador, lo cual le incrementara considerablemente su desempeño.
Por otra parte esta el 15% de las empresas las cuales no cuentan con un plan para administrar los requerimientos, en este sentido las empresas tendrán la oportunidad de crear un escenario para administrar adecuadamente los requerimientos.

No cabe duda que el 40% de las empresas presentan una gran oportunidad de mejorar el desempeño de sus procesos administración de requerimientos. El cual con ayuda de una herramientas automatizada proveerá beneficios como: Mejor control de los requerimientos, identificación y seguimiento de todos los requerimientos, y mejorar la comunicación entre los desarrolladores y los administradores.

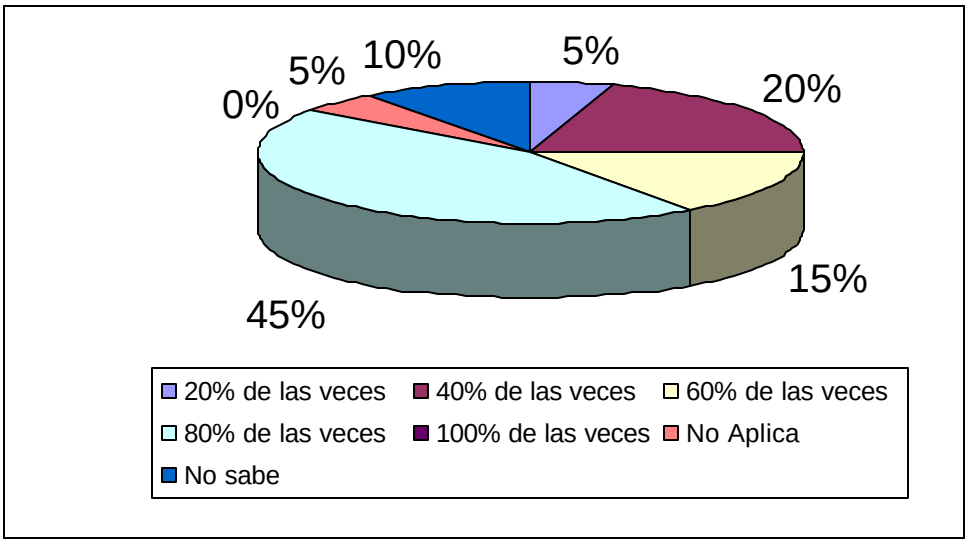
Análisis de los Datos con respecto al grado de éxito en sus procesos

11- Las estimaciones de los presupuestos de nuestros proyectos tienden en general a ser acertadas (costo)

11	***R Estimaron de Costos***	*2*	*4*	*4*	*4*	*2*	*3*	*4*	*3*	*4*	*4*	*6*	*2*	*3*	*4*	*1*	*7*	*4*	*7*	*2*	*4*

Tabla 18 - Detalle de la pregunta 11 (Resultados Estimar costos)

En un 20% de las veces: 1
En un 40% de las veces: 4
En un 60% de las veces: 3
En un 80% de las veces: 9
En un 100% de las veces: 0
No Aplica : 1
No sabe : 2
Total : 20



Grafica 9 – Percepción de éxito en estimación de costos

En la grafica 11 se puede observar que en cuestión de estimar costos, no existe empresa alguna que tenga un éxito asegurado, un 5% de las empresas solo estima bien el 20% de sus proyectos, un 20% lo estima en el 40% de sus proyectos, un 15% de las empresas tiene éxito en el 60% de sus proyectos, el 45 % las empresas encuestadas tiene éxito en 8 de cada 10 proyectos(80%) y ninguna empresa de las entrevistadas lo tiene éxito total en sus proyectos.

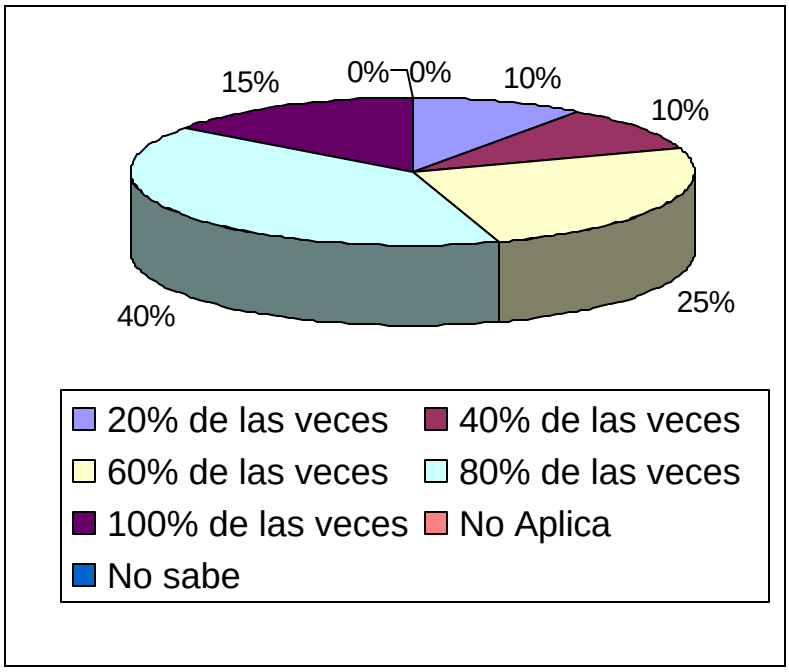
Se podría deducir que todas las empresas tienen una oportunidad de mejora en cuanto a mejorar el proceso de estimación de costos se refiere, algunas en mayor grado que otras. El objetivo de todas las empresas será crear un proceso eficiente para tener una estimación de costos acertada en el 100% de sus proyectos. Y las beneficiara en tener un control estricto sobre los recursos del proyecto, a su vez que incrementa la utilidad del proyecto.

12.- Los proyectos en los que participo se terminan en el tiempo estimado. (Tiempo)

En un 20% de las veces: 2
En un 40% de las veces: 2
En un 60% de las veces: 5
En un 80% de las veces: 8
En un 100% de las veces: 3
No Aplica : 0
No Sabe : 0
Total : 20

12	***R Estimaron de Tiempos***	*1*	*3*	*3*	*4*	*1*	*4*	*4*	*5*	*5*	*3*	*5*	*4*	*3*	*3*	*4*	*2*	*4*	*4*	*2*	*4*

Tabla 19 Detalle de la pregunta 12



Grafica 10 – Percepción de éxito en la estimación de tiempos

En la grafica 12 se encuentran las respuestas a la pregunta numero 12, en la cual se aprecia que un 10% de las empresas entrevistadas tiene un porcentaje de éxito del 20%, mientras que otro 10% lo hace con un 40%, una cuata parte realiza una estimación de tiempos exitosa en el 60% de sus proyectos, el 40% tiene un 60% de éxito y por ultimo el 15% de las empresas realiza su estimación de tiempos de manera exitosa en todos sus proyectos.

A excepción de 15% de las empresas que realizan la estimación de costos con éxito, las demás empresas cuentan suficientes motivos para implantar un instrumento de estimación de costos que les provea la seguridad de administrar y estimar el calendario.

Como ya se ha comentado una buena estimación de calendario beneficia en maximizan los recursos de la empresa y a su vez que se terminan los proyectos en el tiempo justo.

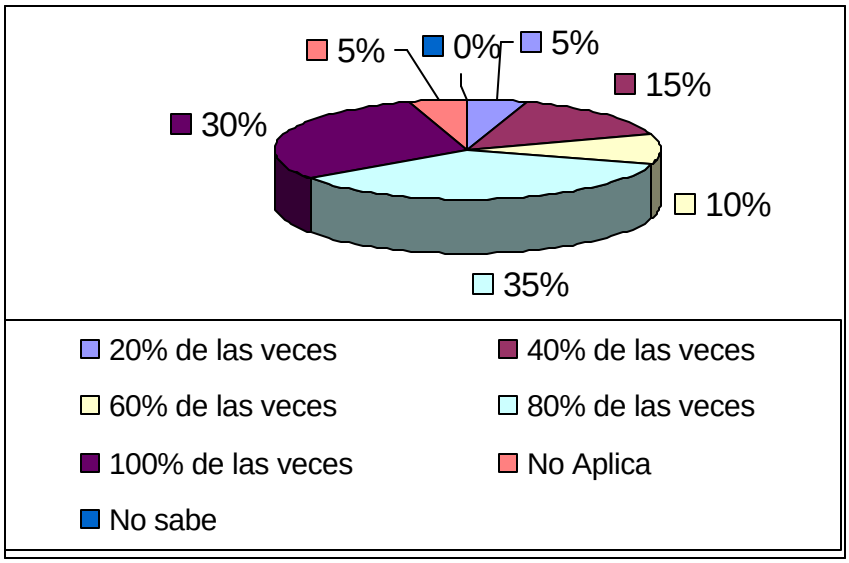
13.- ¿Para diseñar un sistema tomamos en cuenta las diferentes dimensiones de la calidad? Como

Funcionalidad - Cubrir las necesidades del usuario.
Facilidad de uso- esfuerzo requerido para aprender su uso.
Confiabilidad - Soporte a fallas y caídas
Eficiencia - Velocidad de respuesta, empleo de recursos
Capacidad de mantenimiento - Minimizar esfuerzo requerido para administrar, modificar o validar.
Portabilidad - adaptación del software a diferentes ambientes

13	*R Asegurar Calidad*	*4*	*3*	*4*	*3*	*1*	*2*	*5*	*5*	*5*	*5*	*5*	*4*	*6*	*4*	*2*	*4*	*4*	*4*	*2*	*5*

Tabla 20 - Detalle de la pregunta 13

En un 20% de las veces: 1
En un 40% de las veces: 3
En un 60% de las veces: 2
En un 80% de las veces: 7
En un 100% de las veces: 6
No Aplica : 1
No Sabe : 0
Total : 20



Grafica 11 – Percepción de éxito en asegurar la calidad del software

En la grafica que representa las respuestas a la pregunta numero 13 se aprecia que solo 1 de cada 20 empresas asegura la calidad en 20% de sus proyectos, por otro lado se puede observar que el 15% lo realiza en el 40% de sus proyectos, también 1 de cada 10(10%) tiene una efectividad del 60%, la mayoría que corresponde al 35 % de las empresas encuestadas tiene una efectividad del 80%. Solo un 30% de las empresas cuentan con un efectividad completa al asegurarla en todos sus proyectos. Por ultimo solo un 1 empresa que corresponde al 5% de la población comenta que el asegurar la calidad no aplica a sus proyectos.

Es imperativo que las empresas comiencen a buscar nuevos métodos para mejorar el desempeño de los procesos de aseguramiento de calidad del software. En este sentido se pensar que existen magnificas oportunidades para las empresas de instrumentar la metodología de desarrollo propuesta por Racional y que ayuda a mantener y asegurar la calidad del software durante todo el ciclo de desarrollo de software.

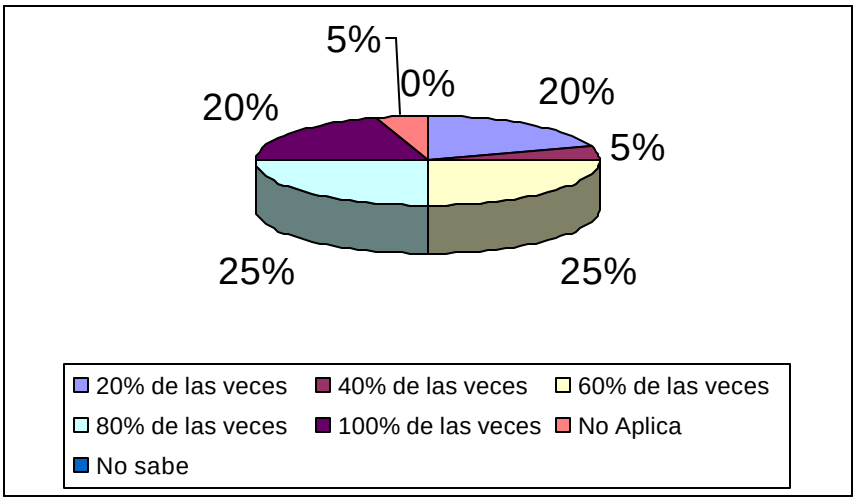
Esta metodología como ya se conocía proporciona beneficios como: minimización de errores, eliminación de re-trabajo, Incremento en la funcionalidad del software, aprendizaje de equipo, entre otros.

14.-Los riesgos de nuestros proyectos son evaluados y actualizados continuamente (incluyendo los planes de contingencia asociados) durante todo el ciclo de vida de cada proyecto. (Riesgos)

14	***R Minimizar Riesgos***	*3*	*3*	*4*	*4*	*1*	*3*	*5*	*5*	*5*	*4*	*1*	*2*	*6*	*3*	*1*	*4*	*4*	*3*	*1*	*5*

Tabla 21 - Detalle de la pregunta 14

En un 20% de las veces: 4
En un 40% de las veces: 1
En un 60% de las veces: 5
En un 80% de las veces: 5
En un 100% de las veces: 4
No Aplica : 1
No Sabe : 0
Total : 20



Grafica 12 – Percepción de éxito al minimizar riesgos

En la grafica que representa las respuestas a la pregunta numero 14 se puede apreciar que 2 de cada 10 empresas manejan una administración de riesgos con una afectividad del 20%, mientras solo el 5% de las empresas lo hacen con el 40%, por otro lado una cuarta parte realiza esta procedimiento con un 60% y 80% respectivamente, y por ultimo una de cada 5 realiza su estimación de riesgos con una efectividad del 100%. Es importante recalcar que una empresa que representa el 5% de la población comento que no aplica a su empresa esta métrica.
Por lo que se detecta que también en este proceso las empresas representan bastantes oportunidades de mejora en el desempeño y resultados de su proceso. En ese sentido las empresas deberán buscar nuevas metodologías o herramientas para incrementar el grado de éxito al maximizar riesgos.

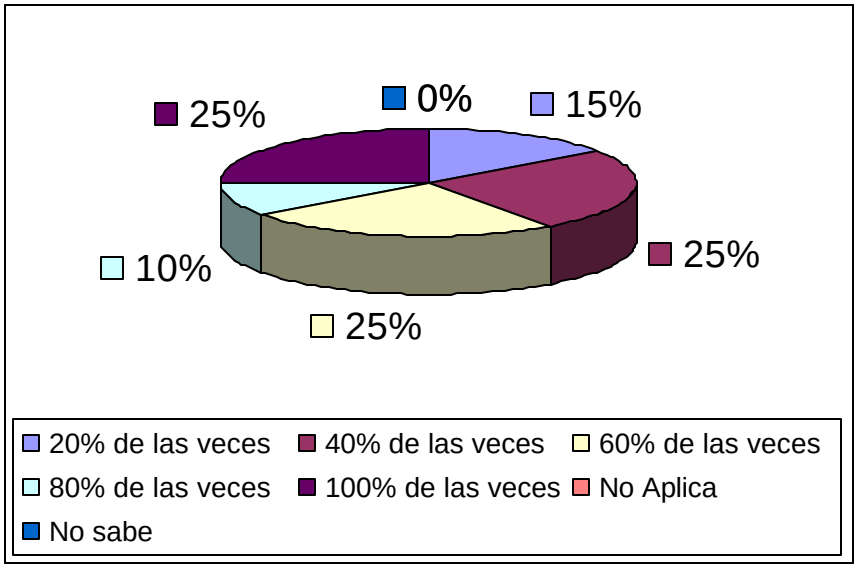
El contar con un proceso eficiente de minimizar riesgos traerá beneficios como: identificación y resolución temprana de riesgos, eliminación de re-trabajo, Aumento en la calidad del software entre otros.

15.- Cuando cambian los requerimientos de software hacemos los ajustes pertinentes a los programas y documentos relacionados (modelos de diseño, manuales de usuario), así como a los planes de trabajo (calendarios de actividades y compromisos) Así como Podemos saber con facilidad quien ha utilizado un determinado elemento del software, que cambios le hizo, cuando lo modificó y por qué.(cambios)

15	***R Administrar Cambios***	*1*	*3*	*4*	*5*	*2*	*2*	*5*	*3*	*5*	*5*	*1*	*3*	*3*	*5*	*1*	*2*	*2*	*4*	*2*	*3*

Tabla 22 - Detalle de la pregunta 15

En un 20% de las veces: 3
En un 40% de las veces: 5
En un 60% de las veces: 5
En un 80% de las veces: 2
En un 100% de las veces: 5
No Aplica : 0
No Sabe : 0
Total : 20



Grafica 13 – Percepción de éxito al administrar requerimientos

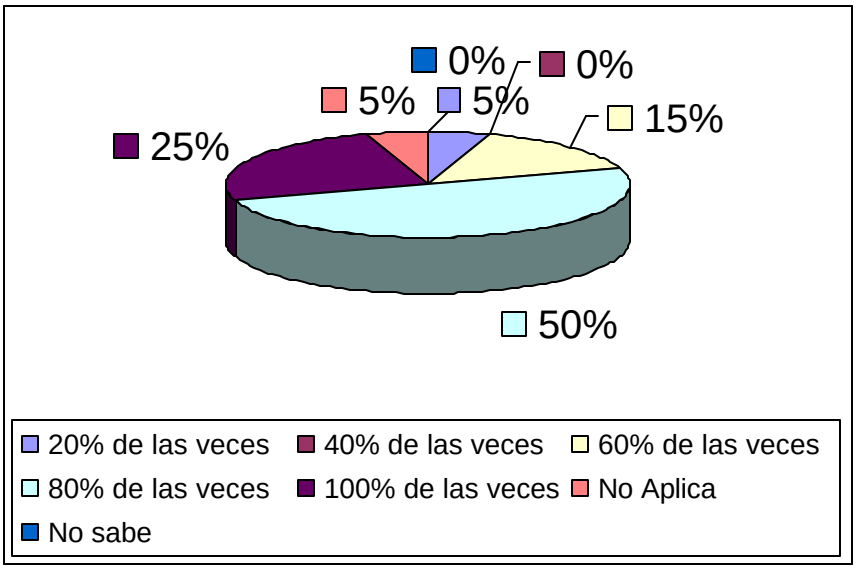
En la grafica que representa las respuestas a la pregunta numero 15 se puede apreciar que un 15 % de las empresa encuestadas realizan su administración de cambios con una efectividad del 20%, mientras que las que lo realizan con una efectividad del 40% y 60% fueron una cuarta parte (25%) cada una, las que obtuvieron el 80% de éxito son el 10% de las empresas, mientras que otra cuarta(25%) parte tienen un 100% de efectividad a la hora de administrar los cambios.
En este sentido se identifica también que la mayoría de empresas presentan oportunidades de mejora al tener la percepción de no tener éxito en la administración de requerimientos. Por ello es importante que implementen un plan de mejoras que les ayude a incrementar el éxito en este proceso.

16.- Existe buena comunicación entre nuestro equipo del proyecto (comunicación)

16	***R Asegurar Comunicación***	*4*	*3*	*4*	*4*	*1*	*4*	*5*	*5*	*4*	*4*	*4*	*4*	*6*	*4*	*3*	*4*	*5*	*5*	*3*	*5*

Tabla 23 - Detalle de la pregunta 16

En un 20% de las veces: 1
En un 40% de las veces: 0
En un 60% de las veces: 3
En un 80% de las veces: 10
En un 100% de las veces: 5
No Aplica : 1
No Sabe :0
Total : 20



Grafica 14 – Percepción de éxito al mejorar la comunicación entre involucrados

En cuestión de asegurar la comunicación en con los involucrados en los proyectos el 5% de las empresas comento que tienen éxito solo en un 20%, mientras que un 15 los hace con el 60% de los proyectos la mitad de las empresas comento que tienen un porcentaje de éxito en un 80%, mientas que solo un 25% de las empresas aseguran la comunicación en un 100%, cabe mencionar que hubo una empresa que comento que asegurar la comunicación no aplica para sus proyectos.

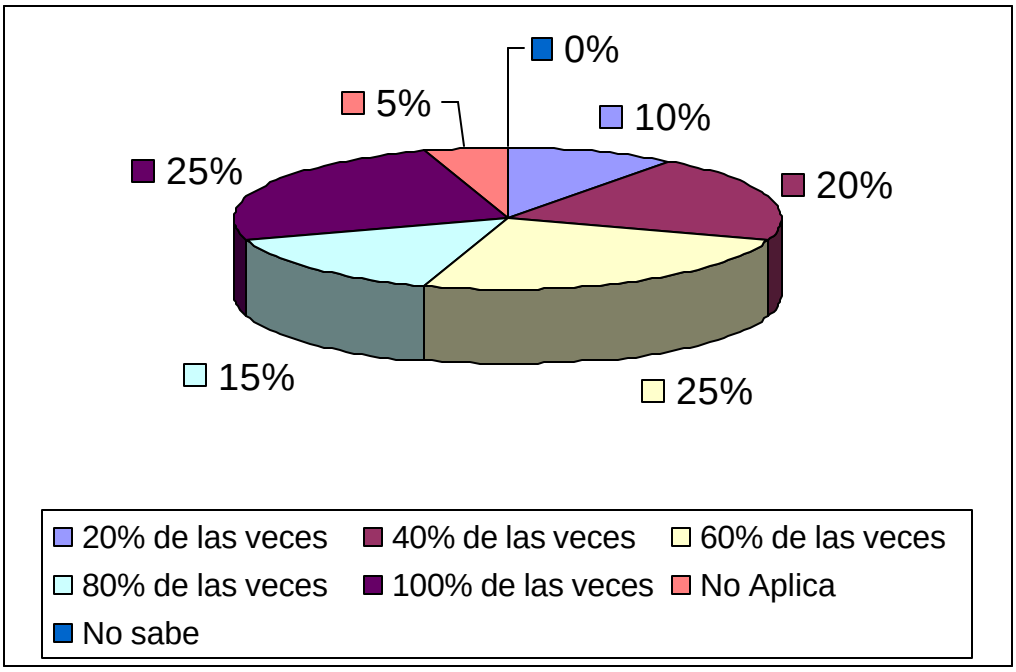
Este proceso se detecta que las empresas cuentan con una buena percepción de éxito en este sentido, que las empresas no han logrado un grado de éxito optimo para tener la certeza de que logran una comunicación eficiente en el 100% de los casos, el gran reto es que estas empresa implementen un plan para lograr que la comunicación entre involucrados se de y permita asegurar una producción de software eficaz.

17.- Identificamos las amenazas y problemas más importantes del proyecto antes de comenzar su diseño y desarrollo (requerimientos)

17	***R Administrar Requerimientos***	*1*	*3*	*1*	*5*	*2*	*2*	*5*	*5*	*6*	*4*	*5*	*3*	*3*	*3*	*2*	*4*	*4*	*3*	*2*	*5*

Tabla 24 - Detalle de la pregunta 17

En un 20% de las veces: 2
En un 40% de las veces: 4
En un 60% de las veces: 5
En un 80% de las veces: 3
En un 100% de las veces: 5
No Aplica : 1
No Sabe : 0
Total : 20



Grafica 15 - Resultados Pregunta 17

Por ultimo en la métrica de administrar requerimientos solo el 10% de las empresas contestaron que son efectivas solo con un 20% mientras que un 20% lo es con el 40%, por otra parte una cuanta parte (25%) comento que tiene una efectividad del 60%, y un 15% del 80%, si embargo otra cuanta parte tienen un porcentaje de éxito del 100%. Mientras que el 5% comento que los requerimientos no aplican, a sus proyectos.

Se detecta que una parte significativa (25%) de las empresas tiene asegurado la administración de requerimientos, mientras el 75% tiene deficiencias, por lo se visualiza un área de oportunidad para que estas empresas logren mejorar el desempeño de la administración de requerimientos y logren obtener beneficios como: identificación y seguimiento de requisitos desde las etapas tempranas de desarrollo.

Debido a que ninguna empresa entrevistada cuenta con una metodología de desarrollo iterativo, en la presente investigación no será posible contrastar los resultados que ofrece una metodología de enfoque iterativo contra los beneficios que ofrece las metodologías de desarrollo tradicionales.

Para complementar este estudio trataremos de encontrar si existe una correlación entre el grado de automatización (instrumentación) de los procesos o métricas con que cuentan las empresas contra los resultados que obtienen al aplicar dichas herramientas, en caso de que esta aseveración resulte afirmativa, se podrá inferir que existen posibilidades que con ayuda de metodologías que propongan el uso de herramientas automatizadas las empresas incrementen el desempeño del desarrollo de software.

Lo que significa que las empresas que tienen metodologías de desarrollo de software clásicas cuentan con oportunidades de mejorar el desempeño del desarrollo de software al implantar una metodología de desarrollo iterativo que se base en una instrumentación de procesos.

Comparación entre nivel de automatización y nivel de éxito en resultado

Se realizo un análisis parametrico con el objetivo de verificar si existe una correlación entre las variables de automatización de procesos y el resultado que obtienen las empresas en sus procesos o tareas de desarrollo de software.

Para ello se realizo la prueba de coeficiente de relación de Pearson el cual consiste en una prueba estadística para analizar la relación entre dos variables medidas en un nivel por intervalos o de razón. (Hernández, Fernández y Baptista (2003))

La prueba de coeficiente de relación de Pearson se simboliza con r.

Normalmente se utiliza para probar una hipótesis del tipo "A mayor X, Mayor Y", "Altos valores en X están asociados con altos valores en Y", "Altos Valores en X se asocian con bajos valores de Y"

Interpretación: el coeficiente "r" de Pearson puede variar de -1.00 a +1.00, donde

r = Interprestacion "r"
-1.00 = Correlación negativa perfecta
-0.90 = Correlación negativa muy fuerte
-0.75 = Correlación negativa considerable
-0.50 = Correlación negativa media
-0.10 = Correlación negativa débil
0.00 = No existe correlación alguna entre las variables
+0.10 = Correlación positiva débil
+0.50 = Correlación positiva media
+0.75 = Correlación positiva considerable
+0.90 = Correlación positiva muy fuerte
+1.00 = Correlación positiva perfecta

Tabla 25- Interpretación de del Nivel de medición de Variables

Se formaran pares de preguntas correspondientes un mismo proceso, por un lado una pregunta sobre el grado de automatización(instrumentación) del proceso (Pregunta 3 a la 8), y por otro lado, el grado de éxito que tiene la empresa en ese mismo proceso (Preguntas 11 a la 17). En búsqueda de una correlación existente que nos ayude a determinar una correlación del tipo "A mayor es el valor de B"

Los pares de preguntas serán los siguientes:

PROCESO	Pregunta relacionada con Automatización del proceso	Pregunta relacionada con el grado de éxito del proceso
Estimación de Costos	**2**	**11**
Estimación de Tiempos	**3**	**12**
Calidad en Software	**4**	**13**
Minimizar riesgos	**5**	**14**
Administrar Cambios	**6**	**15**
Comunicación de resultados	**7**	**16**
Administración de requerimientos	**8**	**17**

Adicionalmente se realizo el cálculo de la media para cada categoría para determinar eol promedio de automatización que cuentan las empresas y el promedio de grado de exito

La formula de la media es la siguiente:

$$\text{Media} = \bar{X} = \frac{\sum X}{N}$$

$$\sum (X - \bar{X})$$

El calculo de la media permite identificar cual es el promedio de las respuesta que dieron las empresas.

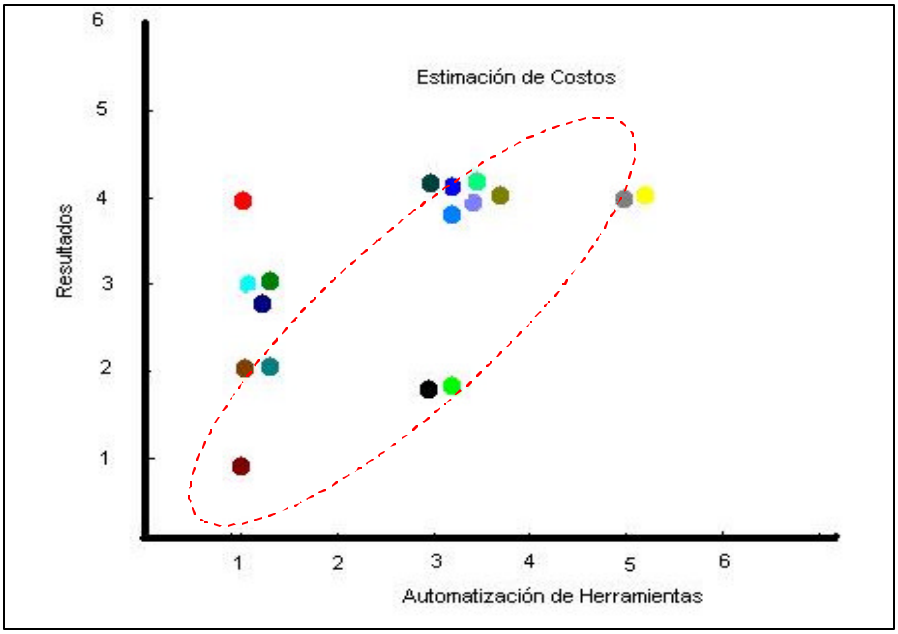
Como complemento se realizo un diagrama de Dispersión el cual tiene como objetivo visualizar gráficamente una correlación existente. (Hernández, Fernández y Baptista (2003))

Todas estas operaciones se realizaron con ayuda de una hoja de Excel.

Los resultados se describen a continuación :

Administración de costos:
En búsqueda de una correlación existente del grado de automatización del proceso de estimación de costos, contra el éxito que obtiene la empresa en el mismo proceso.

2	*Administrar y estimar costos*	*3*	*5*	*1*	*5*	*3*	*1*	*3*	*1*	*3*	*3*	*1*	*1*	*1*	*3*	*1*	*3*	*3*	*1*	*1*	*3*
11	*R Estimaron de Costos*	*2*	*4*	*4*	*4*	*2*	*3*	*4*	*3*	*4*	*4*	*1*	*2*	*1*	*4*	*1*	*0*	*4*	*0*	*2*	*4*



Grafica 16 – Diagrama de Dispersión del proceso de costos

Al realizar el diagrama de dispersión del proceso de costos no es posible identificar una correlación existente a simple vista, por lo que será necesario el uso de otras herramientas para valorar dicha correlación.

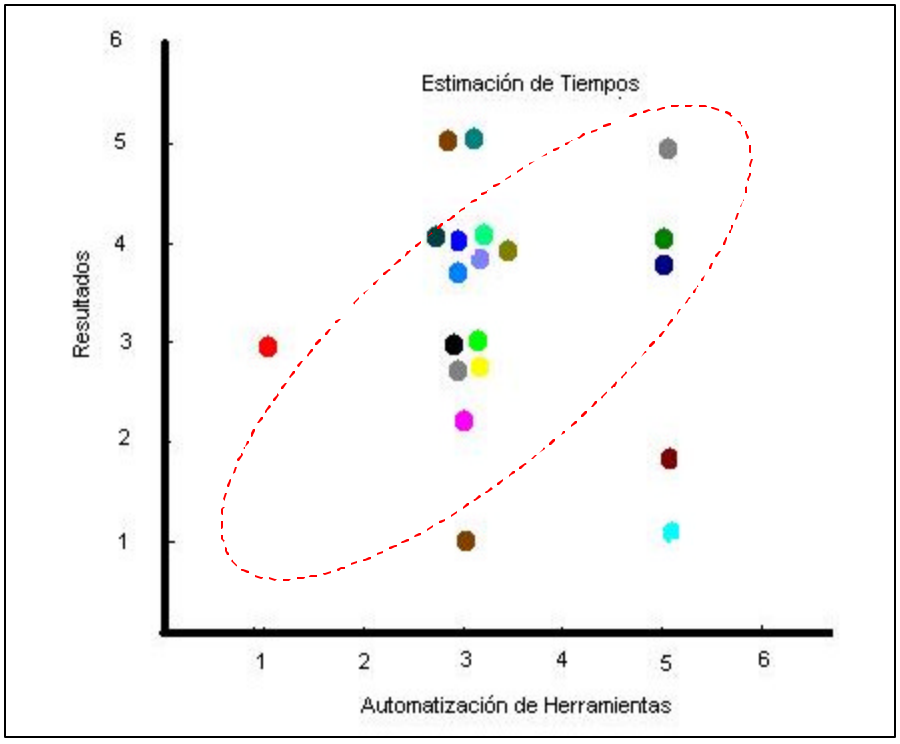
Al realizar la relación de Pearson se obtuvo un valor de 0.512, que al identificarlo en la tabla 25, arroja que existe una **correlación positiva media entre las variables estudiadas**. También es posible observar la empresas cuentan con una media en Automatización de 2.3, lo que significa que en promedio las empresas estiman costos en base a su experiencia en proyectos pasados, lo que da lugar a muchos errores. Y en el renglón 11 en promedio las empresas tienen un éxito del 23%, lo que arroja que en promedio las empresas no cuentan con una procesos bien definido que les asegure una estimación de costos acertada.

Se puede concluir que "Si existe una correlación entre una instrumentación de los procesos de estimación de costos contra el grado de éxito en ese proceso que en promedio las empresas estiman costos con base a su experiencia en proyectos anteriores, pero su porcentaje de éxito tiene a ser del solo el 23% en ese proceso.

Administración y estimación de tiempo

En búsqueda de una correlación existente del grado de automatización del proceso de administración y estimación de Tiempo, contra el éxito que obtiene la empresa en el mismo proceso.

3	***Administrar y estimar tiempos***	*5*	*3*	*1*	*5*	*3*	*3*	*5*	*5*	*3*	*3*	*3*	*3*	*3*	*3*	*3*	*5*	*3*	*3*	*3*	*3*
12	***R Estimaron de Tiempos***	*1*	*3*	*3*	*4*	*1*	*4*	*4*	*5*	*5*	*3*	*5*	*4*	*3*	*3*	*4*	*2*	*4*	*4*	*2*	*4*



Grafica 17 – Diagrama de Dispersión del proceso de Tiempos.

Al realizar el diagrama de dispersión del proceso de tiempos también se puede observar que realmente no existe una correlación entre las dos variables.

Al realizar la relación de Pearson se obtuvo un valor de -0.050, que al identificar en la tabla 25 es posible identificar una **correlación negativa débil, tendiendo a No tener una correlación**. También es posible determinar que la media de automatización es de 3.4 lo cual significa que las empresas en promedio estiman tiempos en base a sus experiencias en proyectos pasados con tendencia a utilización de herramientas automatizadas. Mientras el promedio de éxito que tienen en el proceso de estimación de costos es de 68%.

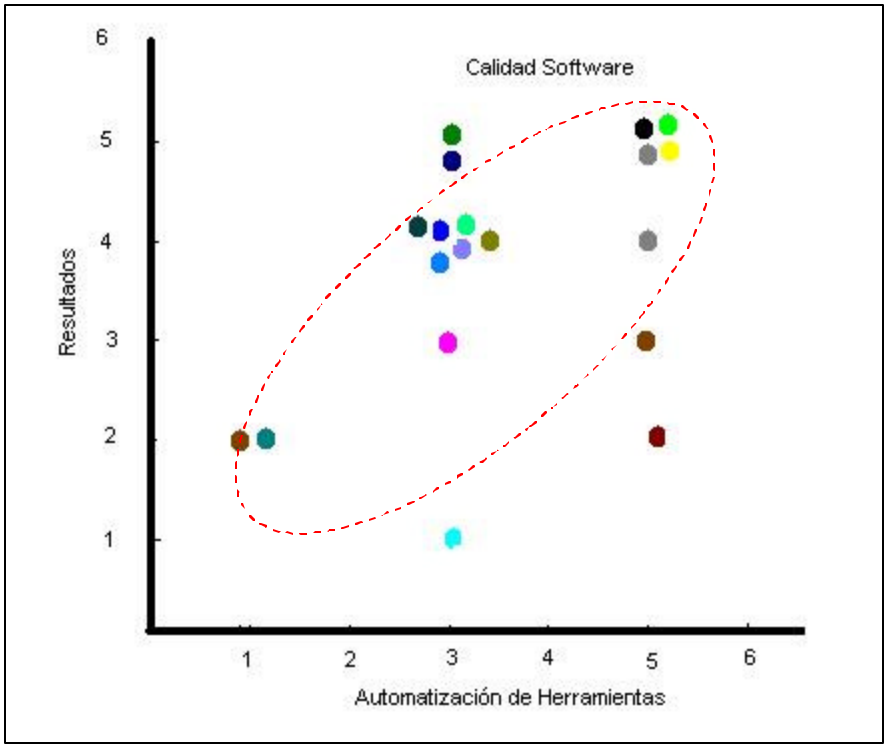
En este procesos se puede concluir que "No existe una correlación entre una instrumentación de los procesos de estimación de costos contra el grado de éxito en ese

proceso, en promedio las empresas estiman tiempos en base a su experiencia en proyectos anteriores, y su porcentaje de éxito es de 68% en ese proceso.

Calidad en el Software

En búsqueda de una correlación existente del grado de automatización del proceso de mejoramiento de calidad del software, contra el éxito que obtiene la empresa en el mismo proceso.

4	*Calidad de software*	*3*	*3*	*3*	*5*	*3*	*5*	*5*	*5*	*3*	*5*	*3*	*3*	*3*	*3*	*1*	*5*	*3*	*3*	*1*	*5*
13	*R Asegurar Calidad*	*4*	*3*	*4*	*3*	*1*	*2*	*5*	*5*	*5*	*5*	*5*	*4*	*0*	*4*	*2*	*4*	*4*	*4*	*2*	*5*



Grafica 18 – Diagrama de Dispersión del proceso de calidad del software

Al realizar el diagrama de dispersión del proceso de calidad de software, se puede observar que podría tener algún tipo de correlación, sin ser esta muy marcada, pues existen puntos dispersos que no permiten que la correlación se aprecie muy bien.

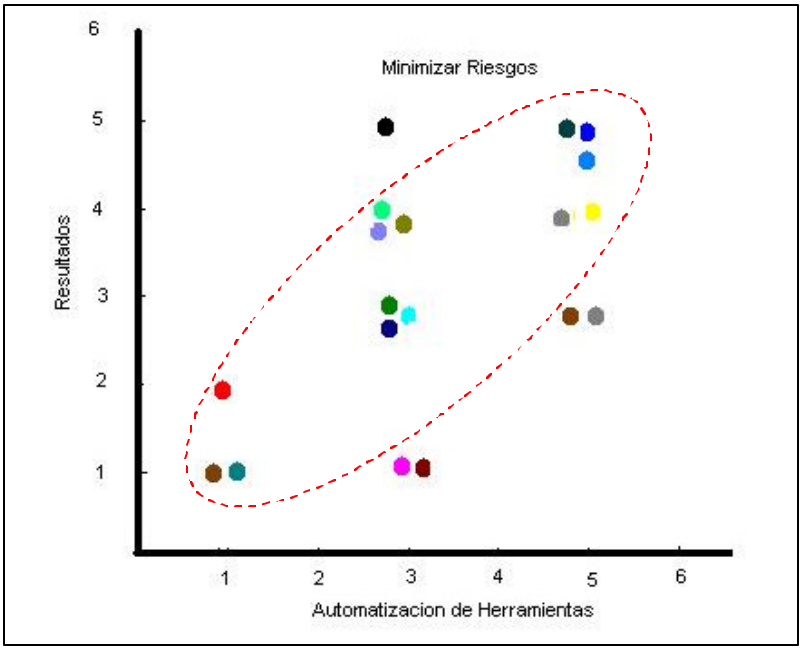
En la relación de Pearson se observa un valor de 0.406, lo cual significa una "**Correlación positiva débil con tendencia a una correlación positiva media**. En promedio (media de 3.4) las empresas cuentan con un plan de aseguramiento de calidad de manera informal, mientras que el grado de éxito en este proceso es 60% (media de 3.4).

En este procesos se podrá concluir que "Si existe una correlación entre una instrumentación de los procesos de aseguramiento de calidad contra el grado de éxito en ese proceso, en promedio las empresas estiman cuentan con un plan de aseguramiento de calidad con un grade de éxito del 68%. Podría ser una gran oportunidad que las empresas instrumenten el procesos de tal manera que este 100% automatizado con el objetivo de mejorar el grado de éxito al asegurar la calidad.

Minimizar Riesgos

En búsqueda de una correlación existente del grado de automatización del proceso de minimizar riesgos, contra el éxito que obtiene la empresa en el mismo proceso.

5	***Minimizar riesgos***	*3*	*5*	*3*	*3*	*1*	*5*	*5*	*5*	*3*	*5*	*3*	*1*	*3*	*3*	*3*	*5*	*3*	*3*	*1*	*5*
14	***R Minimizar Riesgos***	*3*	*3*	*4*	*4*	*1*	*3*	*5*	*5*	*5*	*4*	*1*	*2*	*0*	*3*	*1*	*4*	*4*	*3*	*1*	*5*



Grafica 19 – Diagrama de Dispersión del proceso de minimizar riesgos

Al realizar el diagrama de dispersión del proceso de control de riesgos también se puede observar a simple vista que existe una relación un poco mas marcada que en las anteriores graficas.

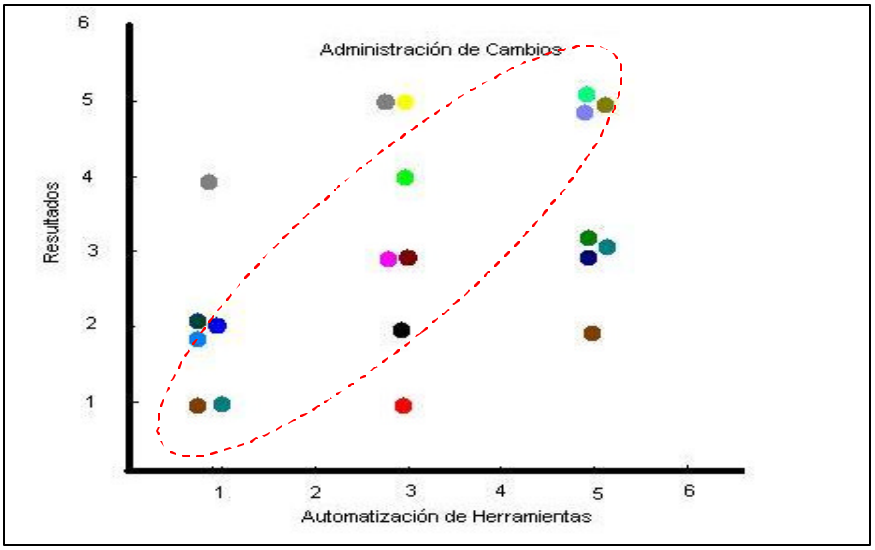
Con respecto a la relación de Pearson la cual arroja un valor 0.61, el cual significa una "**Correlación positiva media con tendencia hacia una correlación positiva considerable**. También se observa que en promedio (media de 3.4) las empresas cuentan con un plan para minimizar riesgos, pero sin ser un plan automatizado, mientras promedio(media de 3.05) del grado de éxito en el proceso de minimización de riesgos es de 60%.

En este procesos se podrá concluir que "**Si existe una correlación entre una instrumentación de los procesos de aseguramiento de calidad contra el grado de éxito en ese proceso**, en promedio las empresas cuentan con un plan para minimizar riesgos de manera informal, un grado de éxito en minimizar riesgos de 60%, por lo será de gran oportunidad para las empresas el implementar instrumentos para minimizar riesgos que les ayude a las empresas a obtener un control estricto de los riesgos y puedan ser atacados en etapas tempranas.

Administrar Cambios

En búsqueda de una correlación existente del grado de automatización del proceso de Administración de cambios, contra el éxito que obtiene la empresa en el mismo proceso.

6	*Administrar cambios*	*1*	*5*	*3*	*3*	*1*	*1*	*5*	*5*	*5*	*5*	*3*	*3*	*3*	*3*	*1*	*3*	*5*	*1*	*1*	*5*
15	*R Administrar Cambios*	*1*	*3*	*4*	*5*	*2*	*2*	*5*	*3*	*5*	*5*	*1*	*3*	*3*	*5*	*1*	*2*	*2*	*4*	*2*	*3*



Grafica 20 – Diagrama de Dispersión del proceso de administración de cambios

Al realizar el diagrama de dispersión del proceso de administración de cambios también se puede observar a simple vista que no existe una relación marcada entre las variables sujetas a estudio.

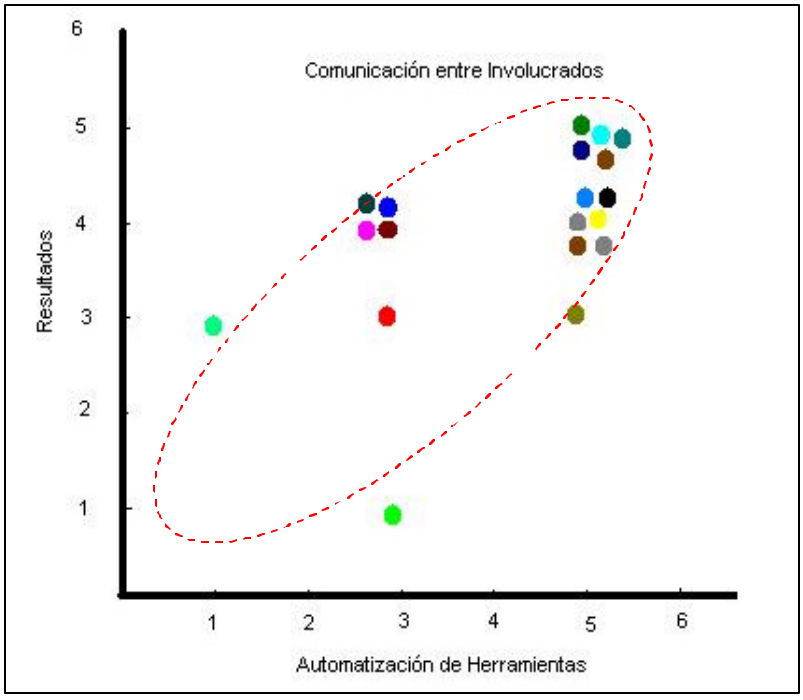
Al realizar la relación de Pearson arroja valor de 0.48, lo cual significa que existe una "**correlación positiva media"**. También se puede observar que en promedio (media de 3.1) las empresas cuentan con un plan informal para el recesos de administración de cambios, por otra parte el promedio de éxito de en este proceso es de (3.1) 60% en lo que respecta a la administración de cambios.

Por lo que se puede concretar que en el proceso de administración de cambios si existe una correlación entre el grado de automatización del procesos de administración de cambios y el grado de éxito que las empresas tienen en este proceso. Por lo que existe grandes oportunidades que al instrumentar el proceso de una manera automática, las empresas puedas mejorar el desempeño al momento de administrar cambios, lo cual traería beneficios como: minimizar riesgos, realizar todos los cambios necesarios, darle seguimiento a cambios, eliminación de re-trabajo y por supuesto maximizar los recursos humanos del equipo de desarrollo.

Comunicación entre involucrados

En búsqueda de una correlación existente del grado de automatización del proceso de Comunicación entre involucrados, contra el éxito que obtiene la empresa en el mismo proceso.

7	***Comunicación entre involucrados***	*5*	*3*	*3*	*5*	*3*	*5*	*5*	*5*	*5*	*5*	*3*	*3*	*3*	*3*	*5*	*5*	*5*	*5*	*1*	*5*
16	***R Asegurar Comunicación***	*4*	*3*	*4*	*4*	*1*	*4*	*5*	*5*	*4*	*4*	*4*	*4*	*0*	*4*	*3*	*4*	*5*	*5*	*3*	*5*



Grafica 21 – Diagrama de Dispersión del proceso de comunicación.

Al realizar el diagrama de dispersión del proceso de asegurar la comunicación, no es fácil de identificar si existe una correlación entre las variables involucradas.

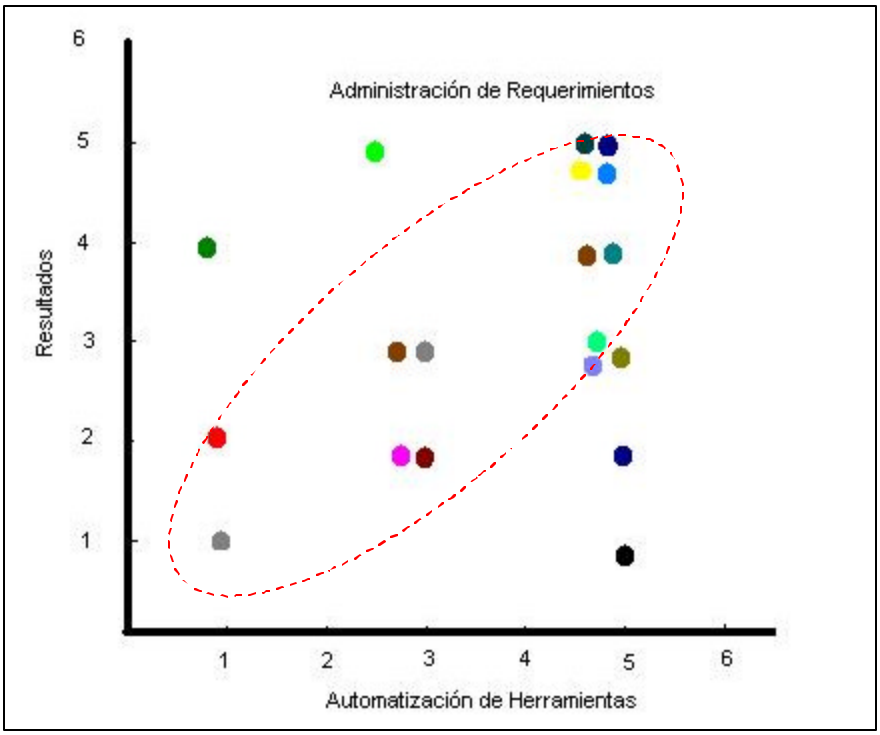
Al realizar la relación de Pearson arroja valor de 0.48 que significa una **correlación positiva media**. También se puede observar que en promedio (media de 4.1) las empresas cuentan tienden a contar con herramientas automatizadas para mejorar la comunicación entre los involucrados de un proyecto. También se observa que en promedio (3.75) 72% en lo que respecta al éxito en asegurar la comunicación entre los involucrados.

Por lo que se puede concretar que en el proceso de aseguramiento de comunicación entre involucrados **si existe una correlación** entre contar con herramientas automatizadas para asegurar la comunicación entre involucrados en un proyecto y el desempeño de la comunicación este los mismos involucrados, por lo que se detecta que la mayoría de las empresas tienen posibilidades de que con ayuda de un procesos de aseguramiento de la comunicación bien instrumentado se pueda mejorar el desempeño en la comunicación.

Administrar Requerimientos

En búsqueda de una correlación existente del grado de automatización del proceso de Administración de requerimientos, contra el éxito que obtiene la empresa en el mismo proceso.

8	***Administrar requerimientos***	*5*	*5*	*1*	*5*	*1*	*3*	*5*	*5*	*5*	*5*	*3*	*5*	*3*	*3*	*5*	*1*	*5*	*5*	*3*	*5*
17	***R Administrar Requerimientos***	*1*	*3*	*1*	*5*	*2*	*2*	*5*	*5*	*0*	*4*	*5*	*3*	*3*	*3*	*2*	*4*	*4*	*3*	*2*	*5*



Grafica 22 – Diagrama de Dispersión del proceso de administrar requerimientos

Después de realizar el diagrama de dispersión del proceso para administrar requerimientos, se puede observar a simple vista que no existe una relación marcada entre las variables sujetas a estudio.

La relación de Pearson arroja valor de 0.23, lo cual significa una **correlación positiva débil**. también al realizar el calculo de la media, se detecto que en promedio (3.9) las empresas tienden a contar con herramientas automatizadas que les ayudan a administrar los requerimientos, y el grado de éxito en promedio es de 80% , pero a pesar que ambas variables muestran valores altos, la correlación es débil.

Se puede concluir que aunque en este proceso se muestra una pobre correlación, las empresas puede implantar metodologías instrumentadas para administrar los requerimientos, pues se recuerda que la administración de requerimientos es uno de los procesos mas críticos, debido a que es uno de los primeros y de ahí se derivan el éxito en la planeación y posterior ejecución del proyecto.

Observaciones

A pesar que no existe una marcada correlación entre los procesos, se pudo observar que existen algunos procesos en los que se podría considerar que existe una correlación media o consistente, lo que refuerza la idea de que a mayor grado de automatización de los procesos, mejor desempeño tendrán, por lo que el grado de éxito en los proyectos se incrementara. Es así que las empresas pueden observar un área oportunidad en la implantación de una metodología de desarrollo iterativo que ayude instrumentar el proceso de desarrollo de software, con el objetivo de crear un proceso maduro que permita el éxito en el desarrollo de proyectos de desarrollo de software.

Capitulo VI. Conclusiones

Durante años las compañías que desarrollan software han tenido los mismos problemas, los equipos de proyectos de desarrollo de software no pueden completar los proyectos en el tiempo, el costo y la calidad especificada desde la etapa de diseño. Así lo demuestran los porcentajes de Caos (1995), donde solo el 9% de los proyectos se consideran exitosos.

A través de los años diversas metodologías han sido desarrolladas e implementadas con el objetivo de que hacer mas eficiente y rápido el procesos de desarrollo de software. Entre dichas metodologías se encuentran: el modelo de cascada propuesto por Walter Royce, el modelo de prototipos, el modelo de espiral y el modelo incremental.

A pesar de los esfuerzos de los especialistas en desarrollo de software, los mismos problemas de antes, son lo mismos problemas de la actualidad. Una mala estimación de tiempos y costos, además de una producción de software de mala calidad la cual cumple con las expectativas de los usuarios.

En los últimos años se esta popularizando una nueva metodología de desarrollo de software que utiliza un enfoque iterativo, la cual busca revolucionar la manera como se desarrolla software en nuestros días. Esta es la metodología de proceso unificado propuesta por racional (RUP).

El RUP enfatiza el desarrollo y el mantenimiento de modelos, representaciones semánticamente enriquecidas del sistema de software bajo desarrollo.

Su meta es la de asegurase una producción de alta calidad en el software y conocer las necesidades del usuario final, con un calendario y presupuesto predecible.

El proceso comprende las mejores prácticas en el área de desarrollo de proyectos de software las cuales son:

1.- Desarrollar software de manera iterativa
2.- Administración de requerimientos
3.- Uso de arquitectura basada en componentes
4.- Uso de software de modelación visual
5.- Verificar la calidad de software
6.- Control de cambios del software

Los cuales tienen beneficios como:

- Calendarios predecibles.
- Costos predecibles.
- Minimizar riesgos.
- Mejorar comunicación entre involucrados.
- Eliminar re-trabajo y desperdicio.
- Comunicación efectiva y eficiente.
- Aprendizaje de equipo.
- Incremento en funcionalidad, confiabilidad y desempeño del software.

Así como la metodología contiene beneficios tangibles para el desarrollo de proyectos de software de manera eficiente. También conlleva algunos inconvenientes los cuales se presentan a continuación.

Costo de Implantación Alto: Al ser necesario desarrollar un ambiente adecuado para el desarrollo de software, es necesaria la compra de herramientas que permitan automatizar el desarrollo. Dichas herramientas tiene un costo adicional y algo elevado.

Administración Compleja: Una vez que el ambiente ha sido desarrollado sera necesario administrarlo, por lo que se debe contar con un equipo especializado en cada una de las áreas que integran la metodología, por lo que se considera que la metodología tiene una administración compleja.

Actualmente en México y en especial en Monterrey, existe una oportunidad de desarrollar software de manera eficiente, este dio como resultado al realizar una investigación de campo en el presente trabajo de investigación, se identifico el estado actual de las metodologías de desarrollo que utilizan las empresas. Y se realizo una confrontación entre el desempeño de las metodologías de desarrollo que utilizan las empresas, contra los beneficios propuestos por la metodología de enfoque unificado de racional, para identificar la posibles áreas de oportunidad existentes en las empresas.

Además como la metodología de RUP se baza muy ampliamente en la instrumentación de los procesos de desarrollo de software, se trato de encontrar la existencia de una correlación entre el grado de éxito que mantienen las empresa, contra el grado de automatización de los procesos de desarrollo de software.
La investigación arrojo los siguientes resultados:

Las empresas de Monterrey que tienen como función desarrollar software no cuentan con una metodología que se pueda considerar lo suficientemente robusta para asegurar el éxito de sus proyectos de software, esto lo vemos reflejado en los resultados que se obtuvieron.

Metodología
Se pudo observar que el 80% de las empresas cuentan con metodologías tradicionales de desarrollo de software, las cuales cuentan con deficiencias, provocan que se sigan elaborando proyectos de desarrollo de software de manera artesanal, basándose completamente en la experiencia de sus empleados.

Estimación de Costos
Con respecto a la estimación e costos solo se detecto que un 45% por ciento de las empresas no realizan estimaciones de costos, este es un grave error, debido a que no existe un control sobre los recursos financieros de proyecto, y puede provocar que no sea un proyecto exitoso.

Al contar con una estimación de costos automatizada es posible incrementar el éxito de la estimación esto es producto de una relación existente entre el grado de automatización de este proceso y la percepción de éxito que tiene las empresas.

Se detecto también que las empresas basan su estimación e costos en base a su experiencia en proyectos pasados, lo cual es un error muy grave pues no existirá un proyecto igual al otro. Por otra parte al basarse en la experiencia de sus empleados, siempre contaran con el riesgo que esa persona salga de la empresa y se convierta en un error muy serio.
Por que es indispensable herramientas que permitan que personal sin experiencia logre una eficiente estimación d e costos.

Estimación de tiempos
También se observo que el 70% estima sus tiempos en base a experiencias pasadas, lo que puede dar lugar a estimaciones erróneas, puesto que no existen dos proyectos iguales debido a las distintas condiciones que cada proyecto tiene. Ejemplo: distintos requerimientos, distintos riesgos, etc.

Se puede considerar un error basarse en estimaciones pasadas para evaluar los tiempos de desarrollo. La gran oportunidad es diseñar e implantar un proceso con ayuda de herramientas automatizadas que provea un marco para una estimación de costos pragmática y se logre administrar esta parte tan difícil de los proyectos.

Aseguramiento de calidad
Tomando en consideración que un proyecto exitoso es aquel que cumple con el tiempo-el costo y la calidad estimada inicialmente se encontró que el 90% de las empresas realmente no cuentan con un proceso que se pueda considerar lo suficientemente maduro para asegurar el desarrollo de un producto de calidad. Sin embargo el restante 10% tienen un proceso, que se puede considerar lo suficientemente maduro al obtener muy buenas calificaciones en los resultados que obtienen.

Por lo tanto es necesario implementar un proceso madure que permita asegurar la calidad de software.

Minimizar riesgos
El seguimiento, identificación y minimización de riesgos es unos de los procesos mas críticos, de este procesos depende que el proyecto se encuentre bajo control y en calendario, es por eso la importancia de que las empresas tengan un proceso de minimización de riesgos muy eficiente.

En ese sentido se observó que solo el 35% de las empresas cuentan con plan establecido con ayuda de herramientas que les permite mantener identificados y bajo control cada uno de los posibles riesgos. En el 75% Restante es prioritario instrumentar el proceso para permitir un control pleno de los riesgos del proyecto, la cual representa beneficios como:

- Control cercano con los recursos
- Calendario siempre predecible
- Costo siempre predecible
- Desempeño favorable por parte del equipo de desarrollo.

Administración de cambios
También se detecto que la administración de cambios es una parte muy importante en el proceso de desarrollo de software. Por lo que es necesario contar con un proceso automatizado que permita la administración, seguimiento y control de cada cambio que se planea o se le realiza al software.
En este sentido solo el 35% entiende la importancia de este proceso al contar con un proceso maduro e instrumentado para la administración de cambios. Para el 65% restante de las empresas es prioritario desarrollar y automatizar este proceso.

Mejorar la Comunicación con los involucrados
Otro punto fundamental en el desarrollo de los proyectos es que exista una comunicación eficiente entre los involucrados. Pero la realidad de las empresas nos muestra otra cosa, solo el 55% de las empresas poseen un plan establecido con ayuda de herramientas automatizadas para lograr una buena comunicación entre los involucrados en el proyecto.
El 45% restante no han detectado esta situación, por lo que es imperativo las empresas implanten un proceso que permita una comunicación eficiente entre los involucrados, lo que les permitirá tener beneficios como:

- Estar enterados de avances, riesgos, cambios
- Eliminación del trabajo doble
- Fluidez de la comunicación
- Sentido de pertenecía al equipo
- Seguimiento de imprevistos

Administración de requerimientos
La administración de requerimientos es una parte crucial en el desarrollo, de ella depende en gran parte el diseño del ciclo de desarrollo, por lo que es también es considerado vital que se tenga un proceso maduro para asegurar una buena administración de

requerimientos. En este sentido solo el 60% de las empresas cuentan con herramientas automatizadas para administrar y dar seguimiento a los requerimientos, El restante 40% carecen de una unos procesos maduros. Estas empresas cuentan con un excelente área de oportunidad para implantar un proceso les ayude a administrar requerimientos

Para cada uno de los procesos anteriores el RUP proporciona un marco de referencia que permite implantar las mejores practicas del desarrollo del software, esto es con el uso de herramientas , artefactos, puntos de chequeo, implantación de estándares de calidad, revisión continua de los resultados, adecuando el proceso de desarrollo, definiendo metas, revisiones, promoviendo el aprendizaje en equipo, desarrollo basado en arquitectura, desarrollo de componentes, constante comunicación con el usuario.

También se observo que en algunas empresas ya comienzan a utilizar herramientas automatizadas para agilizar las disciplinas de administración y desarrollo de software. Alguna de ellas se menciona a continuación:

- WBS
- Pruebas de código automático
- Herramientas Case para generación de código
- UML-Lenguaje de modelado unificado
- Programas de estimación de costos(Cocomo, CocomoII)
- Programas para estimación de tiempos(MS Project)
- Modelos de madurez (CMM)
- Artefactos (Casos de negocio, Especificaciones, valoración de estatus, seguimiento de ordenes de cambio en la base de datos)

Un dato interesante se dio en el sentido que se detecto que existe alguna correlación entre el uso de herramientas automatizadas en las disciplinas de administración y desarrollo de software las empresas y el incremento en el grado de éxito en sus proyectos.

Aunque el 100% de las empresas están interesadas en conocer mejores metodologías que ayuden a incrementar el grado de éxito en los proyectos, solo el 40% comento que están en disposición de invertir dinero en la adquisición de nuevas metodologías que les provean de un proceso lo suficientemente maduro para mejorar el porcentaje de éxitos de sus proyectos de software

Para las restantes empresas no cabe duda de las oportunidades existentes, debido a que cuentan con mayores deficiencias

Cabe mencionar que entre las empresas que demostraron que tienen mejores procesos de desarrollo, se encuentran 4 empresas que utilizan una metodología de prototipos, la cual al parecer arroja buenos resultados, pero sin ser 100% efectiva y existiendo aun áreas de oportunidad en dichas empresas.

Con el presente estudio se puede concluir que en las empresas mexicanas, "Es factible la implantación de una nueva metodología de administración de proyectos de desarrollo de software basado en los principios de la modelación orientada a objetos.

Siempre y cuando dichas empresas cuenten con los recursos financiaros y humanos para poder implantar una metodología compleja y costosa.”.

Estas empresas serán por lo regular medianas o grandes, que se dediquen a desarrollar software (sistemas complejos), ya sea para uso interno o para comercializar y que tengan la capacidad de invertir en la compra de herramientas automatizadas. Y la capacidad de crear el equipo que permita administrar una metodología compleja.

En este sentido solo las empresas que logren identificar las oportunidades y áreas de mejora, y que implementen políticas de mejora continua para lograr una eficiencia significativa, serán las que logren sobresalir en le negocio del desarrollo de software, el cual cuenta con un mercado estimado de 500 millones de dólares anuales. Y poder competir con empresas eficientes (Empresas indias) por las ganancias de dicho mercado.

Las oportunidades se han detectado, y las empresas de Monterrey ante una gran oportunidad de mejorar la calidad y el desempeño de sus proyectos de desarrollo de software.

Y en un futuro se puede pensar en una industria que pueda competir a nivel internacional a la par de países como la india. Esto podría beneficiar tanto a empleados, empresa como al país mismo, al contar con una industria que provea de un gran valor agregado como lo es la industria del desarrollo de software.

6.5 Sugerencias para Investigaciones Futuras

Los resultados de esta investigación dejan abiertas algunas interrogantes por lo que podrían originar nuevos estudios.

Los nuevos estudios o investigaciones que se podrían originar de la presente son:

1.- Elaborar un diagnostico inicial para empresas que desean incursionar en la desarrollo de software de manera iterativa.

2.- Elaborar un plan de implantación para las empresas que ya se han decidido a elaborar software siguiendo una metodología de desarrollo iterativo.

3.- Determinar las barreras de cambio que podrían interferir en la implantación de una metodología de proceso de desarrollo iterativo.

4.- Realizar un estudio de casos en el cual se de seguimiento al proceso de implantación de un enfoque unificado.

5.- Realizar estudios que permitan medir el desempeño de los proyecto en empresas que han implementado metodologías como el enfoque unificado.

Apéndice 1

El siguiente cuestionario es el diseño del cuestionario que se coloco en la página Web, a la cual las empresas tuvieron acceso.

Cuestionario
El presente cuestionario pretende identificar la forma en que las empresas administran algunos procesos críticos que se pueden vincular a los problemas que enfrentan los desarrolladores de software. Esto con el objetivo de encontrar las áreas de oportunidad con que cuentan las empresas mexicanas en la industria de desarrollo de software.

Datos Generales
Nombre Completo:
Correo Electrónico:
Teléfono para contacto:
Empresa:
Cargo que ocupa en la empresa:
Sexo:

Instrucciones: Favor de leer la pregunta y seleccionar el enunciado que más de apegue a la situación de su empresa.

1.- ¿Que metodología de desarrollo de software se utiliza en la empresa?
☐ Cascada
☐ Espiral
☐ Prototipos
☐ Ninguna
Otra _____

2.- Para administrar y estimar costos:
☐ Utilizamos herramientas de estimación de costos.
☐ Estimamos costos en base a la experiencia de proyectos pasados.
☐ No estima costos, sino hasta el final del proyecto.

3.- Para Administrar y estimar tiempos:
☐ Utilizamos herramientas de estimación de calendario.
☐ Estimamos Calendarios en base a la experiencia de proyectos pasados.
☐ No estima tiempos, sino hasta el final del proyecto.

4.- Para asegurar la calidad del software.
☐ Utilizamos herramientas y un plan establecido para asegurar la calidad del software.
☐ Asegurar la calidad de software se realiza de manera informal (No estructurada).
☐ No existe un plan para asegurar la calidad del software.

5.- Para minimizar riesgos:
☐ Utilizamos herramientas y un plan establecido para minimizar riesgos.
☐ La minimización de riesgos se realiza de manera informal (No estructurada).
☐ No existe un plan para minimizar riesgos.

6.- Para administrar los cambios:
☐ Utilizamos herramientas y un plan establecido para administrar los cambios.
☐ La minimización de riesgos se realiza de manera informal (No estructurada).
☐ No existe un plan para administrar cambios.

7.- Para asegurar la comunicación entre los involucrados:
☐ Utilizamos herramientas y un plan establecido para lograr la comunicación entre los involucrados.
☐ La Comunicación se realiza de manera informal (No estructurada).
☐ No existe un plan para asegurar la comunicación.

8.- Para administrar requerimientos:
☐ Utilizamos herramientas y un plan establecido para administrar requerimientos.
☐ La administración de requerimientos se realiza de manera informal (No estructurada).
☐ No existe un plan para administrar requerimientos.

9.- Con respecto a las deficiencias de la metodología de desarrollo de software que utiliza:
☐ Tiene conocimiento de sus deficiencias y esta trabajando en su solución.
☐ Tiene conocimiento de sus deficiencias, pero no se realiza nada al respecto.
☐ No tienen conocimiento de su deficiencia.

10.- En caso de que exista una metodología que ayude a mejorar las métricas de desempeño e incremento el grado de éxito de sus proyectos de desarrollo de software:
☐ Si le interesaría conocer acerca de ella, y si estaría dispuesto a invertir.
☐ Si le interesaría conocer acerca de ella, pero no tienen dinero para invertir.
☐ No le interesan mejores metodologías.

Segunda Parte

Instrucciones: Lea los siguientes enunciados y conteste de acuerdo al grado de aceptación que tenga el enunciado en su empresa, la escala va de menor a mayor aceptación, donde la menor es el 20% y el mayor el 100%, en caso de que el enunciado no aplique a su empresa, favor de seleccionar NA y en caso de no tener la información para contestar seleccionar NS.

11- Las estimaciones de los presupuestos de nuestros proyectos tienden en general a ser acertadas (Costo):

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ 80

12.- Los proyectos en los que participo se terminan en el tiempo estimado. (Tiempo)

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ Ns

13.- ¿Para diseñar un sistema tomamos en cuenta las diferentes dimensiones de la calidad? Como

Funcionalidad - Cubrir las necesidades del usuario.
Facilidad de uso- esfuerzo requerido para aprender su uso.
Confiabilidad - Soporte a fallas y caídas.
Eficiencia - Velocidad de respuesta, empleo de recursos.
Capacidad de mantenimiento - Minimizar esfuerzo requerido para administrar, modificar o validar.
Portabilidad - adaptación del software a diferentes ambientes

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ Ns

14.-Los riesgos de nuestros proyectos son evaluados y actualizados continuamente (incluyendo los planes de contingencia asociados) durante todo el ciclo de vida de cada proyecto. (Riesgos)

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ Ns

15.- Cuando cambian los requerimientos de software hacemos los ajustes pertinentes a los programas y documentos relacionados (modelos de diseño, manuales de usuario), así como a los planes de trabajo (calendarios de actividades y compromisos) Así como Podemos saber con facilidad quien ha utilizado un determinado elemento del software, que cambios le hizo, cuando lo modificó y por qué.(cambios)

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ Ns

16.- Existe buena comunicación entre nuestro equipo del proyecto (comunicación)

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ Ns

17.- Identificamos las amenazas y problemas más importantes del proyecto antes de comenzar su diseño y desarrollo (requerimientos)

Bajo	☐ 20%	☐ 40%	☐ 60%	☐ 80%	☐ 100%	**Alto**	☐ Na	☐ Ns

¡Gracias por participar!

Referencias Bibliograficas

Álvarez, Sofía (0000) "Método para la mejora de los procesos de software de la entidad productora de software", CUBA Disponible en: http://148.204.45.136:9000/labsiybd/eventos/4taller/sistemasinf/ConferenciaMejora.doc

AMITI. Desarrollo de software una oportunidad para México Asociación Mexicana de la Industria de Tecnologías de Información, A.C. [En línea] Disponible en: http://www.amiti.org.mx/biblioteca/El%20desarrollo%20de%20software%20Una%20oportunidad%20para%20M%C3%A9xico.PDF

Boehm, B.W. (1987) "Industrial Software Metrics Top 10 List" IEEE Software; Volumen 4 Numero 5 (Sep 1987)

Boehm, B.W. (1988) "A spiral Model of Software Development and Enhancement" IEEE Computer 21 (May 1988)

Booch, Grady (1997) "Quality Software and the Unified Modeling Language" paper

Rational Software, 1997

Chaos (1998) A White Paper "CAOS: A Recipe for Success", 1998 the Standish Group International Inc.

Castro Tapia, Iván Sebastián (2000) Tesis "Guía Híbrida para la ingeniería de software" 2000

Danhke (1998) "The reserch act: A Theoretical introduction to social methods"(2da Ed), New York; McGraw-Hill.

Franklin Steve (2003) "Applying Rational tools to a simple J2EE-based project Part 2: Starting up the project" Disponible en:

http://www-106.ibm.com/developerworks/rational/library/230.html, Acceso: 16 junio 2003

Galáz Solange (2002) "Ingeniería de software" disponible en: Http://www.monografias.com/trabajos5/inso/inso2.shtml

S. Globerson, O. Zwikael (2002) "Impact of the project manager on project management planning processes" Project Management Journal, V. 31, No. 3, 2002

Hernández, Fernández y Baptista (2003), "Metodología de la investigación", ED. McGraw-Hill, Mexico Df, 2003

Jones, Capers (1996) "Patterns of Software Systems Failure and Success", International Thompson Computes Press, Boston, Massachusetts, 1996.

Laplante Phillip (1999) "Key to Successful Software Development: Selected Readings", IEEE Networking Group1999.

Luzuriaga Juan Manuel (2002) "Inspecciones de Software" disponible en http://www.monografias.com/trabajos6/isof/isof.shtml

Merchant, Khozem (2001), "Sector hopes it can turn problems into solutions: Although heavily dependent on the slowing US economy and short on qualified personnel, Indian IT companies are looking remarkably robust" Financial Times; London; Feb 21, 2001

M&R (1998), Microsoft y Rational (1998), "A White Paper on the Benefits of Integrating Microsoft Solutions Framework and The Rational Process". Rational Software Corporation. Disponible en http://www.rational.com/uml/papers.

Pressman, Roger S. (1998), "Ingeniería de Software, un enfoque practico", Mc Graw Hill. España 1998

Paper New Dimensions of Project Management by Rational (www.rational.com)

Quintana Salinas, Edgar Vladimir (2000), Tesis "Guía para la evaluación de proyectos de tecnología de información" 2000

Rojas Guillermo (2003) "¿Por qué cometemos siempre los mismos errores en los proyectos de software?", itera, México DF.

Royce, Walter (1998) "Software Project Management, a Unified Framework", Addison-Wesley, 1998.

Schach Stephen R (2002). ”Object-Oriented and Classical Software Engineering”, Mc Graw Hill 2002

Sommerville Ian (1989). “Software Engineering”, Ed. Addison-Wesley, 1989.

Symons, Charles (0000). “Software Sizing and Estiming: Mk II FPA”. Chichester, John Wiley& Sons

Van Vliet Hans (2000). “Software Engineering: Principles and practice”, Second Edition, by john Wiley 2000

Williams. John D. (1996). “Managing Iteration in Object Oriented Projects"

IEEE computer, September 1996

Zermeño González Ricardo (2003), “Mercado de Desarrollo de Software”, Presentación Select, 2003