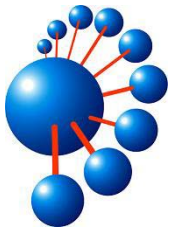
Departamento de Ingeniería Informática y Ciencias de la Computación

Facultad de Ingeniería

Universidad de Concepción



Gestión de Proyectos de Desarrollo de Software

Marcela Varas C.

2000

Contenidos.

0. Presentación.

Los proyectos de desarrollo de software se diferencian de los otros proyectos de ingeniería tradicional en la naturaleza lógica del producto software.

Recordemos que el software se *desarrolla*, no se fabrica en un sentido clásico. En todos los proyectos de ingeniería la buena calidad se adquiere mediante un buen diseño, pero en el caso del software, la etapa de construcción incide pobremente en su calidad, no así en la construcción de hardware o de una obra civil. Otra diferencia es que el software no se *estropea*, el paso del tiempo o males del entorno no inciden en el aumento de la tasa de fallas.

Así, no se puede gestionar un proyecto de desarrollo de software como si se tratara de un proyecto de fabricación.

La gestión del proyecto de software es el primer nivel del proceso de ingeniería de software, porque cubre todo el proceso de desarrollo. Para conseguir un proyecto de software fructífero se debe comprender el ámbito del trabajo a realizar, los riesgos en los que se puede incurrir, los recursos requeridos, las tareas a llevar a cabo, el esfuerzo (costo) a consumir y el plan a seguir.

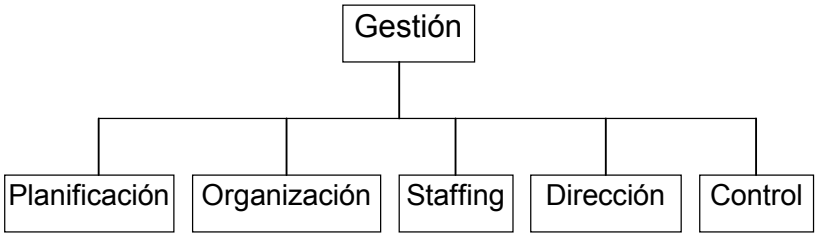
Este apunte cubre los tópicos más relevantes de la gestión de proyectos aplicada al área específica de la ingeniería de software, de modo de proveer las bases conceptuales necesarias para ejercer competentemente el cargo de jefe de proyectos de desarrollo de software u otro cargo similar.

1. Gestión.

Gestión son todas las actividades y tareas ejecutadas por una o más personas con el propósito de planificar y controlar las actividades de otros para alcanzar un objetivo o completar una actividad que no puede ser realizada por otros actuando independientemente.

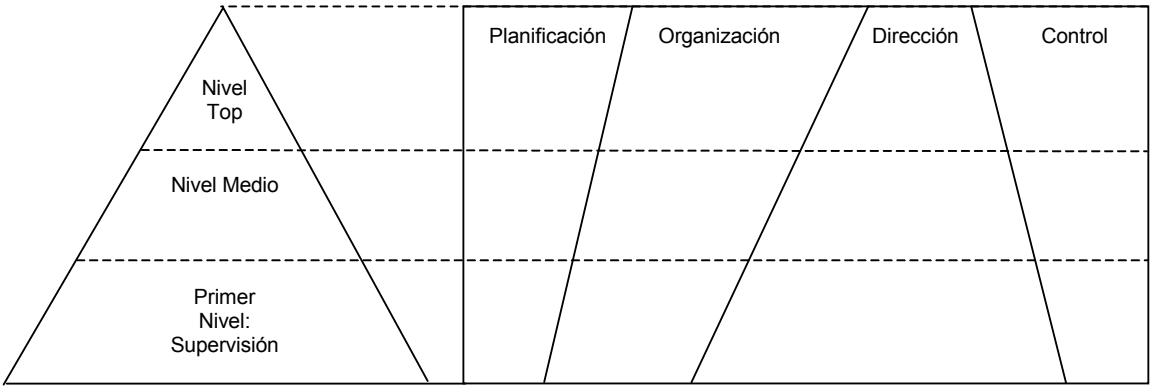
1.1 Definición de las actividades de gestión.

Planificación: Predeterminación de un curso de acción para alcanzar los objetivos organizacionales.
Organización: Arreglo de las relaciones entre las unidades de trabajo para el cumplimiento de objetivos y el otorgamiento de responsabilidad y autoridad para obtener esos objetivos.
Staffing: Selección y entrenamiento de personas para puestos en la organización.
Dirección: Creación de una atmósfera que apoye y motive a la gente para alcanzar los resultados finales deseados.
Control: Establecimiento, medición y evaluación del desempeño de las actividades a través de los objetivos planeados.

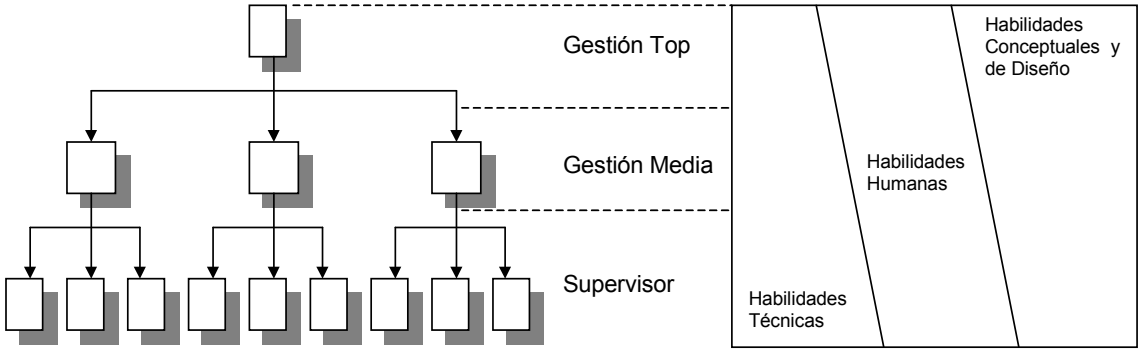


1.2 Universalidad de la gestión

Los administradores realizan las mismas funciones independientemente de su lugar en la estructura organizacional o el tipo de empresa.



1.3 Habilidades de Gestión y la Jerarquía Organizacional.



Habilidades Técnicas. Conocimiento y pericia en actividades que involucran métodos, procesos y procedimientos. Esto implica trabajar con herramientas y técnicas específicas.

Habilidades Humanas. Habilidad para trabajar con gente, del esfuerzo cooperativo, del trabajo en equipo, de la creación de un ambiente donde la gente se sienta segura y libre de expresar sus opiniones.

Habilidades Conceptuales. Habilidad para ver la "Imagen Completa" (the big picture), para reconocer los elementos relevantes de una situación, y para entender las relaciones entre los elementos.

Habilidades de Diseño. Habilidad para resolver problemas de manera que beneficie a la empresa. Para ser efectivos, particularmente en los niveles organizacionales altos, los gerentes deben ser capaces de más que sólo ver un problema. Si los gerentes solamente ven los problemas y se transforman en "observadores de problemas", fallarán. Deben tener, además, la habilidad de un buen ingeniero de diseño para generar una solución práctica a un problema.

1.4 Planificación de un Proyecto de Ingeniería de Software.

La planificación involucra la especificación de objetivos y metas para un proyecto y las estrategias, políticas, planes y procedimientos para alcanzarlos.

Todo proyecto de ingeniería de software debe partir con un buen plan. La planificación es necesaria por la existencia de incertezas sobre el ambiente del proyecto software y sobre fuentes externas. La planificación enfoca su atención en las metas del proyecto, riesgos potenciales y problemas que puedan interferir con el cumplimiento de esas metas.

Los principales problemas en la planificación de un proyecto de ingeniería de software incluyen los siguientes:

- Requerimientos incorrectos e incompletos.
- Muchas especificaciones de requerimientos son inestables y sujetas a cambios mayores.
- La planificación no se lleva a cabo por la creencia errónea de que es una pérdida de tiempo y los planes cambiarán de todos modos.
- La planificación de costos y plazos no es actualizada y se basa en necesidades de mercadeo y no de los requerimientos del sistema.
- Es difícil estimar el tamaño y complejidad del proyecto de software de modo de realizar una estimación de costos y plazos realista.
- Los costos y plazos no son re estimados cuando los requerimientos del sistema o el ambiente de desarrollo cambia.
- No se manejan factores de riesgo.
- La mayoría de las organizaciones de desarrollo de software no recolectan datos de proyectos pasados.
- Las compañías no establecen políticas o procesos de desarrollo de software.

1.4.1 Actividades que se derivan de la planificación.

- Fijar los objetivos y metas
- Desarrollar estrategias
- Desarrollar políticas
- Anticipar futuras situaciones
- Conducir un establecimiento de riesgos
- Determinar posibles cursos de acción
- Tomar decisiones de planificación
- Fijar procedimientos y reglas
- Desarrollar los planes del proyecto
- Preparar presupuestos
- Documentar los planes del proyecto.

1.5 Organización de un proyecto de Ingeniería de Software

Involucra desarrollar una estructura organizacional efectiva y eficiente para asignar y completar las tareas del proyecto y establecer las relaciones de autoridad y responsabilidad entre las tareas.

Los principales problemas en la organización de un proyecto de ingeniería de software incluyen los siguientes:

- Es difícil determinar la mejor estructura organizacional para una organización y/o ambiente particular (por ejemplo tipo proyecto, funcional o matriz) para gestionar el proyecto.
- Una estructura organizacional puede dejar responsabilidades para algunas actividades y tareas del proyecto poco claras o indefinidas.
- Mucho personal de desarrollo de software no acepta una organización matricial.
- Muchos líderes de equipo esperan desarrollarse tanto técnicamente como en la gestión de su equipo de trabajo.

1.5.1 Actividades que se derivan de la organización

- Identificar y agrupar las funciones, actividades y tareas del proyecto.
- Seleccionar estructuras organizacionales
- Crear posiciones organizacionales
- Definir responsabilidades y autoridades.

- Establecer el perfil de cada puesto
- Documentar las decisiones organizacionales

1.6 Consiguiendo Personal para un proyecto de Ingeniería de Software

Consiste en todas aquellas actividades que involucran llenar (y mantener llenos) los puestos que fueron establecidos en la estructura organizacional del proyecto. Esto incluye selección de candidatos, entrenamiento y otros.

Los principales problemas en esta etapa son:

- Los jefes de proyecto son frecuentemente seleccionados por su habilidad para programar o realizar tareas de ingeniería en vez de su habilidad de gestión (pocos ingenieros son buenos gerentes)
- La productividad de los programadores, analistas e ingenieros de software varía mucho de individuo en individuo.
- Hay grandes cambios en el equipo de un proyecto software, especialmente en aquellos organizados matricialmente.
- Las universidades no están produciendo un número suficiente de ingenieros que entiendan el proceso de la ingeniería de software o gestión de proyectos.
- Los planes de entrenamiento para desarrolladores individuales de software no se desarrollan o mantienen.

1.6.1 Actividades derivadas:

- Llenar los puestos de la organización.
- Asimilar al personal recientemente asignado
- Educar o entrenar al personal
- Proveer de desarrollo general
- Evaluar y valorar al personal
- Compensar

1.7 Dirección de un proyecto de ingeniería de software

Dirigir un proyecto de ingeniería de software consiste en aquellas actividades de gestión que involucran aspectos interpersonales y de motivación por medio de las cuales el personal del

proyecto entiende y contribuye a alcanzar los objetivos del proyecto. Una vez que los subordinados son entrenados y orientados, el jefe de proyecto tiene una responsabilidad continua por clarificar sus asignaciones, guiándolos hacia la mejora de la productividad, y motivándolos a trabajar con entusiasmo y confianza hacia las metas del proyecto.

Los principales problemas en la dirección son:

- Fallas para tener una comunicación efectiva entre las entidades del proyecto y aquellas que no pertenecen al proyecto.
- El dinero no es un motivador suficiente para los desarrolladores de software.
- Las compañías y los jefes no poseen las técnicas y herramientas apropiadas para motivar a los ingenieros de software.
- Los clientes y gerentes no reconocen el impacto potencial en el software causado por un aparentemente cambio trivial, por ejemplo, ellos creen que es “sólo un problema simple de programación”.

1.7.1 Actividades de dirección:

- proveer liderazgo
- supervisar personal
- delegar autoridad
- motivar personal
- construir equipos
- coordinar actividades
- facilitar comunicaciones
- resolver conflictos
- manejar cambios
- documentar las decisiones de dirección.

1.8 Control de un proyecto de ingeniería de software

Controlar es el conjunto de actividades de gestión utilizadas para asegurar que el proyecto va de acuerdo a lo planificado. El desempeño y los resultados se miden contra los planes, se notan las desviaciones, y se toman acciones correctivas.

El control es un sistema de retroalimentación que provee información acerca de cuan bien va el proyecto. El control responde las preguntas

1. ¿Está el proyecto en itinerario?
2. ¿Está dentro de los costos?

3. ¿Existen problemas potenciales que causen retrasos en alcanzar los requerimientos dentro del presupuesto y plazo?

Los principales problemas en el control son:

- Muchos métodos de control de proyectos de desarrollo de software confían en los gastos del presupuesto para medir el "progreso" sin considerar el trabajo que lo acompaña.
- La visibilidad del progreso en un proyecto de software es difícil de medir.
- La calidad no es requerida, monitoreada o controlada.
- A menudo los estándares para el desarrollo de software no están escritos o, si lo están, no se fuerzan.
- El cuerpo de conocimiento llamado métricas de software (usadas para medir productividad, calidad, y progreso de un producto software) no está completamente desarrollado.

1.8.1 Actividades de control:

- Desarrollar estándares de desempeño
- Establecer sistemas de monitoreo y reportes
- Medir y analizar resultados
- Iniciar acciones correctivas
- Recompensar y disciplinar
- Documentar los métodos de control.

2. Los problemas y errores comunes.

Los desarrolladores, directivos y clientes normalmente tienen buenas razones para tomar las decisiones que toman, y la apariencia seductora de los errores clásicos es una de las razones de que esos errores se cometan tan a menudo. Pero debido a que se han cometido muchas veces, sus consecuencias se han hecho fáciles de predecir. Y los errores rara vez producen los resultados que la gente espera.

2.1 Personas

A continuación aparecen algunos de los errores clásicos relacionados con las personas.

1: Motivación débil. Estudio tras estudio ha mostrado que la motivación probablemente tiene mayor efecto sobre la productividad y la calidad que ningún otro factor (Boehm 1981). Ejemplo: directivos que a lo largo de todo el proyecto toman medidas que minan la moral: como dar ánimos a diario al principio para pedir horas extras en la mitad, y como irse de vacaciones mientras el equipo esta trabajando incluso los días de fiesta, para dar recompensas al final del proyecto que resultan ser de menos de un dólar por cada hora extra.

2: Personal mediocre. Después de la motivación, la capacidad individual de los miembros del equipo, así como sus relaciones como equipo, probablemente tienen la mayor influencia en la productividad (Boehm, 1981; Lakhanpal, 1993). Contratar apurando el fondo del barril supondrá una amenaza al desarrollo. Ejemplo, hacer la selección del personal buscando quién puede contratarse más rápido, en vez de quién realizará la mayoría del trabajo durante la vida del proyecto. Esta técnica consigue un inicio rápido del proyecto, pero no determina un final rápido.

3: Empleados problemáticos incontrolados. Un fallo al tratar con personal problemático también amenaza la velocidad de desarrollo. Un fallo al tomar una decisión cuando se trata con un empleado problemático es una de las quejas más comunes que tienen los miembros del equipo respecto de sus responsabilidades (Larson y La fasto, 1989). Ejemplo, el equipo sabe que uno de ellos es una manzana podrida, pero el jefe del equipo no hace nada. El resultado es predecible: rehacer el trabajo de la manzana podrida.

4: Hazañas. Algunos desarrolladores de software ponen un gran énfasis en la realización de hazañas en los proyectos (Bach, 1995). Pero lo que hacen tiene más de malo que de bueno. Ejemplo, los directivos de nivel medio dan mayores aplausos a actitudes del tipo *ser capaz de* que a los progresos firmes y consistentes y a los informes significativos de progreso. El resultado es un modelo de planificación al límite en el que las amenazas de desajuste del plan no se detectan, no se conocen o ni se informan a la cadena de directivos hasta el último minuto. Un pequeño equipo de desarrollo y sus jefes inmediatos toman como rehenes a una compañía entera por no admitir que tiene problemas para cumplir su plan. El énfasis en los comportamientos heroicos fomenta correr un riesgo extremo, e impide la cooperación entre los múltiples elementos que contribuyen al proceso de desarrollo del software.

Algunos directivos fomentan el comportamiento heroico cuando se concentran con demasiada firmeza en actitudes del tipo "ser capaz de". Elevando estas actitudes por encima de informes del estado exactos y a veces pesimistas, los directivos de estos proyectos coartan su capacidad de tomar medidas correctivas. Ni siquiera saben que tienen que emprender acciones correctoras hasta que el daño ya está hecho. Como dijo Tom DeMarco, las actitudes <<ser capaz de>> convierten pequeños contratiempos en auténticos desastres (DeMarco, 1995).

5: Añadir más personal a un proyecto retrasado. Este es quizás el más clásico de los errores clásicos. Cuando un proyecto se alarga, añadir más gente puede quitar más productividad a los miembros del equipo existente de la que añaden los nuevos miembros. Fred Brooks compara añadir gente a un proyecto retrasado con echar gasolina en un fuego (Brooks, 1975).

6: Oficinas repletas y ruidosas. La mayoría de los desarrolladores consideran sus condiciones de trabajo como insatisfactorias. Alrededor del 60 por 100 indican que no tienen suficiente silencio ni privacidad (DeMarco y Lister, 1987). Los trabajadores que están en oficinas silenciosas y privadas tienden a funcionar significativamente mejor que aquellos que ocupan cubículos en salas ruidosas y repletas. Los entornos repletos y ruidosos alargan los planes de desarrollo.

7: Fricciones entre los clientes y los desarrolladores. Las fricciones entre los clientes y los desarrolladores pueden presentarse de distintas formas. A los clientes pueden parecerles que los desarrolladores no cooperan cuando rehusan comprometerse con el plan de desarrollo que desean los clientes o cuando fallan al entregar lo prometido. A los desarrolladores puede parecerles que los clientes no son razonables porque insisten en planes irreales o cambios en los requerimientos después de que éstos hayan sido fijados. Pueden ser simplemente conflictos de personalidad entre dos grupos.

El principal efecto de esta fricción es la mala comunicación, y los efectos secundarios de la mala comunicación incluyen el pobre entendimiento de los requerimientos, pobre diseño de la interfaz de usuario y, en el peor caso, el rechazo del cliente a aceptar el producto acabado. En el caso medio, las fricciones entre clientes y desarrolladores de software llegan a ser tan severas que ambas partes consideran la cancelación del proyecto (Jones, 1994). Para

remediar estas fricciones se consume tiempo, y distraen tanto a desarrolladores como a clientes del trabajo real en el proyecto.

8: Expectativas pocos realistas. Una de las causas más comunes de fricciones entre los desarrolladores y sus clientes o los directivos son las expectativas poco realistas. Ejemplo, no tener razones técnicas para pensar que un software se podrá desarrollar en 6 meses, pero ése es el plazo en que lo quiere el comité ejecutivo de la compañía. La incapacidad del jefe de proyecto para corregir esta expectativa irreal será la principal fuente de problemas.

En otros casos, los directivos o los desarrolladores de un proyecto se buscan problemas al pedir fondos basándose en estimaciones de planificación demasiado optimistas. Aunque por sí mismas las expectativas irreales no alargan el plan, contribuyen a la percepción de que el plan de desarrollo es demasiado largo, y de que puede ser malo. Una inspección de Standish Group marcó las expectativas realistas como uno de los cinco factores principales necesarios para asegurar el éxito de los proyectos internos de software de gestión (Standish Group, 1994).

9: Falta de un promotor efectivo del proyecto. Para soportar muchos de los aspectos del desarrollo rápido es necesario un promotor del proyecto de alto nivel, incluyendo una planificación realista, el control de cambios y la introducción de nuevos métodos de desarrollo. Sin un promotor ejecutivo efectivo, el resto del personal de alto nivel de la empresa puede forzar a que se acepten fechas de entrega irreales o hacer cambios que debiliten el proyecto. El consultor australiano Rob Thomstt afirma que la falta de un promotor efectivo garantiza virtualmente el fracaso del proyecto (Thomstt, 1995).

10: Falta de participación de los implicados. Todos los principales participantes del esfuerzo de desarrollo de software deben implicarse en el proyecto. Incluyendo a los promotores, ejecutivos, responsables del equipo, miembros del equipo, personal de ventas, usuarios finales, clientes y cualquiera que se juegue algo con el proyecto. La cooperación estrecha sólo se produce si se han implicado todos los participantes, permitiendo una coordinación precisa del esfuerzo para el desarrollo rápido, que es imposible conseguir sin una buena participación.

11: Falta de participación del usuario. La inspección de Standish Group descubrió que la razón número uno de que los proyectos de Sistemas de Información tuviesen éxito es la

implicación del usuario (Standish Group, 1994). Los proyectos que no implican al usuario desde el principio corren el riesgo de que no se comprendan los requerimientos del proyecto, y son vulnerables a que se consuma tiempo en prestaciones que más tarde retrasarán el proyecto.

12: Política antes que desarrollo. Larry Constantine indicó que si hay cuatro tipos diferentes de orientaciones políticas (Constantine, 1995a). Los "políticos" están especializados en la "gestión", centrándose en las relaciones con sus directivos. Los "investigadores" se centran en explorar y reunir la información. Los "aislacionistas" están solos, creando fronteras para el proyecto que mantienen cerradas a los que no son miembros del equipo. Los "generalistas" hacen un poco de todo: establecen relaciones con sus directivos, realizan investigaciones y exploran actividades, y se coordinan con otros equipos como parte de su modo de trabajo. Constantine indicó que inicialmente los equipos políticos y generalistas están bien vistos por los directivos de alto nivel. Pero después de un año y medio, los equipos políticos llegan a la muerte súbita. Primar la política en vez de los resultados es fatal para el desarrollo orientado a la velocidad.

13: Ilusiones. Muchos problemas del desarrollo del software se deben a la ilusión. Cuántas veces hemos escuchado cosas como éstas a distintas personas:

"Ninguno de los miembros del proyecto cree realmente que pueda completarse el proyecto de acuerdo con el plan que tienen, pero piensan que quizás si trabajan duro, y nada va mal, y tienen un poco de suerte, serán capaces de concluir con éxito".

"Nuestro equipo no hace mucho trabajo para la coordinación de las interfaces entre las distintas partes del producto, pero tenemos una buena comunicación para otras cosas, y las interfaces son relativamente simples, así que probablemente sólo necesitaremos un día o dos para eliminar los errores"

"Sabemos que contamos con un desarrollador externo de poco talento para el subsistema de la base de datos, y que es difícil ver cómo va a acabar el trabajo con los niveles de personal que ha especificado en su propuesta. No tienen tanta experiencia como algunos de los demás desarrolladores externos, pero puede que compensen con energía lo que les falta en experiencia. Probablemente acaben a tiempo"

"No necesitamos reflejar la última lista de cambios en el prototipo para el cliente. Estoy seguro de que por ahora sabemos lo que quiere."

"El equipo está diciendo que realizará un esfuerzo extraordinario para cumplir con la fecha de entrega, y que no han llegado a su primer hito por pocos días, pero creo que alcanzarán éste a tiempo."

Las ilusiones no son sólo optimismo. Realmente consisten en cerrar los ojos y esperar que todo funcione cuando no se tienen las bases razonables para pensar que será así. Las ilusiones al comienzo del proyecto llevan a grandes explosiones al final. Impiden llevar a cabo una planificación coherente y pueden ser la raíz de más problemas en el software que todas las otras causas combinadas.

2.2 Proceso

Los errores relacionados con el proceso malgastan el talento y el esfuerzo del personal. A continuación se muestran algunos de los peores errores relacionados con el proceso.

14: Planificación excesivamente optimista. Los retos a los que se enfrenta alguien que desarrolla una aplicación en tres meses son muy diferentes de aquellos a los que se enfrenta alguien que desarrolla una aplicación que necesita un año. Fijar un plan excesivamente optimista predispone a que el proyecto falle por infravalorar el alcance del proyecto, minando la planificación efectiva, y reduciendo las actividades críticas para el desarrollo, como el análisis de requerimientos o el diseño. También supone una excesiva presión para los desarrolladores, quienes a largo plazo se ven afectados en su moral y su productividad.

15: Gestión de riesgos insuficiente. Algunos errores no son lo suficientemente habituales como para considerarlos clásicos. Son los llamados "riesgos". Como con los errores clásicos, si no ejercemos una gestión activa de los riesgos, con qué sólo vaya mal una cosa se pasará de tener un proyecto con un desarrollo rápido a uno con un desarrollo lento. El fallo de no gestionar uno solo de estos riesgos es un error clásico.

16: Fallos de los contratistas. Las compañías a veces contratan la realización de partes de un proyecto cuando tienen demasiada prisa para hacer el trabajo en casa. Pero los

contratados frecuentemente entregan su trabajo tarde, con una calidad inaceptable o que falla al no coincidir con las especificaciones (Boehm, 1989). Riesgos como requerimientos inestables o interfaces mal definidas pueden ser enormes cuando un contratado entre en escena. Si las relaciones con los contratados no se gestionan cuidadosamente, la utilización de desarrolladores externos pueden retardar el proyecto en vez de acelerarlo.

17: Planificación insuficiente. Si no planificamos para conseguir un desarrollo rápido, no podemos esperar obtenerlo.

18: Abandono de planificación bajo presión. Los equipos de desarrollo hacen planes y rutinariamente los abandonan cuando se tropiezan con un problema en la planificación (Humphrey, 1989). El problema no está en el abandono del plan, sino más bien en fallar al no crear un plan alternativo, y caer entonces en el modo de trabajo de codificar y corregir. Ejemplo, un equipo abandona su plan después de fallar en la primera entrega, y esto es lo habitual. A partir de este punto, el trabajo no tiene coordinación ni elegancia.

19: Pérdida de tiempo en el inicio difuso. El "inicio difuso" es el tiempo que transcurre antes de que comience el proyecto; este tiempo normalmente se pierde en el proceso de aprobar y hacer el presupuesto. No es poco común que un proyecto desperdicie meses o años en un inicio difuso, y entonces se está a las puertas de un plan agresivo. Es mucho más fácil y barato y menos arriesgado suprimir unas pocas semanas o meses del inicio difuso en vez de comprimir el plan de desarrollo en ese mismo tiempo.

20: Escatimar en las actividades iniciales. Los proyectos se aceleran intentando acortar las actividades "no esenciales", y puesto que el análisis de requerimientos, la arquitectura y el diseño no producen código directamente, son los candidatos fáciles.
Los resultados de este error, también conocido como "saltar a la codificación", son todos demasiado predecibles. Los proyectos que normalmente escatiman en sus actividades iniciales tendrán que hacer ese trabajo en otro momento, con un costo de 10 a 100 veces superior a haberlo hecho bien inicialmente (Fagan, 1976; Boehm y Papaccio, 1988). Si no podemos encontrar cinco horas para hacer el trabajo correctamente la primera vez, ¿cómo vamos a encontrar 50 para hacerlo correctamente más tarde?

21: Diseño inadecuado. Un caso especial de escatimar en las actividades iniciales es el diseño inadecuado. Proyectos acelerados generan un diseño indeterminado, no asignado suficiente tiempo para él y originado un entorno de alta presión que hace difícil la posibilidad de considerar alternativas en el diseño. El énfasis en el diseño está más orientado a la conveniencia que a la calidad, por lo que necesitará varios ciclos de diseño de poder finalizar completamente el sistema.

22: Escatimar en el control de calidad. En los proyectos que se hacen con prisa se suele cortar por lo sano, eliminando las revisiones del diseño y del código, eliminando la planificación de las pruebas y realizando sólo pruebas superficiales. Acortar en un día las actividades de control de calidad al comienzo del proyecto probablemente supondrá de 3 a 10 días de actividades finales (Jones, 1994).

23: Control insuficiente de la directiva. Poco control de la directiva para detectar a tiempo los signos de posibles retrasos en el plan, y los pocos controles definidos al comienzo se abandonan cuando el proyecto comienza a tener problemas. Antes de encarrilar un proyecto, en primer lugar debemos ser capaces de decir si va por buen camino.

24: Convergencia prematura o excesivamente frecuente. Bastante antes de que se haya programado entregar un producto, hay un impulso para preparar el producto para la entrega, mejorar el rendimiento del producto, imprimir la documentación final, incorporar entradas en el sistema final de ayuda, pulir el programa de instalación, eliminar las funciones que no van a estar listas a tiempo y demás. En proyectos hechos con prisa, hay una tendencia a forzar prematuramente la convergencia. Puesto que no es posible forzar la convergencia del producto cuando se desea, algunos proyectos de desarrollo rápido intentan forzar la convergencia media docena de veces o más antes de que finalmente se produzca. Los intentos adicionales de convergencia no benefician al producto, Sólo son una pérdida de tiempo y prolongan el plan.

25: Omitir tareas necesarias en la estimación. Si la gente no guarda cuidadosamente datos de proyectos anteriores, olvida las tareas menos visibles, pero son tareas que se han de añadir. El esfuerzo omitido suele aumentar el plan de desarrollo en un 20 o 30 por 100 (Van Genuchten, 1991).

26: Planificar ponerse al día más adelante. Un tipo de reestimación es responder inapropiadamente el retraso del plan. Si hemos trabajado en un proyecto durante 6 meses, y hemos empleado tres meses en llegar al hito correspondiente a los dos meses ¿qué hacer?. En muchos proyectos simplemente se plantea recuperar el retraso más tarde, pero nunca se hace. Aprenderemos más del producto conforme lo estamos construyendo, incluyendo más sobre lo que nos llevará construirlo. Estos conocimientos necesitan reflejarse en la reestimación del plan.

Otro tipo de error es la reestimación que se debe a cambios en el producto. Si el producto que estamos construyendo cambia, la cantidad de tiempo necesaria para construirlo cambiará también. El crecimiento de las nuevas prestaciones sin ajustar el plan garantiza que no se alcanzará la fecha de entrega.

27: Programación a destajo. Algunas organizaciones creen que la codificación rápida, libre, tal como salga, es el camino hacia el desarrollo rápido. Piensan que si los desarrolladores están lo suficientemente motivados, pueden solventar cualquier obstáculo. Este enfoque muchas veces se presenta como un enfoque "empresarial" al desarrollo de software, pero realmente es sólo la envoltura del viejo paradigma a destajo combinado con una planificación ambiciosa, y esta combinación raras veces funciona. Es un ejemplo de que dos negaciones no constituyen una verdad.

2.3 Producto

A continuación se muestran los errores clásicos relacionados con la forma en la que se define el producto.

28: Exceso de requerimientos. Algunos proyectos tienen más requerimientos de los que necesitan, desde el mismo inicio. La eficiencia se fija como requisito más a menudo de lo que es necesario, y puede generar una planificación del software innecesariamente larga. Los usuarios tienden a interesarse menos en las prestaciones complejas que en las de las secciones de marketing o de desarrollo, y las prestaciones complejas alargan desproporcionadamente el plan de desarrollo.

29: Cambio de las prestaciones. Incluso si hemos evitado con éxito los requerimientos excesivos, los proyectos sufren como media sobre un 25 por 100 de cambios en los requerimientos a lo largo de su vida (Jones, 1994). Un cambio de este calibre puede producir un aumento en el plan de al menos un 25 por 100, lo que puede ser fatal para los proyectos.

30: Desarrolladores meticulosos. Los desarrolladores encuentran fascinante la nueva tecnología, y a veces están ansiosos por probar nuevas prestaciones de su lenguaje o entorno, o por crear su propia implementación de una utilidad bonita que han visto en otro producto, la necesite o no su producto. El esfuerzo requerido para diseñar, implementar, probar, documentar o mantener estas prestaciones innecesarias alarga el plan.

31: Tiras y aflojas en la negociación. Cuando un directivo aprueba un retraso en el plan de un proyecto que progresa más lento de los esperado, y entonces añade tareas completamente nuevas después de un cambio en el plan, se produce una situación curiosa. La razón subyacente de esto es difícil de localizar, puesto que el directivo que aprueba el retardo en el plan lo hace sabiendo implícitamente que el plan estaba equivocado. Pero una vez que se corrige, la misma persona realiza acciones explícitas para volver a equivocarse. Esto sólo puede ir en contra del plan.

32: Desarrollo orientado a la investigación. Seymour Cray, el diseñador de los supercomputadores Cray, dijo que no intentaba sobrepasar los límites de la ingeniería en más de dos áreas a la vez, porque el riesgo de un fallo es demasiado alto (Gilb, 1988). Muchos proyectos software deberán aprender la lección de Cray. Si el proyecto fuerza los límites de la informática porque necesita la creación de nuevos algoritmos o de nuevas técnicas de computación, no estamos desarrollando software. Los planes de desarrollo de software son razonablemente predecibles; los planes en la investigación sobre software ni siquiera son predecibles teóricamente.

Si el producto tiene objetivos que pretenden aumentar los conocimientos existentes, como algoritmos, velocidad, utilización de la memoria y demás, debemos asumir que la planificación es altamente especulativa. Si queremos mejorar el estado del arte y tenemos algún otro punto débil en el proyecto, recortes de personal, debilidades en el personal, requerimientos vagos, interfaces inestables con contratados externos, etc., podemos tirar por la ventana la

planificación prevista. Si queremos superar el estado del arte por todos los medios, hagámoslo. ¡Pero no debemos esperar hacerlo rápidamente!

2.4 Tecnología

El resto de los errores clásicos están relacionados con el uso correcto o incorrecto de la tecnología moderna.

33: Síndrome de la panacea. A veces se confía demasiado en las ventajas proclamadas de tecnologías que no se han usado antes (generadores de informes, diseño orientado a objetos y C++) y se tiene poca información sobre lo buenas que serían en un entorno de desarrollo concreto. Cuando el equipo del proyecto se aferra sólo a una nueva técnica, una nueva tecnología o un proceso rígido, y espera resolver con ello sus problemas de planificación, está inevitablemente equivocado (Jones, 1994).

34: Sobreestimación de las ventajas del empleo de nuevas herramientas o métodos. Las organizaciones mejoran raramente su productividad a grandes saltos, sin importar cuántas nuevas herramientas o métodos empleen o lo bueno que sean. Los beneficios de las nuevas técnicas son parcialmente desplazados por las curvas de aprendizaje que llevan asociadas, y aprender a utilizar nuevas técnicas para aprovecharlas al máximo lleva su tiempo. Las nuevas técnicas también suponen nuevos riesgos, que sólo descubriremos usándolas. Más bien experimentaremos mejoras lentas y continuas en un pequeño porcentaje por proyecto en lugar de grandes saltos.

Un caso especial de sobreestimaciones de las mejoras se produce cuando se reutiliza código de proyectos anteriores. Este tipo de reutilización puede ser una técnica muy efectiva, pero el tiempo que se gana no es tan grande como se espera.

35: Cambiar de herramientas a mitad del proyecto. Es un viejo recurso que funciona raramente. A veces puede tener sentido actualizar incrementalmente dentro de la misma línea de productos, de la versión 3 a la 3.1, o incluso a la 4. Pero cuando estamos a la mitad de un proyecto, la curva de aprendizaje, rehacer el trabajo y los inevitables errores cometidos con una herramienta totalmente nueva, normalmente anulan cualquier posible beneficio.

36: Falta de control automático del código fuente. Un fallo en la utilización del control automático del código fuente expone a los proyectos a riesgos innecesarios. Sin él, si dos desarrolladores están trabajando en la misma parte del programa, deben coordinar su trabajo manualmente. Deberían ponerse de acuerdo para poner la última versión de cada archivo en el directorio maestro y verificarlos con los demás antes de copiarlas en este directorio. Pero invariablemente alguno sobreescribirá el trabajo del otro. Se desarrolla nuevo código con interfaces desfasadas, y después se tiene que rediseñar el código al descubrir que se ha utilizado una versión equivocada de la interfaz. Los usuarios avisan de errores que no podemos reproducir porque no hay forma de volver a crear los elementos que han utilizado. Como media, los cambios manual del código fuente no deberían crecer (Jones, 1994).

3. Ciclo de Vida del Software.

3.1 Ciclo de Vida del Software.

Un modelo de ciclo de vida define el estado de las fases a través de las cuales se mueve un proyecto de desarrollo de software.
El primer ciclo de vida del software, “Cascada”, fue definido por Winston Royce a fines del 70. Desde entonces muchos equipos de desarrollo han seguido este modelo. Sin embargo, ya desde 10 a 15 años atrás, el modelo cascada ha sido sujeto a numerosas críticas, debido a que es restrictivo y rígido, lo cual dificulta el desarrollo de proyectos de software. En su lugar, muchos modelos nuevos de ciclo de vida han sido propuestos, incluyendo modelos que pretenden desarrollar software más rápidamente, o más incrementalmente o de una forma más evolutiva, o precediendo el desarrollo a escala total con algún conjunto de prototipos rápidos.

3.1.1 Definición de un Modelo de Ciclo de Vida.

Un modelo de ciclo de vida de software es una visión de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas.
Un modelo de ciclo de vida del software:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo, y
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software.

Así, los modelos por una parte suministran una guía para los ingenieros de software con el fin de ordenar las diversas actividades técnicas en el proyecto, por otra parte suministran un marco para la administración del desarrollo y el mantenimiento, en el sentido en que permiten estimar recursos, definir puntos de control intermedios, monitorear el avance, etc.

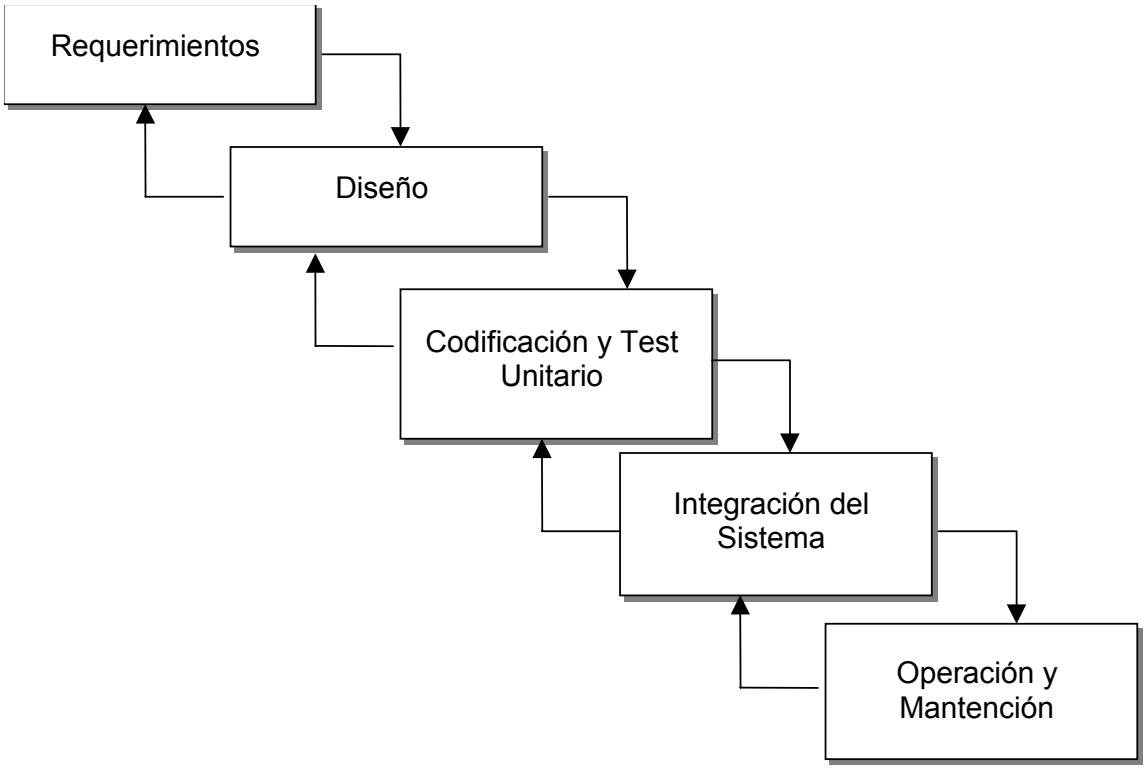
3.1.2 Alternativas de Modelos de Ciclo de Vida.

3.1.2.1 Modelo Cascada.

Este es el más básico de todos los modelos, y sirve como bloque de construcción para los demás modelos de ciclo de vida. La visión del modelo cascada del desarrollo de software es muy simple; dice que el desarrollo de software puede ser a través de una secuencia simple de fases. Cada fase tiene un conjunto de metas bien definidas, y las actividades dentro de una fase contribuye a la satisfacción de metas de esa fase o quizás a una subsecuencia de metas de la fase. Las flechas muestran el flujo de información entre las fases. La flecha de avance muestra el flujo normal. Las flechas hacia atrás representan la retroalimentación.

El modelo de ciclo de vida cascada, captura algunos principios básicos:

- Planear un proyecto antes de embarcarse en él.
- Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
- Documentar los resultados de cada actividad.
- Diseñar un sistema antes de codificarlo.
- Testear un sistema después de construirlo.



Modelo de Ciclo de Vida Cascada.

Una de las contribuciones más importantes del modelo cascada es para los administradores, posibilitándoles avanzar en el desarrollo, aunque en una escala muy bruta.

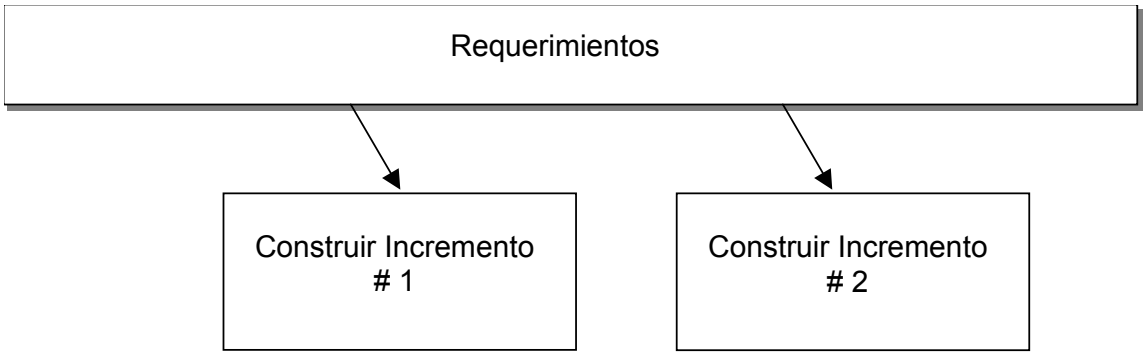
3.1.2.2 Modelo de Desarrollo Incremental.

Los riesgos asociados con el desarrollo de sistemas largos y complejos son enormes. Una forma de reducir los riesgos es construir sólo una parte del sistema, reservando otros aspectos para niveles posteriores. El desarrollo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos del sistema. Típicamente, un documento de requerimientos es escrito al capturar todos los requerimientos para el sistema completo.

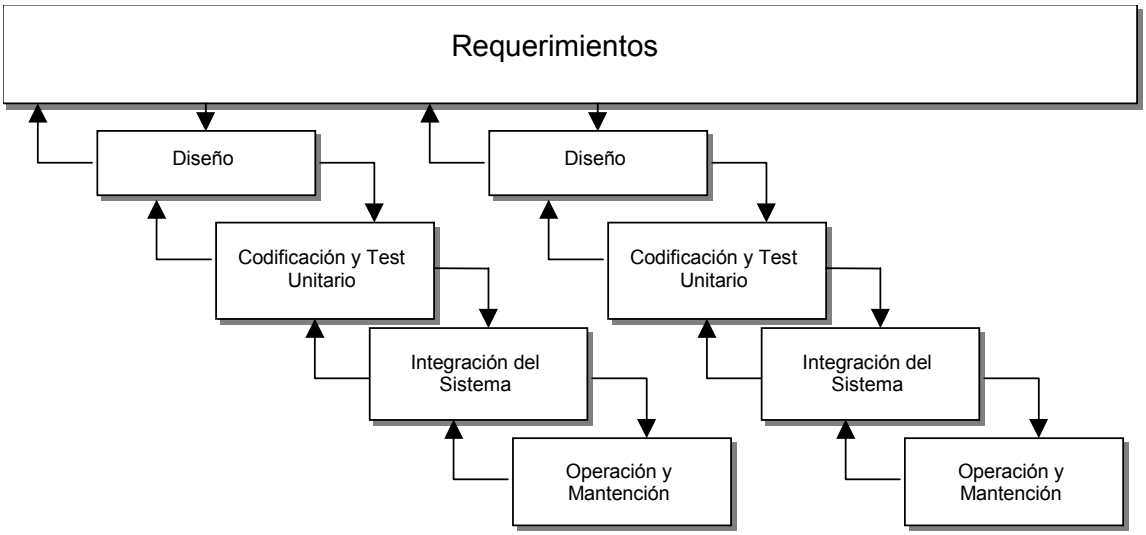
Note que el desarrollo incremental es 100% compatible con el modelo cascada. El desarrollo incremental no demanda una forma específica de observar el desarrollo de algún otro incremento. Así, el modelo cascada puede ser usado para administrar cada esfuerzo de desarrollo, como se muestra en la figura.

El modelo de desarrollo incremental provee algunos beneficios significativos para los proyectos:

- Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
- Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
- Si un error importante es realizado, sólo la última iteración necesita ser descartada.
- Reduciendo el tiempo de desarrollo de un sistema (en este caso en incremento del sistema) decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo.
- Si un error importante es realizado, el incremento previo puede ser usado.
- Los errores de desarrollo realizados en un incremento, pueden ser arreglados antes del comienzo del próximo incremento.



Modelo de Desarrollo Incremental.



Modelo de Desarrollo Incremental con desarrollo en cascada de los incrementos.

3.1.2.3 Modelo De Desarrollo Evolutivo.

Como el modelo de desarrollo incremental, el modelo de desarrollo evolutivo (algunas veces denominado como prototipado evolutivo) construye una serie de grandes versiones sucesivas de un producto. Sin embargo, mientras que la aproximación incremental presupone que el conjunto completo de requerimientos es conocido al comenzar, el modelo evolutivo asume que los requerimientos no son completamente conocidos al inicio del proyecto.

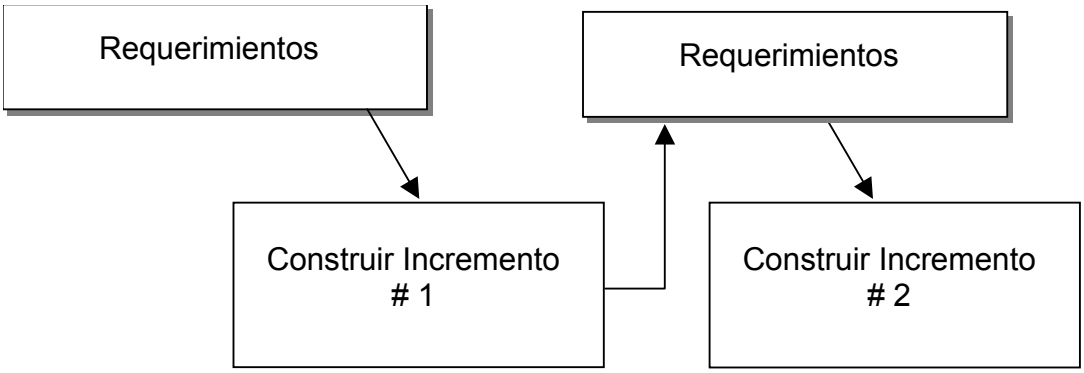
En el modelo evolutivo, los requerimientos son cuidadosamente examinados, y sólo esos que son bien comprendidos son seleccionados para el primer incremento. Los desarrolladores construyen una implementación parcial del sistema que recibe sólo estos requerimientos.

El sistema es entonces desarrollado, los usuarios lo usan, y proveen retroalimentación a los desarrolladores. Basada en esta retroalimentación, la especificación de requerimientos es actualizada, y una segunda versión del producto es desarrollada y desplegada. El proceso se repite indefinidamente.

Note que el desarrollo evolutivo es 100% compatible con el modelo cascada. El desarrollo evolutivo no demanda una forma específica de observar el desarrollo de algún incremento. Así, el modelo cascada puede ser usado para administrar cada esfuerzo de desarrollo. Obviamente, el desarrollo incremental y evolutivo puede ser combinado también.

Todo lo que uno tiene que hacer es construir un subconjunto de requerimientos conocidos (incremental), y comprender al principio que muchos nuevos requerimientos es probable que aparezcan cuando el sistema sea desplegado o desarrollado.

El desarrollo de software en forma evolutiva requiere un especial cuidado en la manipulación de documentos, programas, datos de test, etc. desarrollados para distintas versiones del software. Cada paso debe ser registrado, la documentación debe ser recuperada con facilidad, los cambios deben ser efectuados de una manera controlada.



Modelo Evolutivo.

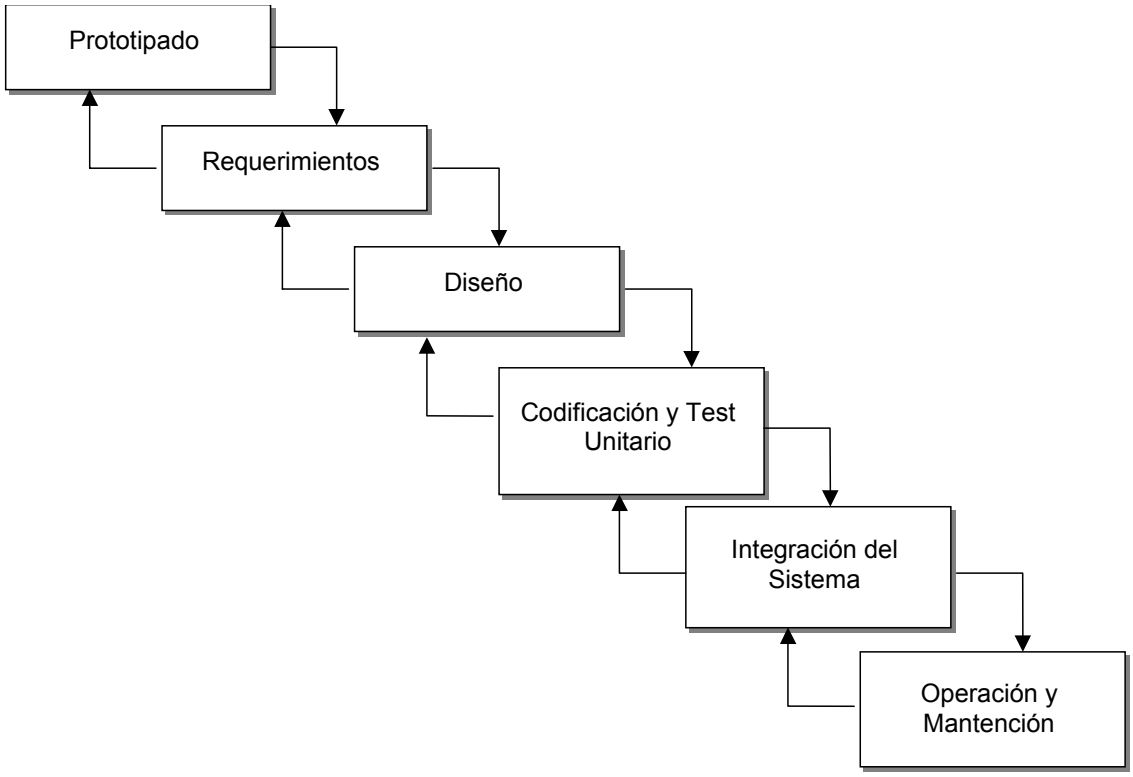
3.1.2.4 Modelo de Prototipado de Requerimientos.

El prototipado de requerimientos es la creación de una implementación parcial de un sistema, para el propósito explícito de aprender sobre los requerimientos del sistema. Un prototipo es construido de una manera rápida tal como sea posible. Esto es dado a los usuarios, clientes o representantes de ellos, posibilitando que ellos experimenten con el prototipo. Estos individuos luego proveen la retroalimentación sobre lo que a ellos les gustó y no les gustó acerca del prototipo proporcionado, quienes capturan en la documentación actual de la especificación de requerimientos la información entregada por los usuarios para el desarrollo del sistema real. El prototipado puede ser usado como parte de la fase de requerimientos (determinar requerimientos) o justo antes de la fase de requerimientos (como predecesor de requerimientos). En otro caso, el prototipado puede servir su papel inmediatamente antes de algún o todo el desarrollo incremental en modelos incremental o evolutivo.

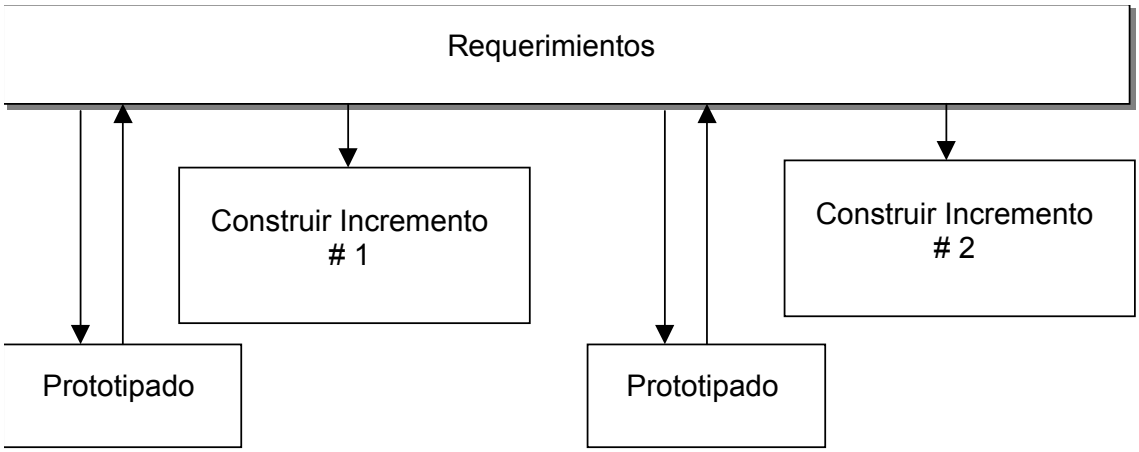
El Prototipado ha sido usado frecuentemente en los 90, porque la especificación de requerimientos para sistemas complejos tienden a ser relativamente dificultoso de cursar. Muchos usuarios y clientes encuentran que es mucho más fácil proveer retroalimentación convenientemente basado en la manipulación, desde un prototipo, en vez de leer una especificación de requerimientos potencialmente ambigua y extensa.

Diferente del modelo evolutivo donde los requerimientos mejor entendidos están incorporados, un prototipo generalmente se construye con los requerimientos entendidos más pobremente.

En caso que ustedes construyan requerimientos bien entendidos, el cliente podría responder con “sí, así es”, y nada podría ser aprendido de la experiencia.



Prototipado de Requerimientos basado en el modelo Cascada.



Prototipado de Requerimientos basado en el modelo de desarrollo incremental.

3.1.2.5 Modelo Espiral.

El modelo espiral de los procesos software es un modelo del ciclo de *meta-vida*. En este modelo, el esfuerzo de desarrollo es iterativo. Tan pronto como uno completa un esfuerzo de desarrollo, otro comienza. Además, en cada desarrollo ejecutado, puedes seguir estos cuatros pasos:

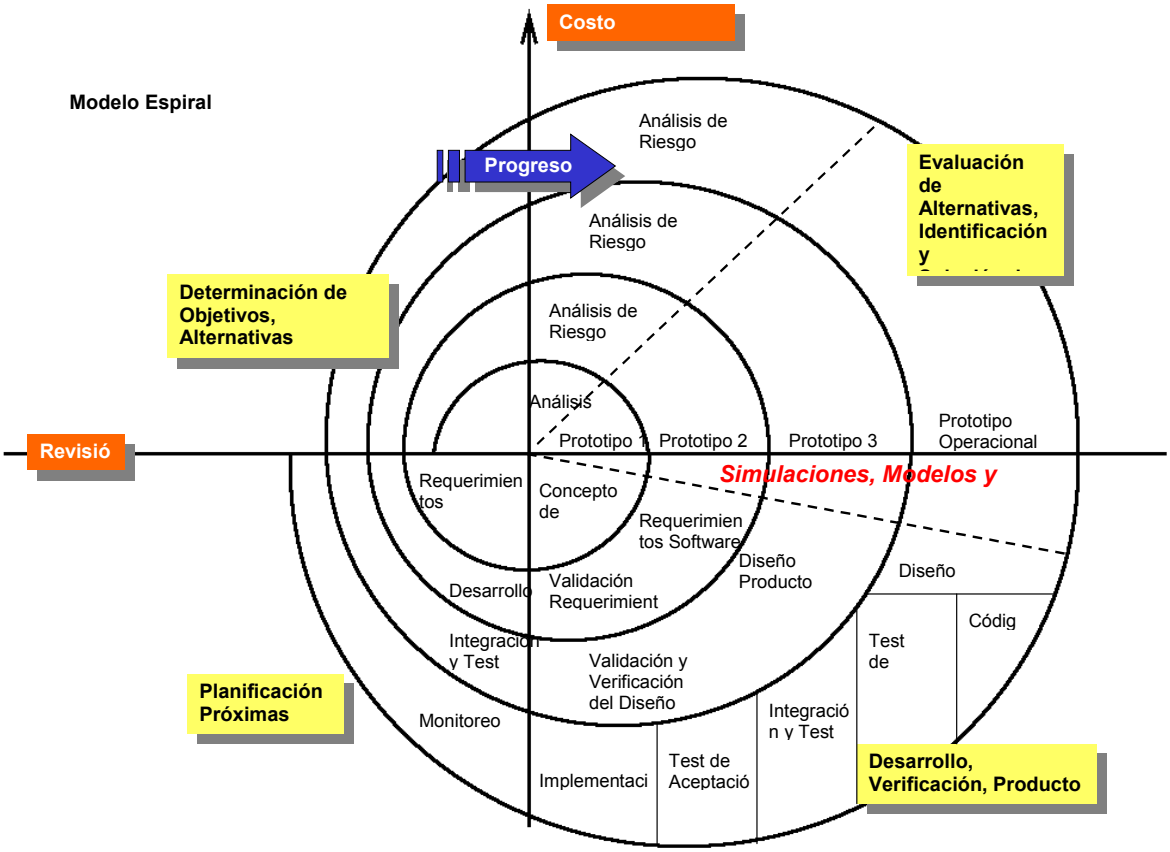
1. Determinar qué se quiere lograr.
2. Determinar las rutas alternativas que se pueden tomar para lograr estas metas. Por cada una, analizar los riesgos y resultados finales, y seleccionar la mejor.
3. Seguir la alternativa seleccionada en el paso 2.
4. Establecer qué se tiene terminado.

La dimensión radial en la figura refleja costos acumulativos incurridos en el proyecto.
Observemos un escenario particular. Digamos que en este proyecto, nosotros viajaremos a resolver un conjunto particular de problemas del cliente. Durante el primer viaje alrededor de la espiral, analizamos la situación y determinamos que los mayores riesgos son la interfaz del usuario. Después de un cuidadoso análisis de las formas alternativas de direccionar esto (por ejemplo, construir un sistema y esperar lo mejor, escribir una especificación de requerimientos y esperar que el cliente lo entienda, y construir un prototipo), determinamos que el mejor curso de acción es construir un prototipo.
Lo realizamos. Luego proveemos el prototipo al cliente quien nos provee con retroalimentación útil. Ahora, comenzamos el segundo viaje alrededor de la espiral. Este tiempo decidimos que el mayor riesgo es ese miedo a que muchos nuevos requerimientos comiencen a aparecer sólo después de que el sistema sea desplegado. Analicemos las rutas alternativas, y decidimos que la mejor aproximación es construir un incremento del sistema que satisfaga sólo los requerimientos mejor entendidos. Hagámoslo ya. Después del despliegue, el cliente nos provee de retroalimentación que dirá si estamos correctos con esos requerimientos, pero 50 nuevos requerimientos ahora se originarán en las cabezas de los clientes. Y el tercer viaje alrededor de la espiral comienza.

El modelo espiral captura algunos principios básicos:

- Decidir qué problema se quiere resolver antes de viajar a resolverlo.
- Examinar tus múltiples alternativas de acción y elegir una de las más convenientes.
- Evaluar qué tienes hecho y qué tienes que haber aprendido después de hacer algo.
- No ser tan ingenuo para pensar que el sistema que estás construyendo será “EL” sistema que el cliente necesita, y
- Conocer (comprender) los niveles de riesgo, que tendrás que tolerar.

El modelo espiral no es una alternativa del modelo cascada, ellos son completamente compatible.



3.1.2.6 Modelo Concurrente.

Como el modelo espiral, el modelo concurrente provee una meta-descripción del proceso software. Mientras que la contribución primaria del modelo espiral es en realidad que esas actividades del software ocurran repetidamente, la contribución del modelo concurrente es su capacidad de describir las múltiples actividades del software ocurriendo simultáneamente.

Esto no sorprende a nadie que ha estado involucrado con las diversas actividades que ocurren en algún tiempo del proceso de desarrollo de software. Discutamos un poco tales casos:

Los requerimientos son usualmente “líneas de base”, cuando una mayoría de los requerimientos comienzan a ser bien entendidos, en este tiempo se dedica un esfuerzo considerable al diseño. Sin embargo, una vez que comienza el diseño, cambios a los requerimientos son comunes y frecuentes (después de todo, los problemas reales cambian, y nuestro entendimiento de los problemas desarrollados también). Es desaconsejado detener el diseño en este camino cuando los requerimientos cambian; en su lugar, existe una necesidad de modificar y rehacer líneas de base de los requerimientos mientras progresa el diseño. Por

supuesto, dependiendo del impacto de los cambios de los requerimientos el diseño puede no ser afectado, medianamente afectado o se requerirá comenzar todo de nuevo.

Durante el diseño de arquitectura, es posible que algunos componentes comiencen a ser bien definidos antes que la arquitectura completa sea estabilizada. En tales casos, puede ser posible comenzar el diseño detallado en esos componentes estables. Similarmente, durante el diseño detallado, puede ser posible proceder con la codificación y quizás regular testeando en forma unitaria o realizando testeo de integración previo a llevar a cabo el diseño detallado de todos los componentes.

En algunos proyectos, múltiples etapas de un producto se han desarrollado concurrentemente. Por ejemplo, no es inusual estar haciendo mantención de la etapa 1 de un producto, y al mismo tiempo estar haciendo mantención sobre un componente 2, mientras que se está haciendo codificación sobre un componente 3, mientras se realiza diseño sobre una etapa 4, y especificación de requisitos sobre un componente 5.

En todos estos casos, diversas actividades están ocurriendo simultáneamente. Eligiendo seguir un proyecto usando técnicas de modelación concurrente, se posibilita el conocimiento del estado verdadero en el que se encuentra el proyecto.

3.1.3 Seleccionando Modelos de Ciclo de Vida.

Los modelos presentados, suministran una guía con el fin de ordenar las diversas actividades técnicas en el proyecto de desarrollo de software e intentan suministrar un marco para la administración en el desarrollo y el mantenimiento. Aunque todos ellos son compatibles unos con otros, un proyecto puede decidir cuáles enfoques son más útiles en situaciones especiales.

Criterios a considerar:

- Madurez de la aplicación (relacionado a la probabilidad que muchos requerimientos comenzarán a conocerse sólo después del uso del sistema).
- Complejidad del problema y de la solución.
- Frecuencias y magnitudes esperadas de los cambios de requerimientos.
- Financiamiento disponible, y su perfil como una función del tiempo.
- Acceso de los desarrolladores a los usuarios.
- Certeza de requerimientos conocidos.

Otros que pueden incluirse:

- Tolerancia al riesgo.

- Planes y presupuestos críticos
- Grado de lentitud de construcción dentro de los planes y presupuestos.

Considerando la importancia de la planificación se recomienda realizar el desarrollo de un proyecto de software bajo el modelo espiral insertando en él, cualquier otro modelo que se requiera dependiendo de las necesidades que se presenten. Este modelo permite realizar una planificación del proceso de desarrollo del software considerando los riesgos asociados en cada etapa identificada. El identificar los riesgos en proyectos, evaluar su impacto, monitorear y controlar el avance del desarrollo del proyecto, permite al administrador aumentar las posibilidades de éxito de un proyecto o, minimizar las posibilidades de fracaso de éste.

Uno de los factores que más influyen en el proceso de desarrollo de software y que prácticamente acompaña a toda aplicación es el hecho de que en su mayoría, no hay forma de tener todos los requerimientos corregidos antes del desarrollo del software. Muchas veces los requerimientos emergen a medida que la aplicación o partes de ella están disponible para experimentación práctica. En todos los casos, el trabajo comienza con la determinación de objetivos, alternativas y restricciones, paso que a veces se llama recolección preliminar de requisitos.

El prototipado es ampliamente recomendado para realizar la especificación de requerimientos, se debe notar que la idea del prototipado es capturar por retroalimentación los objetivos, necesidades y expectativas del cliente por lo cual no se debe caer en una utilización de estos prototipos como partes finales del sistema, ya que en su desarrollo generalmente no se consideran aspectos de calidad, ni otros asociados con facilitar la etapa de mantención del sistema. El prototipo trata de minimizar los cambios en los requerimientos, mientras que el diseño modular (incremental, en funcionalidades) trata de minimizar el impacto de los cambios en los requerimientos.

El cambio es una propiedad intrínseca del software. Hoy en día el software debe poseer un enfoque evolutivo, un sistema debe evolucionar para acomodar la naturaleza evolutiva de los grandes sistemas. El software cambia constantemente, debido a la necesidad de reparar el software (eliminando errores no detectados anteriormente) como a la necesidad de apoyar la evolución de los sistemas a medida que aparecen nuevos requerimientos o cambian los antiguos.

Por lo cual es importante enfatizar que no tiene sentido entonces que un proyecto tome estrictamente una decisión concerniente con cual modelo se adherirá. Los modelos de ciclo de vida presentados, son complementarios en vez de excluyentes.

En muchos casos, los paradigmas pueden y deben combinarse de forma que puedan utilizarse las ventajas de cada uno en un único proyecto. El paradigma del modelo en espiral lo hace directamente, combinando la creación de prototipos y algunos elementos del ciclo de vida clásico, en un enfoque evolutivo para la ingeniería de software.

No hay necesidad por tanto de ser dogmático en la elección de los paradigmas para la ingeniería de software: la naturaleza de la aplicación debe dictar el método a elegir.

4. Estimación de Costo y Plazos para la Planificación.

Una de las actividades cruciales del proceso de gestión es la planificación, la cual se basa en una buena estimación del esfuerzo requerido para realizar el proyecto, duración cronológica del proyecto y el costo (en miles de pesos o dólares).

4.1 Técnicas comúnmente utilizadas

La técnica más utilizada para realizar estimaciones de costos y plazos es la que denomino "juicio experto", donde el administrador del proyecto recurre a alguien que haya desarrollado aplicaciones similares para que realice una estimación de los recursos y tiempo a necesitar para el desarrollo.
Otra técnica muy natural es utilizar descomposición. Esto es, dividir el problema en partes más pequeñas y estimar cada una por separado, utilizando un juicio experto o algún método más formal (como estimar sobre la base del tamaño en una métrica formal).
Además, se suele realizar una estimación optimista (EO), otra más probable (EMP) y una pesimista (EP), y asignarle una probabilidad a cada una, obteniendo así nuestra estimación mediante:

$$E=EO *Po + EMP*Pmp + EP*Pp$$

Donde Po es la probabilidad asignada a la estimación optimista, Pmp la asignada a la más probable y Pp la asignada a la pesimista.
Así, por ejemplo, dado un proyecto software X, estimamos que lo mínimo que nos podríamos demorar son 5 meses con probabilidad de un 10%, y lo máximo de 12 meses con probabilidad de un 30%, y seguramente nos tardaremos 10 meses. Por otro lado, lo mínimo que nos costará es US$2000 con probabilidad 15%, lo máximo US$6200 con probabilidad 20%, y lo más probable es que el costo sea de US$4000. Con estos datos podemos obtener dos estimaciones:
Tiempo = 5 *0.1 + 12 * 0.3 + 10 * 0.6 =0.5 + 3.6 +6 = 10.1 meses
Costo = 2000 * 0.15 + 6200*0.2 + 4000 * 0.65 = 300 + 1240 + 2600 = US$4140

4.2 Métricas

La medición es muy común en el mundo de la ingeniería. Se miden potencias, consumos, pesos, fuerzas, voltajes, niveles de ruido, etc. Desgraciadamente, la medición no es una práctica común en el mundo de la ingeniería de software.

Razones que justifican la medición del software son

(a) para indicar la calidad del producto,

(b) para evaluar la productividad de la gente que desarrolla el producto,

(c) para evaluar los beneficios (en términos de productividad y calidad) derivados del uso de nuevos métodos y herramientas de ingeniería de software,

(d) para establecer una línea base para la estimación y

(e) para ayudar a justificar el uso de nuevas herramientas o formación adicional.

Las mediciones del mundo físico pueden englobarse en dos categorías: medidas directas (el largo de un tornillo) y medidas indirectas (la calidad de los tornillos producidos). Las métricas del software pueden ser catalogadas en forma análoga.

Entre las medidas directas del proceso de ingeniería de software se encuentra el costo y el esfuerzo aplicado. Entre las medidas directas del producto se encuentran las líneas de código producidas, velocidad de ejecución y los defectos observados en un período de tiempo. Entre las medidas indirectas se encuentran la calidad, funcionalidad, eficiencia, facilidad de mantenimiento, etc.

Las métricas de software se pueden clasificar como métricas orientadas a la función o métricas orientadas al tamaño. También se pueden clasificar según la información que entregan: métricas de productividad, las que se centran en el rendimiento del proceso de ingeniería de software, métricas de calidad, proporcionan una indicación de cómo se ajusta el software a los requisitos explícitos e implícitos del cliente y las métricas técnicas, que se centran más en el software que en el proceso a través del cuál se ha desarrollado (por ejemplo grado de modularidad o grado de complejidad lógica).

Para resumir, podemos decir que una métrica de software es una función cuyas entradas son datos del software (o el proceso del software) y cuya salida es un valor numérico único, que puede ser interpretado como el grado en que el software (o el proceso del software) posee un atributo dado.

4.2.1 Métricas orientadas al tamaño

Las métricas del software orientadas al tamaño son medidas directas del software y el proceso de software.

Si en una organización se realizan registros como en la tabla siguiente, se puede obtener alguna información interesante.

Supongamos que contamos con la siguiente información:

Proyecto	KLOC	Esfuerzo (personas-mes)	Documentación (# páginas)	# personas	errores	Costo total (US$)
Proy1	12,1	26	370	3	29	180
Proy2	27,8	60	1200	5	86	450
Proy3	20	43	1050	6	64	320

Con los datos contenidos en esta tabla, se pueden desarrollar un conjunto de métricas de productividad y calidad orientadas al tamaño.

Productividad = KLOC/Esfuerzo

Calidad=Errores/KLOC

Coste= Costo/KLOC

Tasa Documentación Documentación/KLOC

Para efectos de la estimación del esfuerzo de desarrollo como base de la planificación, las métricas orientadas al tamaño han sido utilizadas en varias propuestas. La mayoría de ellas son lo que se llaman modelos de estimación, que en este caso son funciones de las líneas de código o de las miles de líneas de código que tendrá el software a desarrollar.

Los principales problemas de utilizar líneas de código como métrica son la falta de una definición universal de línea de código, su dependencia con el lenguaje de desarrollo y la dificultad de estimar en fases tempranas del desarrollo la cantidad de líneas que tendrá una aplicación para efectos de estimación del esfuerzo.

4.2.2 Métricas orientadas a la función.

Las métricas de software orientadas a la función son medidas indirectas del software y el proceso por el cual se desarrolla. En lugar de calcular las líneas de código, éstas se centran en la funcionalidad o utilidad del software.
La primera propuesta de este tipo fue los puntos de función, realizada por Albrecht, métrica que hasta hoy es muy utilizada. De los puntos de función de Albrecht han salido algunas variacionestales como los puntos de característica y los puntos de función para estimación temprana.

4.2.2.1 Puntos de Función

El análisis por puntos de función es un método para cuantificar el tamaño y la complejidad de un sistema software en término de las funciones de usuario que este desarrolla (o desarrollará). Esto hace que la medida sea independiente del lenguaje o herramienta utilizada en el desarrollo del proyecto.
El análisis por puntos de función está diseñado para medir aplicaciones de negocios; no es apropiado para otro tipo de aplicaciones como aplicaciones técnicas o científicas. Esas aplicaciones generalmente median con algoritmos complejos que el método de puntos de función no está diseñado para manejar.
El enfoque de puntos de función tiene características que sobrellevan los principales problemas de utilizar líneas de código como métrica del tamaño del software. Primero, los puntos de función son independientes del lenguaje, herramientas o metodologías utilizadas en la implementación; por ejemplo, no tienen que considerar lenguajes de programación, sistemas de administración de bases de datos, hardware, o cualquier otra tecnología de procesamiento de datos. Segundo, los puntos de función pueden ser estimados a partir de la especificación de requisitos o especificaciones de diseño, haciendo posible de este modo la estimación del esfuerzo de desarrollo en etapas tempranas del mismo. Como los puntos de función están íntimamente relacionados con la declaración de requisitos, cualquier modificación a ésta, puede ser reflejada sin mayor dificultad en una reestimación. Tercero, como los puntos de función están basados en una visión externa del usuario del sistema, los usuarios no técnicos del software poseen un mejor entendimiento de lo que los puntos de función están midiendo. El método resuelve muchas de las inconsistencias que aparecen cuando se utiliza líneas de código como métrica del tamaño del software.

En resumen, los puntos de función aparecen con ventajas sustanciales por sobre las líneas de código, para fines de estimación temprana del tamaño del software. Además es una medida ampliamente utilizada, y con éxito, en muchas organizaciones que desarrollan software en forma masiva.

4.2.2.2 Puntos de Característica

Debido a que el análisis por Puntos de Función fue diseñado para software de negocios y no es fácil de generalizar a aplicaciones científicas, de tiempo real y otras, Caper Jones propuso ampliaciones al método que denominó Puntos de Característica, que da cabida a aplicaciones cuya complejidad algorítmica es alta.

Este método considera los mismos elementos que considera Albrecht en su método de análisis por puntos de función, sólo que añade la variable número de algoritmos y elimina los niveles de complejidad.

4.2.2.3 Hoja resumen de puntos de función.

Aplicación:

Appl ID:

Preparado por: (/ /)

Revisado por: (/ /)

Notas:

.Cuentas de Función.

ID Tipo	Descripción	Complejidad			Total
		Simple	Promedio	Compleja	
IT	Entradas Externas	___*3=___	___*4=___	___*6=___	
OT	Salidas Externas	___*4=___	___*5=___	___*7=___	
FT	Archivos Lógicos Internos	___*7=___	___*10=___	___*15=___	
ET	Archivo de Interfaz Externos	___*3=___	___*7=___	___*10=___	
QT	Consultas Externas	___*3=___	___*4=___	___*6=___	

FC	Total de Puntos de Función sin ajustar (brutos)	

.Complejidad de proceso.

	Característica	GI	ID	Característica	GI
C1	Comunicación de Datos	___	C8	Actualización en Línea	___
C2	Funciones Distribuidas	___	C9	Complejidad de procesos	___
C3	Desempeño	___	C10	Reusabilidad del Software	___
C4	Esfuerzo para la Configuración	___	C11	Facilidad de instalación	___
C5	Tasa de Transacciones	___	C12	Facilidad de operación	___
C6	Entrada de Datos en Línea	___	C13	Múltiples lugares	___
C7	Eficiencia del Usuario	___	C14	Facilidad de cambios	___
PC	Total Grado de Influencia				

. Valores de GI.

0: No está presente o no influencia
1: Influencia insignificante
2: Influencia moderada
3: Influencia promedio
4: Influencia significativa
5: Influencia fuerte, en todo

PCA: (Processing Complexity Adjustement) = 0.65 + (0.01*PC) = ___
FP : Function Point Measure = FC * PCA = ___

4.3 Estimación del Esfuerzo de Desarrollo

Una vez que conocemos como obtener la métrica por puntos de función, estamos interesados en su utilización como apoyo a la etapa de planificación. Para ello deseamos utilizar esta medida para estimar costos y plazos.

4.3.1 El tamaño de la aplicación.

El manejador de costo principal en el software es el tamaño. El tamaño de la aplicación representa su complejidad, en el sentido de que involucra directamente aquellas características que inciden en el esfuerzo de desarrollo.

Para obtener medidas consistentes del tamaño del software, la métrica a utilizar debe ser independiente de la tecnología de desarrollo, esto es, metodologías, hardware, herramientas de programación.

Nos interesa medir el software para poder compararlo con otros, conocer cuanto produce una organización o utilizarlo para obtener otras medidas de interés, como productividad, tasa de errores, estimación del costo, estimación de plazos.

El conocimiento del tamaño de la aplicación es vital para poder hacer un dimensionamiento del desarrollo, además el tener una medida del tamaño como puntos de función tiene la gran ventaja de ser comparable, tanto dentro de la misma organización, como con otras.

4.3.2 Especificación de Requisitos.

Para efectos de estimación de esfuerzo, deseamos poder obtener el tamaño de la aplicación a desarrollar en PF con sólo la especificación de requisitos del software.

Esta especificación debe ser tal que podamos extraer de ella toda la información necesaria para realizar este cálculo.

Propongo seguir una pauta como la que sigue, que tiene una clara diferenciación entre requisitos funcionales y no funcionales.

Pauta Documento de Requisitos de un Software.

I. Introducción.

II. Modelo del Sistema.

III. Evolución del Sistema.

IV. Requisitos Funcionales

V. Requisitos no Funcionales

VI. Anexos.

Observemos que de los requisitos funcionales se obtienen las Cuentas de Función, ya que es en este capítulo donde aparecen las Entradas, Salidas, Consultas y Archivos Lógicos. Se recomienda cohesionar los requisitos funcionales según algún criterio consistente, por ejemplo por conjunto de información. Los Archivos Lógicos Internos se pueden obtener sin aproximaciones a partir de un modelo conceptual de datos, pero personas con experiencia podrán visualizar desde los requisitos funcionales conjuntos cohesionados de datos que con una probabilidad muy alta pasarán a formar parte de un archivo lógico interno.

De los requisitos no funcionales, obtenemos las características generales de la aplicación, que deben complementarse con una visión global y sistémica del software a desarrollar, para evaluar correctamente el grado de influencia de cada característica en la aplicación.
Por último, cabe destacar que aunque el método de análisis por puntos de función es bueno, no garantiza la independencia de las mediciones de las personas que las realizan, lo que derivará en estimaciones diferentes. Existen ciertos grados de subjetividad que deben ser reducidos estandarizando la semántica de algunos conceptos y procedimientos incluidos en el método de análisis por puntos de función. Esta tarea debe realizarse al interior de cada organización.

4.3.3 Método de apoyo a las estimaciones tempranas.

Se propone un método que posibilita realizar estimaciones tempranas disminuyendo el costo de obtener registros de proyectos terminados en la organización. Se posibilita la disponibilidad de estimaciones tempranas apenas se comienza a utilizar el método.

1. Para cada proyecto de desarrollo, clasificarlo de acuerdo a sus características (sistema de información administrativo, científico) y a la tecnología de desarrollo (si es relevante).
2. Para cada proyecto de desarrollo de software que haya finalizado la etapa de especificación de requisitos, registrar su tamaño en PFET.
3. Estimar el esfuerzo de desarrollo utilizando algún modelo disponible para su tipo (en su defecto, utilizar algún modelo de pre existente).
4. Corregir dicha estimación por el Factor de Apoyo Tecnológico, en caso de estar utilizándose tecnologías nuevas, que no posean un modelo empírico asociado.
5. Para cada proyecto de desarrollo de software que haya finalizado la etapa de diseño externo, registrar su tamaño en PF y PC (si posee complejidad algorítmica).
6. Estimar el esfuerzo de desarrollo utilizando algún modelo disponible para su tipo (en su defecto, utilizar algún modelo pre existente)
7. Corregir dicha estimación por el Factor de Apoyo Tecnológico, en caso de estar utilizándose tecnologías nuevas, que no posean un modelo empírico asociado.
8. Para cada proyecto terminado, registrar el esfuerzo de desarrollo en horas hombre.
9. Determinar un nuevo modelo empírico de estimación, agregando la nueva información al conjunto de datos de proyectos anteriores.
10. Determinar un modelo empírico de estimación asociado a proyectos desarrollados con tecnologías que mejoran la productividad, si corresponde. De este modo, se evitará la

realización de correcciones por factor de apoyo tecnológico a los futuros proyectos de características similares.

Se propone la generación de modelos asociados a tipos de proyectos, esto es, diferenciar proyectos por criterios tecnológicos o humanos (desarrollados con herramientas distintas o por equipos de desarrollo diferentes).

Para sobrellevar el problema de la no-existencia de modelos, se propone la "sintonización" de un modelo externo documentado, para ir adaptándolo incrementalmente a la organización, a medida que se van terminando proyectos de cada tipo.

Para solucionar el problema de la constante migración hacia herramientas que mejoran la productividad, se ha incluido una componente de "Incremento de la productividad por Apoyo Tecnológico", que constituye un factor asignado por el administrador del proyecto, que corregirá las estimaciones realizadas utilizando modelos de estimación basados en proyectos desarrollados con tecnologías anteriores. En este caso, se debe crear un nuevo tipo de proyectos (desarrollados con tecnología nueva) y utilizar el factor de corrección (para estimaciones hechas con el modelo determinado a partir de proyectos desarrollados con tecnología antigua) hasta que se tenga información suficiente como para generar un modelo independiente.

4.4 Consideraciones al Método.

No existen soluciones universales en esto de la estimación del esfuerzo de desarrollo de software. Lo propuesto corresponde a una solución general que debe ser adaptada a las realidades particulares de la organización donde se implantará.

La solución propuesta se basa en el análisis por puntos de función, principalmente por las ventajas que esta métrica tiene. De todos modos, al ser este método propuesto hace más de una década y basado en proyectos realizados con herramientas de aquellos años, es necesario realizar una actualización de los conceptos presentes en el mismo.

Es así como resulta necesario definir claramente la semántica de cada uno de los conceptos presentes en las cuentas de los puntos de función. De este análisis resultará el significado de "una entrada externa" para esta organización, por ejemplo. Así como también es necesario adaptar el significado de las características generales de la aplicación.

Lo que se debe realizar es replantear la semántica de estas características generales y de las ponderaciones de los parámetros relevantes. Para esto, es necesario realizar un estudio acucioso de las aplicaciones actualmente realizadas, poniendo énfasis en un sólo tipo, ya que no existen soluciones universales.

El proceso de estimación del esfuerzo no es gratuito. Es necesario esfuerzo adicional en cada organización. Esto se debe tener muy presente a la hora de implementar un método probado en alguna organización externa, lo que no significa que otorgará buenos resultados en cualquier otra.

5. Organización del Proyecto.

5.1 Introducción.

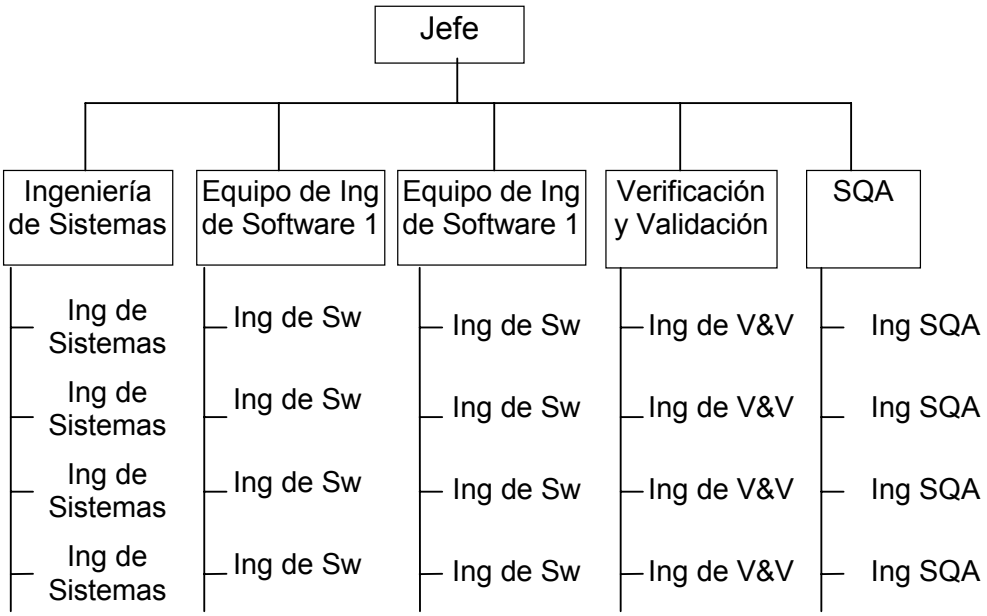
El proceso de la dirección denota el influjo interpersonal mediante el cual los gerentes o directores se comunican con sus subalternos para la ejecución de un trabajo. La justificación de este proceso radica en que se facilita el trabajo cuando existe una buena comunicación acerca de los problemas técnicos, de coordinación y de motivación.
Para poder realizar la dirección de una forma efectiva, es necesario la utilización de una estructura organizacional que nos permita trabajar en forma óptima con los recursos disponibles.
Lamentablemente, una única estructura organizacional que se adapte perfectamente para dirigir todos los tipos de proyectos no existe, aunque, sin embargo, existe una variada gama de estructuras organizacionales alternativas que se adaptan de mejor o peor forma a los distintos tipos de proyectos, y que son presentadas a continuación.

5.2 Organización funcional.

Esta es la forma de organización más común, y se basa en una estructura jerárquica básica.
El organigrama de este tipo de organización tiene forma piramidal, encontrándose en el parte superior a los grados mas altos de dirección, con los grados de dirección medio y bajo distribuyéndose hacia abajo de la pirámide. Esta organización generalmente se descompone en varias unidades funcionales diferentes, como por ejemplo ingeniería, contabilidad, administración.
La razón de ser de esta estructura radica en las teorías de administración basadas en la especialización, las cuales consideran que es más fácil dirigir a especialistas si estos son agrupados según su especialidad y el jefe de departamento tiene conocimientos de esa disciplina en particular.
Este tipo de organización posee ventajas que descansan sobre su centralización de recursos similares, ya que, por ejemplo, brinda caminos de integración claros y bien definidos para especialistas jóvenes y además, la ayuda mutua es fácilmente encontrada gracias a la proximidad física.
Sin embargo, este tipo de organización posee debilidades. Por ejemplo, cuando se trabaja en proyectos múltiples, generalmente surgen problemas por el uso de los recursos. Otro ejemplo

a citar es el de que cada departamento funcional pone énfasis en su especialidad mas que en los objetivos del proyecto.

Pese a esto, la mayoría de las compañías usan este tipo de organización no sólo para sus proyectos, sino para todas sus actividades.



5.3 Organización de proyecto.

Este tipo de organización representa el opuesto a la organización funcional.

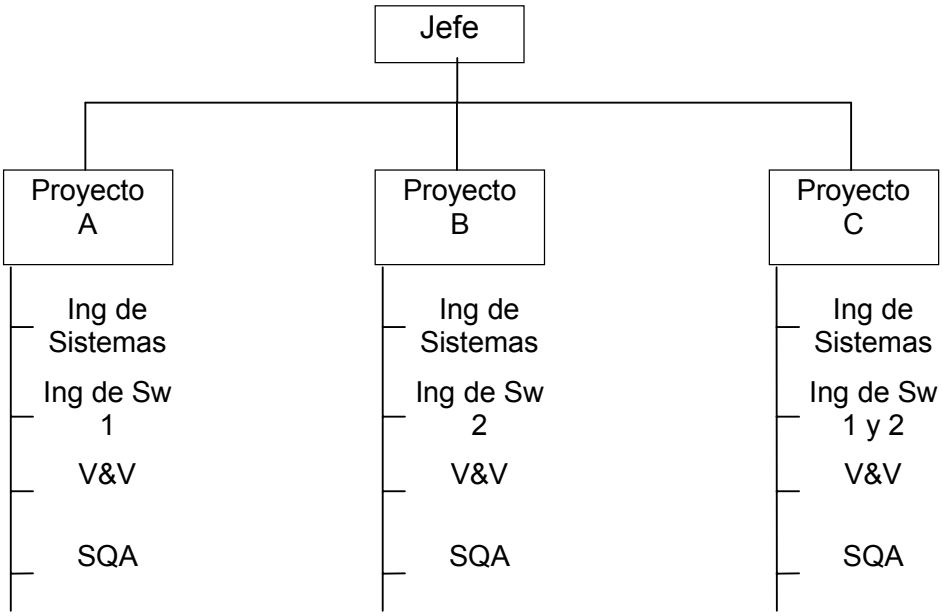
En una organización de proyecto, todos los recursos necesarios para realizar un proyecto son separados de sus unidades funcionales regulares y se reúnen como una unidad independiente de trabajo dirigida por un jefe de proyecto. Al jefe de proyecto se le concede una amplia autoridad sobre los recursos del proyecto y pueden adquirir nuevos recursos ya sea dentro o fuera de la organización. Todo el personal del proyecto esta bajo la autoridad del jefe de proyecto por el periodo que el proyecto dure.

Las ventajas de este tipo de organización vienen de su objetivo único y su unidad de comando. Se desarrolla un verdadero espíritu de equipo bajo el claro entendimiento de, y enfocado a, un objetivo único. La comunicación informal es efectiva en un equipo fuertemente enlazado, y el director de proyecto tiene todos los recursos bajo su disposición.

Sin embargo, la organización de proyecto no es la solución perfecta para todos los problemas de la dirección de proyectos, ya que por lo general, las instalaciones se duplican y los recursos

son utilizados de forma ineficiente. Otro problema serio radica en que se produce inestabilidad laboral luego de terminado el proyecto temporal.

La organización jerárquica está estructurada alrededor de entradas técnicas. La organización de proyecto estructura un propósito único construido alrededor de las salidas del proyecto. Estas son estructuras unidimensionales en un mundo multidimensional. El problema de cada una es conseguir un balance apropiado entre los objetivos a largo plazo de los departamentos funcionales en formar técnicos expertos y los objetivos a corto plazo del proyecto.



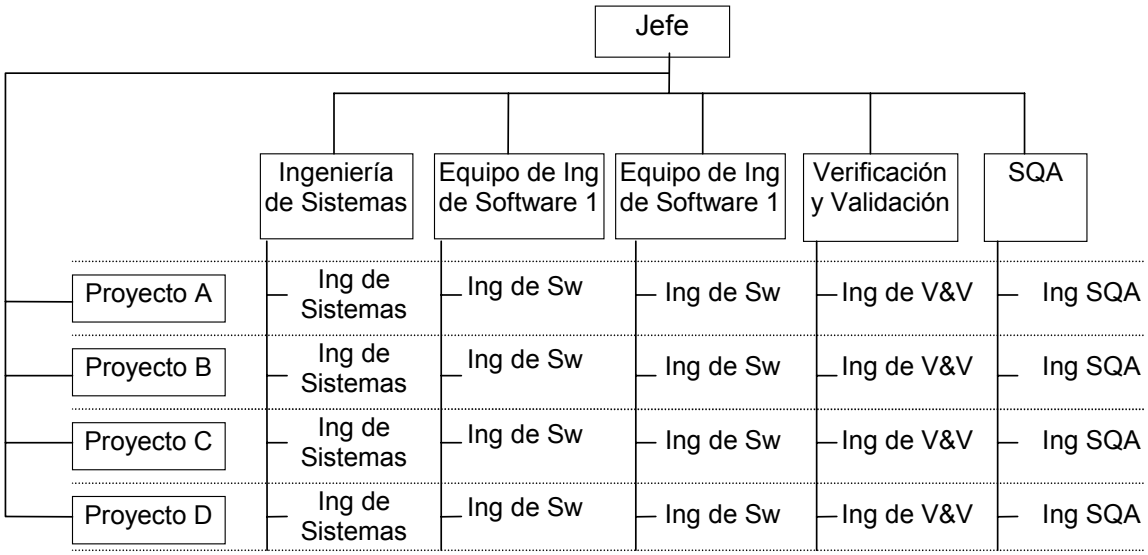
5.4 Organización de matriz.

La organización de matriz es una estructura multidimensional que trata de maximizar los puntos fuertes y minimizar los débiles tanto de la estructura funcional como la de proyectos. Combina la estructura jerárquica vertical standard con una estructura lateral u horizontal sobrepuesta de un coordinador de proyecto.

Los mayores beneficios de la estructura de matriz son el balance de objetivos, la coordinación a través de las líneas de los departamentos funcionales, y la visibilidad de los objetivos del proyecto a través de la oficina del coordinador de proyecto.

La mayor desventaja es que el hombre en el medio trabaja para dos jefes. Verticalmente, responde al jefe del departamento funcional. Horizontalmente, responde al coordinador de proyecto o jefe de proyecto. En una situación de conflicto puede quedar atrapado en el medio.

El director de proyecto a menudo siente que no posee la autoridad suficiente sobre los departamentos funcionales. Por otro lado, el jefe del departamento funcional a menudo siente que el coordinador de proyecto esta interviniendo en su territorio.



5.5 Criterio de elección de una estructura organizacional.

La experiencia ha demostrado que ningún acercamiento en particular es perfecto para todas las situaciones y proyectos. La moda actual en la literatura de dirección es el modelo de contingencia. Esta teoría plantea que la mejor solución es contingente a los factores claves en el ambiente en el cual la solución tendrá que operar.

Lo mismo es cierto para la elección de la estructura organizacional. La base para decidir está en los diversos factores claves en un proyecto especifico los cuales nos ayudarán a elegir la estructura organizacional correcta para las condiciones dadas, con una organización dada y en un ambiente particular.

Por ejemplo, una organización desarrollando muchos, pero pequeños, nuevos proyectos utilizando una tecnología standard tendería a funcionar de la mejor forma con una estructura funcional. Por otra parte, una compañía con un proyecto grande, extenso, complejo e importante optará por una estructura organizacional por proyecto. Una firma en el negocio farmacéutico con muchas tecnologías complejas probablemente trabajara con una estructura matricial.

Es también posible ocupar las tres estructuras en una misma compañía, en diferentes proyectos. También todas estas estructuras pueden ser utilizadas en un mismo proyecto en diferentes niveles –por ejemplo, una organización global de matriz con estructura funcional en ingeniería y una organización de proyecto dentro de otra sub área funcional.

5.6 Jefes de proyecto y diseño organizacional

No es posible decidir el diseño organizacional sin definir también quien será seleccionado como el jefe de proyecto y que tipo de diseño se quiere para los sistemas de reporte y planeamiento. Por ejemplo, una organización de proyecto exitosa requiere un jefe de proyecto con amplia capacidad para la dirección general. Debe combinar conocimiento técnico en la materia además de habilidades de dirección antes de poder dirigir a todo el personal del proyecto. No tiene sentido elegir una forma de organización de proyecto si tal jefe de proyecto no se encuentra.

Los sistemas de reporte y planeamiento en una organización de proyecto pueden ser bastante simples ya que el equipo trabaja con una gran proximidad. Lo opuesto es cierto en la dirección de proyectos a través de una organización funcional. Por esto, un sistema de planeamiento y reporte mas sofisticado es necesario en una organización funcional que en una organización de proyecto.

5.7 Métodos para mejorar las comunicaciones laterales en la estructura funcional.

Las organizaciones usualmente cambian a una organización de proyecto o una organización de matriz debido a que la estructura funcional regular falló en una serie de proyectos. Esto no es absolutamente necesario. Antes de abandonar la organización funcional, se debe analizar el problema real y ver si se puede tomar algún camino para una rápida reorganización. Algunos resultados de la reorganización pueden resultar favorables, pero otras consecuencias no intencionadas pero lógicas seguramente serán desfavorables.

Los métodos de comunicación lateral u horizontal necesitan ser construidos en los limites de los departamentos funcionales. Acercamientos alternativos para la comunicación lateral incluyen:

- Procedimientos como planes, presupuestos, inventarios y juntas de revisión.
- Contacto directo entre jefes de proyecto.
- Roles de relación informales.
- Equipos

Estos son mecanismos de integración de la organización matricial. Ayudan a romper las barreras que separan a las diferentes disciplinas, departamentos y ubicaciones geográficas.

5.8 Resumen.

No existe una estructura organizacional perfecta para manejar los distintos tipos de proyecto. La organización funcional, la de proyecto y la matricial tienen todas fuerzas y debilidades. La elección final deberá venir luego de considerar varios factores de la naturaleza de la tarea a realizar, las necesidades de la organización y el ambiente del proyecto.

La estructura funcional funcionara para muchos proyectos, en muchas organizaciones, especialmente si las comunicaciones laterales son reforzadas a través de mecanismos de integración.

Si se escoge una estructura de matriz, toda la organización deberá poner gran empeño para que esta funcione. En particular, el director de proyecto debe ser cuidadosamente escogido y entrenado. Sus habilidades interpersonales son más importantes que sus habilidades técnicas.

En muchas situaciones, una organización de proyecto puede parecer la solución más simple desde la perspectiva del director de proyecto. Sin embargo, los directores funcionales y la alta administración tal vez no la consideren como la mejor solución a largo plazo o la decisión más estratégica.

6. Selección de personas para conformar el equipo.

¿Cuál es el objetivo durante un proceso de selección de recursos humanos?

En esta área el principal objetivo es sumar adecuadamente personas a cargos vacantes en un equipo de desarrollo de sistemas Software, o bien la conformación de un equipo. En este momento hay que establecer cuáles son las condiciones deseables para dicho equipo. Y para poder establecer estas condiciones deseables se debe tener un adecuado conocimiento de como actúan los equipos de trabajo. Este conocimiento, necesario para poder realizar la selección de las personas adecuadas, es responsabilidad de los administradores de Software y futuros jefes de proyecto, de acuerdo con las políticas de la empresa.

De esta forma, con relación a los equipos de trabajo se debe tener en cuenta:

- Los equipos de trabajo están conformados por individuos, y estos individuos interactúan entre sí. El administrador debe entender las limitaciones humanas e intentar no llevar a cabo proyectos irrealizables.
- Los sistemas de Software son utilizados por personas. Si las limitaciones y capacidades de los desarrolladores no son tomadas en cuenta, difícilmente los sistemas serán bien aprovechados por los usuarios.

De esta forma la selección es importante teniendo en cuenta muchos factores de la personalidad de los postulantes. Sin embargo, no sólo se debe tener en cuenta esta vista al momento de escoger, sino también considerar experiencias anteriores, habilidades en ciertos aspectos del desarrollo, etc. Para tener éxito en la selección de los futuros integrantes de los equipos de trabajo, basarse sólo en la personalidad de los postulantes puede ser una mala decisión, debido a que:

- La personalidad es dinámica, no estática. Esta puede cambiar en el transcurso de una carrera profesional o por cambios en el ambiente de trabajo.
- Diferentes personalidades pueden resultar adecuadas a diferentes aspectos del desarrollo, tales como diseño, testing y otras.
- Los postulantes pueden hacer trampa al completar los tests de personalidad, llenándolos no con sus datos reales, sino con lo que piensan que podría ser la personalidad más adecuada al trabajo. Por esto la realización de una buena entrevista es vital para la selección.

6.1 La Entrevista.

La entrevista ayudará a encontrar las personas adecuadas a los cargos definidos en un equipo de desarrollo. Además de definir la personalidad de los postulantes, ayudará a establecer cuales son las capacidades del individuo frente al trabajo de equipo, adaptación a la organización y resto del equipo, capacidad de liderato y soportar presiones al acabarse los plazos de entrega o frente a problemas en el desarrollo, es decir, capacidad de aceptar cierta dosis de stress. Definir la personalidad del postulante permitirá definir cual es la orientación en la cual se puede clasificar. Esto permite poder establecer ubicaciones dentro del grupo y reconocer cuales personalidades entran en conflicto y como puede lograrse un trabajo armónico, principal preocupación del jefe de proyecto, ya que un trabajo armónico posibilita una mayor productividad y calidad del trabajo.

A grandes rasgos los individuos pueden clasificarse en tres tipos:

a) *Orientados al trabajo*: El individuo esta motivado por el trabajo mismo, inducido por un desafío intelectual.

b) *Auto-orientados*: Está motivado por un éxito personal, buscando alcanzar sus propias metas. Generalmente calzan en la parte administrativa de la organización.

c) *Orientados a la interacción*: Está motivado por la presencia y acción de sus colegas.

Se tiene que destacar que esta clasificación no es fija, y la personalidad puede variar en el tiempo de un tipo hacia otro.

La entrevista propiamente tal debe programarse con anterioridad, debiendo ser los entrevistadores lo mas objetivos posible, tratando de encontrar impresiones del candidato a lo largo de la entrevista y no formarse primeras impresiones en los primeros minutos de la entrevista. Estas son, por lo general, incorrectas. A continuación se describe algunos pasos que debe tener una entrevista. Es decir un modelo de la entrevista destinada a seleccionar al mejor candidato.

1. Preparando la entrevista:
    - Buscar un lugar adecuado y cómodo.
    - Evitar el uso del teléfono e interrupciones.
    - Mantener en mente el seguimiento de estas acciones.
    - Tener, por escrito, unas cuatro o cinco preguntas que ayuden a definir el potencial, habilidades, temperamento y otros atributos del candidato.

- Revisar los requerimientos de candidato y descripción del cargo que debe ser llenado (alto o bajo requerimiento de calidad).
- Revisar el curriculum del candidato.

2. Estructuración de la entrevista:
    - Usar varios entrevistadores en forma separada. Esto aumenta las visiones objetivas del candidato.
    - Realizar anotaciones cortas.
    - Acordar preguntar a todos los candidatos el mismo tipo de preguntas.
3. Conduciendo la entrevista:
    - Conocer que los primeros minutos de la entrevista son críticos en la formación de impresiones, tanto de parte del entrevistado como del entrevistador.
    - Recibir amistosamente al candidato.
    - Establecer una conversación y discusión sobre puntos en común. Esto puede determinarse a partir del curriculum del candidato.
    - Hablar en una forma relajada y afable, tratando de mantener una sonrisa en todo momento.
    - Tratar de llevar una conversación cara a cara, evitando mesas o escritorios entre el entrevistado y entrevistador.
    - Empezar conversando sobre aspectos generales para relajar al candidato. Después de esto, entrar a temas más específicos.
    - Realizar preguntas abiertas que permitan al candidato hablar a lo menos un 50% de lo conversado. Es decir, no obligar con una pregunta larga a responder solamente un 'sí' o 'no'.
    - Mientras sea posible, dar al candidato felicitaciones por su experiencia y habilidad basándose en el curriculum.
    - Usar pausas para permitir responder al candidato. Sin embargo, establecer claramente que las respuestas deben ser concisas y breves.
    - Terminar la entrevista explicando quien será el próximo entrevistador, o quien se comunicará con él y cuando. Encaminarlo hacia el siguiente entrevistador.
    - Agradecer al candidato por su tiempo e interés.
4. Después de la entrevista:
    - Escriba o almacene las respuestas a las preguntas y primeras impresiones del candidato mientras estén frescas en su mente.

- Organice una reunión con los otros entrevistadores a fin de comparar y discutir acerca de todos los candidatos
- Enviar comunicaciones citando a los candidatos seleccionados y notificaciones a los candidatos no seleccionados. Esta última debería provenir del administrador de proyectos, explicando el porqué de la decisión de no seleccionarlos. Es de sentido común hacer esta notificación, diciendo además que de todas formas pueden ser seleccionados en una futura necesidad de empleados.

6.2 Manejo del equipo.

Idealmente el equipo de desarrollo de un proyecto debe alcanzar, con la totalidad de sus miembros, el sistema final que se ha planeado. Es decir, todos los miembros de una organización o equipo deben mantenerse en ella en el tiempo, ya que de esta forma el equipo se va solidificando en su accionar.
En la realidad hay que considerar la rotación de los miembros de un equipo ya sea dentro de la misma organización, teniendo ascensos o cambios de proyectos; o bien alejándose de la organización. Esta rotación puede traer consecuencias tanto positivas como negativas al trabajo del equipo y a las relaciones dentro de estas. Dentro de las posibles consecuencias negativas pueden encontrarse:

- Interrupción en el desarrollo: Cuando se produce un alejamiento de un integrante clave del equipo de trabajo existe la posibilidad de un retraso en el desarrollo. Esta interrupción es mas o menos grave dependiendo de la fase en que se encuentre el desarrollo, siendo las primeras las mas críticas. Por ejemplo, en la etapa de análisis la perdida de un integrante puede producirse una grave perdida de información no documentada. En etapas posteriores puede producirse un retraso importante al acercarse las fechas de entrega del producto final.
- Reemplazos inadecuados: Por la situación de perdida de un integrante del equipo se busca rápidamente un reemplazo que no siempre será lo mas adecuado debido a la inexperiencia de trabajar en ese equipo particular. Dependiendo de la etapa de avance del proyecto, en determinadas situaciones no se hace conveniente reemplazar la persona que ha dimitido del proyecto y es mejor dejar el equipo conformado de igual forma.

- Perdida de oportunidades estratégicas: La interrupción del trabajo en el desarrollo en un proyecto, cuando los tiempos de entrega son críticos, pueden hacer que la organización pierda oportunidades en el mercado relativas a sus competidores.
- Costos de selección y contratación: Cuando una organización debe conformar o completar un equipo de trabajo para un nuevo proyecto, se deben considerar los costos de selección y contratación de los nuevos integrantes. Estos abarcan desde publicación de avisos, honorarios a agencias de colocación, realización de entrevistas y gastos administrativos en general.
- Costos del entrenamiento y del desarrollo. Una vez seleccionados los nuevos integrantes del equipo estos deben ser entrenados para el nivel de desarrollo esperado. Tanto los costos anteriores como estos costos de entrenamiento involucran no sólo desembolsos económicos sino también un costo en tiempo, como el entrenamiento en las técnicas utilizadas o la integración a un equipo que ya esta trabajando o pronto a iniciar un proyecto. Este coste de tiempo, aunque el nuevo integrante tenga experiencia, involucra el tiempo que lleva en hacerse familiar con la organización.
- Declinación en la moral del grupo de trabajo. Cuando un líder, o una persona estimada, se aleja del equipo se produce una declinación en el rendimiento del desarrollo. Esta situación puede llevar a que otros miembros se alejen, sobre todo cuando el líder se ha ido contra su voluntad; o bien, un rechazo hacia el reemplazante cuando se contrata alguien para llenar el cupo dejado. También se produce un desaliento en los integrantes de un equipo cuando ha habido errores importantes en alguna etapa del proyecto o existe una rivalidad entre dos miembros del equipo.

Entre las consecuencias positivas de la rotación de los integrantes de los equipos de trabajo se pueden mencionar:

- Funcionamiento Creciente: Es la suposición de que el reemplazante es mejor capacitado y habilidoso que el integrante que ha abandonado el equipo. Esto puede mejorar la calidad del trabajo del equipo.
- Ahorro de costos: Al tener equipos de trabajo conformado por antiguos miembros de la organización, los costos de mantener este equipo pueden ser mas elevados que mantener un equipo joven. Por ejemplo, disminución de sueldos al pagar individuos jóvenes y con menos experiencia, menos extensión de los periodos de vacaciones de los miembros

antiguos, pago de pensiones y beneficios, etc. En todo caso, la organización debe evaluar a conciencia el sacrificar experiencia por la disminución de costos de trabajo.

- Innovación y adaptabilidad: La contratación de nuevos individuos puede traer consigo nuevos puntos de vista en el uso de la tecnología y experiencia ganada de otra organización. De esta forma, la organización tendría un método rápido de adaptación al uso de nuevas tecnologías y de ambientes cambiantes en el área de desarrollo de sistemas.
- Ascensos internos crecientes: Cuando los integrantes mayores e importantes de los equipos de desarrollo abandonan la organización se abre la posibilidad de una movilidad interna que permita a integrantes de inferior grado ascender en la escala interna de responsabilidades y sueldos. Esto puede considerarse como un premio al esfuerzo demostrado al desarrollar junto al equipo.
- Moral creciente: Cuando existe un conflicto al interior de un equipo, la productividad decae debido a una baja anímica o moral del resto de los integrantes del equipo. Cuando un individuo no cumple con los plazos, o la calidad de su trabajo es menor a la esperada, resiente la calidad del trabajo del resto. Por ello la separación del equipo de esta persona hace elevar la moral del trabajo. Esta separación de una persona que no cumple con las metas de la organización puede elevar las opiniones de que la organización valora el trabajo de alta calidad.
- Eliminación de los malos comportamientos: La rotación de personas infelices dentro del equipo termina con conductas contrarias a la organización. Se ha observado que estos individuos descontentos no abandonan la organización por su propia cuenta e incurren en conductas impropias, tales como, ausentismo, baja calidad del trabajo, e incluso sabotaje. Este sabotaje, incluso, puede ser peligroso a la organización, cuando el individuo tiene acceso a la seguridad o programación de los sistemas en los cuales se está trabajando.

Estas consecuencias positivas y negativas de la rotación de integrantes de equipos de desarrollo siguiere que, como ya se dijo, los equipos de trabajo son dinámicos, al igual que la personalidad de sus integrantes. Es por ello que la principal tarea de un jefe de proyecto, en compañía con la organización, es evaluar las necesidades de recursos humanos que se necesitan para conformar los equipos de trabajo con relación a los proyectos que se pretendan desarrollar. Por otro lado, también es necesario evaluar el desempeño de los integrantes ya existentes en los equipos ya conformados, con relación a la calidad de su trabajo y la posibilidad de reemplazo de ellos.

Para ello se utiliza una matriz Calidad-Reemplazo que ayuda a determinar al jefe de proyecto, cual es la real necesidad de los individuos dentro de los equipos de desarrollo. Como ayuda a esta tarea se debe mantener una clasificación, dinámica en el tiempo, de la categoría del individuo dentro de la organización y de su integración. Es importante decir que, aparte de mantener relaciones estables de los integrantes dentro del equipo, esta clasificación está orientada a mantener y aumentar la calidad y productividad del trabajo de desarrollo.

Junto a las clasificaciones anteriores, el jefe de proyecto debe realizar una serie de actividades que permitan el buen desarrollo del trabajo en equipo y un adecuado manejo de los recursos humanos, tarea que contribuye a solidificar la posición de la organización en el área de la Ingeniería de Software.

7. Una Guía para la Definición del plan.

Esta guía describe la estructura y contenido de los planes de gestión de proyectos software(SPMP), este identifica algunos elementos esenciales y opcionales que pueden aparecer en todo SPMP.
Esta guía esta pensada para asistir al jefe de proyectos o ingeniero de sw. en la selección, organización y presentación de la información de planes necesitada por jefes, clientes, ingenieros de sistemas o de sw., o miembros de un equipo de proyectos de sw.
Su campo de aplicación va desde sw. comercial, científico o proyectos de sw. militar, todo en tiempo real o orientados a batch. El campo de aplicabilidad no esta restringida por el tamaño, o complejidad del producto sw; por el contrario puede ser usada para producir un SPMP para algún segmento del ciclo de vida de desarrollo de un producto o por una parte o todo el comienzo de desarrollo del producto.

7.1 Conceptos y terminología de la planificación de proyectos sw.

Un proyecto sw. cubre por entero el esfuerzo técnico y administrativo requerido para entregar un producto o un conjunto de productos a un cliente, tales que satisfacen los términos de acuerdo del proyecto. Ese esfuerzo en el proyecto debe ser planeado, iniciado, monitoreado y controlado para asegurar un seguimiento de los planes, presupuesto y criterios de calidad.
Un proyecto sw. tiene una duración, consumo de recursos y produce productos específicos. Algunos de los producto son entregados al cliente; aquellos corresponden a los productos entregables (deliverables) del proyecto.
El acuerdo del proyecto es un documento o conjunto de documentos de acuerdo entre el desarrollador y el cliente que define el alcance, duración, costo y productos a entregar del proyecto. Un acuerdo de proyecto típicamente toma la forma de un contrato, una declaración de trabajo, una especificación de sistema ingenieril, una especificación de requerimiento, un plan de negocio, o una carta de proyecto.
Desde el punto de vista del jefe, la variedad de esfuerzos requeridos para completar un proyecto de sw. puede ser categorizadas como funciones, actividades y tareas del proyecto.
Una tarea es la más pequeña unidad de manejo de responsabilidad, esto es por lo tanto a la unidad atómica de planificación para un proyecto sw. Las tareas tienen duración finita,

consume recursos, y produce resultados tangibles; esos resultados son típicamente un documento o un modulo de código que pueden ser valorado de acuerdo a un criterio de aceptación predeterminado.

La naturaleza exacta del trabajo para realizar completamente las tareas, son especificadas en un paquete de trabajo. El paquete de trabajo típicamente consiste de un nombre, una descripción del trabajo a realizar, las condiciones para iniciar las tareas, la duración estimada de la tarea, recursos requeridos (ejemplo: número y tipo de personal, máquinas, sw, herramientas, viajes, soporte secretarial), el producto del trabajo a ser producido, los criterios de aceptación para el producto, el riesgo, los criterios de término de la tarea y las tareas sucesoras.

Una tarea está exitosamente terminada cuando los criterios de término de las tareas son satisfechos, el criterio de aceptación incluye y puede ser idéntico al criterio de aceptación para el trabajo de los productos producidos por la tarea. El producto de trabajo puede ser la planificación del proyecto, una porción del proyecto planificado, los requerimientos funcionales, un diseño de documento, un conjunto de test de planificación, el manual de usuario, una acta de reunión, memos, listas, el presupuesto del proyecto, o reportes de anomalía.

El tamaño apropiado para una tarea es algo problemático. Durante la planificación inicial del proyecto, las tareas son por necesidad largas unidades de trabajo. Debido a que las tareas son unidades atómicas de medida, planificación y control, las tareas largas deben ser descompuestas a tamaños que permitan un adecuado monitoreo del proyecto. Por otro lado, una tarea puede ser bastante larga para evitar microdiseños del proyecto y permitir alguna autonomía sobre las partes de las tareas trabajadas. Un típico nivel de descomposición de tareas es por duración de una “provisión de personal” semanal a una “provisión” mensual.

Las tareas relacionadas son usualmente agrupadas jerárquicamente dentro de un conjunto de actividades y funciones.

Una *actividad* es la mayor unidad de trabajo (por ejemplo: diseño preliminar y test de integración) que culmina con el alcance de un hito principal del proyecto. Una actividad precisa el día de comienzo y termino, incorpora un conjunto de tareas a ser completadas, recursos que consume, y produce resultados tangibles. Una actividad puede contener otras actividades de una manera jerárquica.

Una *función* del proyecto es una actividad que cubre la duración total del proyecto (ejemplo: Configuración de manejo, aseguramiento de la calidad, y declaración de costos del proyecto).

Actividades y funciones pueden tener subactividades y subfunciones, las tareas son los elementos de más bajo nivel en la jerarquía. La agrupación de tareas relacionadas dentro de

funciones y actividades impone una estructura jerárquica sobre un proyecto permite la separación de la organización y proporciona una guía para organizar el equipo del proyecto. Las relaciones de jerarquía entre actividades, funciones y tareas son frecuentemente representadas usando una estructura de trabajo “breakdown” (bajo quiebre).

Típicamente, una actividad o tarea no puede comenzar hasta que otra actividad halla sido terminada y una tarea o actividad terminada puede ser condición previa para el inicio de otro elemento de trabajo. Relaciones de precedencia pueden ser representadas usando cartas de precedencia, cartas PERT, y cartas de trayectoria crítica.

Las actividades y tareas terminadas resultan en los acontecimientos importantes del proyecto, esto es una lista de eventos usada para medir el progreso (ejemplo: revisión de un diseño preliminar exitoso, o test de integración completado), la obtención de esta lista usualmente resulta de una o más líneas de referencia las cuales son producto del trabajo que tiene que ser formalmente revisados y aceptados.

Un modelo de procesos (o modelo de ciclo de vida) para un proyecto sw. representa la relación de las funciones, actividades y tareas del proyecto. Ejemplos de modelos de procesos incluyen a los modelos de cascada, espiral, incremental, evolucionario y el modelo funcional. Los modelos de proceso para un proyecto sw. deben incorporar explícitamente la iniciación del proyecto y terminación de las actividades de este.

7.2 Una Revisión a la Planificación de Proyectos Software

El plan de gestión de proyectos sw (spmp) es el documento controlador para un proyecto sw. Este especifica los enfoques técnicos y administrativos a ser usados en el desarrollo de un producto sw. o de un componente sw. de un producto extenso. Los planes relacionados para el proyecto (por ejemplo: gestión de la configuración, aseguramiento de la calidad, verificación y validación) son considerados parte del SPMP y ellos deben ser incorporados directamente o por referencia en el SPMP. El plan debe especificar las funciones, actividades y tareas técnicas y administrativas, en suficiente detalle para asegurar que el resultado del proyecto satisfaga las necesidades, requerimientos y acuerdos establecidos del proyecto.

Las variadas partes y subpartes de un SPMP pueden ser ordenadas en la secuencia mostrada a continuación, pero para el desarrollo inicial y la subsecuente actualización del SPMP no se necesita proceder en la secuencia indicada.

Los elementos esenciales de un SPMP son los que se indican a continuación.

1. *Parte Frontal*: Incluye la página de título, una hoja de revisión que contiene la historia de vida del SPMP, un prefacio que resume el alcance y propósito del SPMP, una tabla de contenidos y una lista de figuras y tablas.

2. *Introducción*: Una cláusula de SPMP que especifica el alcance, propósito, la entrega y el mecanismo de manejo de la evolución del SPMP

3. *Referencias* : Se especifica todos los documentos y otros recursos de información referenciados en el SPMP.

4. *Definición* : Especifica todas las definiciones, acronismos y abreviaciones necesarias para la apropiada comprensión del SPMP.

5. *Organización del Proyecto*: Especifica el modelo de proceso, describe la estructura organizacional del proyecto y define claves de responsabilidad para el proyecto.

6. *Proceso de Administración*: Especifica el proceso de administración del proyecto. Los ítems incluidos en esta parte serian consistente con la declaración del alcance e incluiría objetivos administrativos y prioridades y restricciones.

7. *Procesos Técnicos*: Especifica planes para administrar el alcance del producto, los métodos técnicos, herramientas y técnicas usadas para entregar el producto especificado y planes de documentación del sw.

8. *Planificación de las actividades de trabajo*: Especifica las actividades del proyecto y sus alcances; identifica las relaciones de dependencia entre las actividades, establece programas, condición de los recursos requeridos, condiciones para asegurar la calidad y consideraciones de riesgo.

9. *Componentes Adicionales*: Especifica otras ciertas componentes que pueden ir anexados. Áreas adicionales de importancia para un proyecto en particular serían posibles que incluyera un plan de recorrido para los usuarios del producto, un plan de instalación y mantención del producto.

7.3 Descripción Detallada del SPMP

Los elementos esenciales de un SPMP son:

Pagina de Título
Carta de Revisión
Prefacio
Tabla de Contenidos
Lista de Figuras
Lista de Tablas
1 Introducción
1.1 Alcance
1.2 Propósito
1.3 Acuerdo del proyecto
1.4 Evolución de SPMP
2 Referencias
3 Definiciones
4 Organización del Proyecto
4.1 Modelo de Proceso
4.2 Estructura Organizacional
4.3 Limites e Interfaces Organizacionales
4.4 Responsabilidades del Proyecto
5 Procesos Administrativos
5.1 Objetivos y Prioridades Administrativas
5.2 Dependencias, Restricciones y Supuestos.
5.3 Procesos Integrales
5.4 Gestión del Alcance Administrativo
5.5 Planes de gestión del Itinerario
5.6 Plan de gestión del Presupuesto
5.7 Plan de gestión de los Recursos
5.8 Planes de gestión de la Calidad
5.9 Planes de gestión de Riesgos

5.10 Planes de Obtención de Recursos
5.11 Planes de Manejo Comunicacional
6 Procesos Técnicos
6.1 Alcance del Producto
6.2 Métodos, Herramientas y Técnicas
6.3 Documentación del Software
7 Planificación de las Actividades del Trabajo
7.1 Definiciones de las Actividades y Alcance
7.2 Dependencia de las Actividades
7.3 Itinerario de Actividades
7.4 Presupuesto de Actividades
7.5 Requerimientos de Recursos de las Actividades
8 Componentes Adicionales
8.1 Anexo
8.2 Índice.

La página de título contiene el título de y una nota de revisión para identificar unívocamente el documento.
La carta de revisión es una hoja separada que contiene el número de versión del documento, la fecha de revisión, firma de aprobación, una lista de las páginas que han sido modificadas en la actual versión y una lista de los números de versión y fechas de revisión para cada una de las versiones previas del SPMP.
El prefacio indica el alcance de las actividades del SPMP.
La tabla de contenido y las listas de figuras y tablas proveen los títulos y los números de página.

7.3.1 Introducción

Esta parte provee una revisión tanto del proyecto como del producto a ser confeccionado. El alcance del proyecto y el producto, una lista de la entrega del proyecto y las consideraciones de evolución para e SPMP.

7.3.1.1 Alcance

Esta parte definiría el alcance tanto del proyecto como del producto a ser entregado.

7.3.1.2 Propósito

Aquí se provee una resumida declaración de las necesidades del negocio a ser satisfechas por el proyecto, con un conciso resumen de los objetivos del proyecto.

7.3.1.3 Acuerdo del proyecto

En esta parte se listan los productos que van a ser entregados al cliente, las fechas y lugar de entrega y la cantidad requerida para satisfacer los acuerdos del proyecto.

7.3.1.4 Evolución del SPMP

Se especifica que mecanismos son usados para lograr la versión inicial del SPMP y cambios de control del SPMP.

7.3.2 Referencias

Se provee una completa lista de todos los documentos y otros recursos de información referenciados en el SPMP. Cada documento puede ser identificado por el título, número de edición, fecha, autor, y organización editora. Otros recursos, como los archivos electrónicos, deben ser identificados de una manera no ambigua usando identificadores como la fecha y número de versión.

7.3.3 Definiciones

Se puede definir o provee referencias de la definición de todos los términos y acronismos requeridos para interpretar adecuadamente el SPMP.

7.3.4 Organización del Proyecto

Esta parte especifica el modelo del proceso para el proyecto, describe la estructura organizacional del proyecto, identifica el fin de la organización y define las responsabilidades individuales para el proyecto.

7.3.4.1 Modelo de Procesos

Esta parte define las relaciones entre las funciones del proyecto principal y las actividades por especificación de tiempo. El modelo de proceso puede ser descrito usando una combinación de gráficos y notación textual.

7.3.4.2 Estructura Organizacional

Esta parte describe la estructura de organización interna del proyecto. Herramientas gráficas tales como una matriz de diagramas podría ser usada para describir las líneas de autoridad, responsabilidad y comunicación dentro del proyecto.

7.3.4.3 Limites e Interfaces Organizacionales

Aquí se describe los limites administrativos y gerenciales entre el proyecto y cada una de las siguientes entidades: La organización de origen, la organización del cliente, la organización subcontratada, o cualquier otra organización que interactua con el proyecto.

7.3.4.4 Responsabilidades del Proyecto

En esta parte se identifica el estado natural de cada función y actividad del proyecto principal, y identifica individualmente quienes son responsables por esas actividades y funciones. Una matriz de funciones y actividades versus responsabilidades individuales pueden ser usadas para describir las responsabilidades del proyecto.

7.3.5 Procesos Administrativos

Esta parte especifica los procesos administrativos del proyecto que deben ser consistentes con la declaración del alcance y puede incluir objetivos y prioridades administrativas.

7.3.5.1 Objetivos y Prioridades Administrativas

Especificar la filosofía, metas y prioridades para la administración de las actividades durante el proyecto. Tópicos específicos que pueden ser incluidos son la frecuencia y mecanismos de reporte a ser usados, las prioridades relativas entre los requerimientos y presupuesto del proyecto, procedimientos administrativos de riesgos a seguir y una nota para la adquisición, modificación y uso del software existente.

7.3.5.2 Dependencias, Restricciones y Supuestos

Condiciones sobre las cuales el proyecto esta basado, los eventos externos de los cuales el proyecto depende y las restricciones con las cuales el proyecto va a ser elaborado.

7.3.5.3 Procesos Integrales

Planifica los procesos integrales necesarios para una exitoso término de los proyectos sw. Esos procesos pueden incluir: la configuración de manejo, asegurar la calidad, verificación y validación del sw.

7.3.5.4 Gestión del Alcance Administrativo

Se especifica el plan de manejo del alcance del proyecto, también puede incluir procedimientos para cambios en el alcance del SPMP, además de la probabilidad de cambios, incluyendo los factores que pudiesen resultar por el cambio del alcance del proyecto.
El plan indicaría factores que causaron el cambio, los resultados de los cambios y los métodos por los cuales los cambios fueron documentados, comunicados y controlados.

7.3.5.5 Planes de Manejo del Itinerario.

Se define los planes para asegurar que el proyecto este terminado a tiempo, además de, especificar los documentos que sirven como entrada al proyecto. Se deben especificar las herramientas o metodología que serán usadas para administrar el itinerario.
El plan indicaría factores que causaron el cambio, los resultados de los cambios y los métodos por los cuales los cambios fueron documentados, comunicados y controlados.

7.3.5.6 Plan de Manejo del Presupuesto

Se definen los planes para asegurar que el proyecto sea terminado con el presupuesto establecido, además de especificar los documentos que sirven como entrada al presupuesto. Se deben especificar las herramientas o metodología que serán usadas para administrar el presupuesto.
El plan indicaría factores que causaron el cambio, los resultados de los cambios y los métodos por los cuales los cambios fueron documentados, comunicados y controlados.

7.3.5.7 Plan de Manejo de los Recursos

Se especifica los planes de manejo de los recursos requeridos para la exitosa culminación de esta. El plan puede especificar los métodos usados para estimar el material, servicios y requerimientos de recursos humanos. Se puede incluir el número y nivel de calificación del personal requerido para completar el proyecto, el número y atributos de calidad requerida por el personal y la naturaleza de los servicios contratados.

7.3.5.8 Planes de Manejo de la Calidad

El plan para asegurar la calidad de los procesos y del proyecto se define en este apartado. El plan puede identificar los estándares del proyecto y los métodos y recursos requeridos para implementar y asegurar el cumplimiento de esos estándares.

7.3.5.9 Planes de Manejo de Riesgos

Se definen los planes de manejo de factores de riesgos asociados al proyecto. Esta parte describe los métodos que serían usados para identificar los factores de riesgo, como fueron evaluados y impacto potencial de los riesgos identificados.

7.3.5.10 Planes de Obtención de Recursos

Esta parte incluiría una descripción de los procesos de obtención de recursos, incluyendo responsabilidades para todos los aspectos del proceso. El plan podría incluir los procesos de obtención tangible e intangible de recursos; procesos de obtención de: equipamiento, estimaciones de tiempo de computado, Hw. y Sw. y transportes.

7.3.5.11 Planes de Manejo Comunicacional

Esta parte maneja la comunicación relativa del proyecto**.** Se definirían los mecanismos de reporte, formas de reporte, información anticipada, mecanismos de intervención, y otras herramientas y técnicas usadas para monitorear y controlar los objetivos del proyecto.

7.3.6 Procesos Técnicos

Esta parte planea el manejo del alcance del producto, los métodos técnicos, herramientas y técnicas usadas para la entrega del producto y documentación del software.

7.3.6.1 Alcance del Producto

Esta parte contempla los planes de manejo del alcance del producto. En este plan serían incluidos los métodos por los cuales el alcance del proyecto va a ser medido además de los requerimientos del producto, además se incluye los factores que pueden cambiar el alcance del producto.

7.3.6.2 Métodos, Herramientas y Técnicas

Este parte describe el sistema de computación, metodologías de desarrollo, estructuras de equipos, lenguaje de programación, herramientas y técnicas a ser usadas para la especificación, diseño, construcción, test, integración, documentos, modificaciones y mantención del proyecto.

7.3.6.3 Documentación del Software

Se describe los planes de documentación para el proyecto software. Los planes de documentación especificarían los requerimientos de documentación y la importancia,

referencias, revisiones para la documentación del sw. El plan de documentación puede además contener un estilo de guía, convenciones de nombre y formatos de documentación.

7.3.7 Planificación de las Actividades del Trabajo

Esta parte definen las actividades y sus alcances, identifica las relaciones de dependencia entre las actividades, estado de los recursos, asegurar la calidad y las consideraciones de riesgo.

7.3.7.1 Definiciones de las Actividades y Alcance

Aquí se especifican y definen las actividades a ser completadas para satisfacer los requerimientos del proyecto. El alcance de cada actividad seria claramente definida, esta identificación puede ser basada sobre la numeración de proyectos y/o títulos descriptivos.

7.3.7.2 Dependencia de las Actividades

Se especifica el orden entre las actividades del proyecto y tareas asociadas para describir las relaciones de dependencia entre actividades y eventos externos.

7.3.7.3 Itinerario de Actividades

Estas listas expresan calendarios de tiempo o de incrementos relativos del producto clave sobre el proyecto principal.

7.3.7.4 Presupuesto de Actividades

Especifica el presupuesto de varias tareas y actividades.

7.3.7.5 Requerimientos de Recursos de las Actividades

En esta parte se especifica, como una función de tiempo, los recursos totales estimados para completar la actividad. Números y tipo de personal, soporte de software y hardware y requerimientos de mantención para las actividades del producto y tareas.

7.3.8 Componentes Adicionales

Ciertos componentes adicionales que puedan ser requeridos pueden ser incluidos para ser anexados como material adicional al SPMP.

7.3.8.1 Anexo

Incluiría detalles específicos del personal, detalles de los costos estimados, detalles de las estructuras de trabajo “breakdown” e información suplementaria para una adecuada comprensión del SPMP

8. Control del avance.

Mucho se habla de cómo establecer parámetros para determinar avances en los proyectos de SW y más aún cuando este proyecto se encuentra con problemas. Cuando tal situación se presenta una revisión del proyecto puede ayudar.

A continuación se presenta el QUIÉN, QUÉ, CUANDO, DONDE, POR QUÉ y el CÓMO la administración del proyecto de SW revisa, incluyendo las herramientas adecuadas para el equipo de revisión, además de como esto permite descubrir las fortalezas y debilidades de un proyecto. La revisión ayuda a identificar los problemas de la organización que carecen de una disciplina de administración. Esta revisión entrega el estado del proyecto en algún instante.

El equipo de revisión debe ser sensible al clima de la organización y respecto de lo que concierne a la línea de manejo. Uno de los aspectos más importantes es que el equipo de revisión no debe sobredimensionar los problemas de la organización ya que el proceso de revisión podría destruir el proyecto. Otro problema menos importante pero que puede ser desastroso a la larga es cuando la administración del proyecto se comporta en forma defensiva debido al método de alta administración del equipo de revisión y rechazar las recomendaciones aún cuando estas puedan ser útiles.

Las revisiones de la administración de proyectos de SW pueden cambiar la forma en la cuál un proyecto es manejado. Además, con la participación en una revisión, los revisores pueden relacionar la teoría de administración de proyectos SW a situaciones reales sin exponerse a capacitaciones. Un equipo de revisión tiene "El efecto de lograr del grupo que esta siendo revisado piense más en donde ellos están, que necesitan hacer y dar al personal una oportunidad de expresar sus temores y problemas sin trabas de admitir fallas a sus supervisores "

Además, aquellos que han sido revisados a menudo se sienten halagados cuando consultan su opinión, estas personas al estar trabajando en el proyecto casi siempre saben que cosas están bien, cuáles están mal y cuáles se pueden mejorar. El equipo de revisión hace la función de catalizador para obtener recomendaciones escondidas de la organización, para esto los revisores deben establecer un nivel de confianza con aquellos que están en el proyecto y así estos puedan hablar libremente.

8.1 Qué cosas considera la revisión

Una revisión es un repaso de los procesos, herramientas y la organización empleada en el desarrollo de un sistema SW. La revisión se concentra en la tecnología del ciclo de vida de desarrollo de SW que está siendo aplicado. El foco de revisión está en el análisis del itinerario, herramientas de administración del proyecto y los tópicos cargados emocionalmente de las relaciones de trabajo. El equipo de revisión debería buscar temas comunes más allá de los limites de la organización y niveles de administración.

8.2 Qué cosas no debería considerar

Los administradores top no deberían considerar la revisión como una forma de reparar, ya que no es una vista completa del estado del proyecto. Existen tentaciones para poner a alguién en el equipo de revisión, el cual más tarde será transferido al proyecto como un administrador clave. Esto puede originar desconfianza del nuevo administrador simplemente porque la confidencialidad del proceso de revisión es quebrantada cuando el revisador se une al proyecto.
La revisión debe instituirse cuando se necesita, y no debiera planificarse como parte del ciclo de vida del proyecto. La revisión es más efectiva en una área del proyecto donde hay una falla real o potencial.

8.3 Cosas para buscar

Aquí se presentan algunas pistas para encontrar los problemas del proyecto. Hay que centrarse en los siguientes aspectos.

1. Factores en desacuerdo entre administradores
2. Desacuerdo con clientes , especialmente en lo entregado
3. Afirmaciones como "Las cosas obtenidas no son lo que originalmente fueron proyectadas, "Los usuarios no han sido capacitados, esa es la razón de ..."
4. La gente clave está siendo cambiada de crisis en crisis
5. La gente clave se aleja del proyecto.
6. Roles no definidos de los administradores y desarrolladores y una pobre relación interorganizacional
7. Se establece poco compromiso y los que se asumen, no se comunican a los miembros del proyecto

8. Se embarcan en la primera situación antes que las pruebas estén completas y a situaciones adicionales sin el tiempo para un adecuado afiatamiento
10. Los puntos críticos no están siendo chequeados, por ejemplo no existe fijado un día para entregar el SW a una prueba
11. Listados de programas ilegibles, demasiados comentarios o muy pocos
12. Una preocupación en dar énfasis a forzar el uso de estándares más que en su utilidad
13. Errores encontrados tardíamente en el ciclo de vida del SW
14. Programas de administración ignorados
15. Una variedad de enfoques al diseño de SW y ninguna correlación aparente entre diseño de SW y los programas

El proyecto puede estar en serios problemas cuando el administrador del proyecto dice " La carga es el doble de lo que esperábamos y necesitamos una maquina más grande", "El cliente está retrasado con los requerimientos".
Las revisiones han identificado donde hay una falta de herramientas de desarrollo, cuando el almacenamiento y el tiempo real obliga a contar con diseño y donde una revisión del diseño de planes de prueba o código de revisión no existe con la poca visión por parte de los administradores del proyecto de las consecuencias de estos defectos.
Un equipo de revisión de un proyecto extenso emplea sobre 100 personas y toma de 4 a 6 días. Esto debería ser ocupado en la planta de desarrollo, con cada sesión durando 2 días. Las sesiones deben realizarse con 2 semanas de diferencia.
Para mantener la revisión en vigencia , es esencial que la propaganda sea eliminada. Esto puede ser realizado antes de la reunión teniendo información distribuida a los revisadores acerca del estado del proyecto y la función que el proyecto se supone realiza, incluyendo cualquier información de marketing que este disponible.

8.4 Por qué revisar

Las revisiones encuentran los problemas tempranamente. Cuando los planes , objetivos y/o asignación de recursos parecen irreales , la revisión puede ser usada para socorrer el llamado de ayuda de los administradores del proyecto. Por otro lado la revisión puede ser usada para alejar a administradores negativos.

8.5 Cómo revisar

Las entrevistas individuales y grupales son las herramientas fundamentales en revisiones. El cuestionario en el apéndice A proporciona un esquema útil para estructurar las entrevistas. Los revisadores deberían pasar al menos una tercera parte de su tiempo en entrevistas informales de uno en uno con desarrolladores y la primera línea de supervisores para ganar comprensión en el trabajo de la organización. Esto es útil en la mantención de discusiones que se alejan del tema.

8.6 Quién revisa

El equipo de revisión debería consistir de 3 a 5 personas externas a la organización que esta siendo revisada. El líder del equipo de revisión debería ser alguién en una organización similar que tenga alguna familiaridad con el proyecto y conozca a los miembros del proyecto. Es recomendable que alguno de los revisadores tenga experiencia previa. Es importante que ellos sean desarrolladores y de preferencia administradores de proyecto con experiencia. Es deseable que al menos uno de los miembros del equipo de revisión sea un administrador de proyecto altamente experimentado que pueda hacer preguntas perspicaces y facilite la identificación del problema.

8.7 Herramientas para el equipo de revisión

La herramienta más importante que el equipo de revisión puede usar es escuchar a los no directivos. Es esencial escuchar cuidadosamente lo que la gente esta diciendo y tratar de obtener de sus discusiones los temas comunes.
Una segunda herramienta es el cuestionario en el apéndice A que provee un plan de entrevista estructurada y que se focaliza en las conclusiones de la administración las cuales pueden ser de poco agrado de discutir para el desarrollador. Es mucho más agradable hablar de que hará el proyecto , que es hablar de cómo las personas trabajan juntos , particularmente cuando hay conflictos.
La tercera herramienta es usar un conjunto de de entrevistas. Personas de todos los niveles deberían ser seleccionadas para la entrevista y dando la oportunidad de discutir el cuestionario y cualquier otro punto que llegue a ser importante.

La cuarta herramienta es elogio público y simpatía. Esto es útil para los revisadores, para indicar cuando ellos están enfrentando los mismos problemas en sus proyectos. Pregunte a las personas en forma aleatoria que están haciendo y por qué. Los revisadores pueden obtener rápidamente un sentimiento de cuan bien los procedimientos están siendo seguidos y cuando obedece a los estándares existentes.

8.8 Reacciones a la entrevista de revisión

A continuación se dan algunos comentarios de las personas que han sido entrevistadas , "Me siento bien al ser capaz de decir lo que había estado en mi mente "," Nunca nadie me pregunto antes", "Nosotros estamos más tranquilos al oír que otros proyectos tienen problemas", "esto nos ha ayudado a ser más reflexivos".
Durante la entrevista es importante que cada persona diga algo y que algunas preguntas sean dirigidas a aquellos que son más callados. El equipo de revisión debe proporcionar un clima de participación y no dejar que ninguna persona domine la discusión.

8.9 Apéndice A : Cuestionario entrevista de revisión

I. Cuál es el estado del proyecto
II. Objetivos y requerimientos del Proyecto
   A. Cómo sienten los requerimientos, son ellos :
      1. Demasiados vagos
      2. Demasiados específicos
      3. Precisos
   B. Están claras las prioridades de los requerimientos individuales
III. Definición y organización del proyecto
   A. Cómo está siendo planificado el proyecto
   B. Cuál es el rol y la relación con el cliente
IV. Staffing y destreza en los laboratorios
   A. Como y en qué proporción está el trabajo dividido entre las organizaciones ? Describa la interacción entre las diversas organizaciones

B. Como fue el proyecto dotado de personal como una función del tiempo (incluyendo proyecciones en el futuro) ? (Incluya nivel de experiencia de la gente inicial tanto de administradores como personal) .¿ Hay problemas de staffing.?

C. Que programas de capacitación tiene para los nuevos empleados.

D. Tiene alguna organización especial propia del staff ? Si es así , cuál es su rol ?

E. Cuanto sobre tiempo se trabajo?

V. Controles de proyecto y seguimientos

A. Cuales son y quienes los monitorean ? Para

1. Planes
2. Hitos (Incluye hitos iniciales y planes actuales)
3. Diseño de documentación de registros
4. Documentación del usuario
5. Técnicas de desarrollo

B. Cuales parámetros del proyecto son seguidos ? Por ejemplo :

1. Planes
2. Tiempo real
3. Uso de memoria
4. Uso de ancho de banda para los I/O

C. Cómo son controlados los cambios

D. Cómo son descubiertos y tratados los problemas

E. Cuánto tiempo se pasa estimando estados

F. Quién es el administrador de más alto nivel que siempre lee los códigos ?

G. Quién es el administrador de más alto nivel que frecuentemente lee los códigos ?

VI. Estableciendo decisiones

A. Cómo son establecidas las decisiones claves ?

B. Cómo fue resuelta la crisis más significativa ?

C. Qué nivel de aprobación es requerido para qué tipo de decisiones (Quién se encarga de qué?)

D. Cuál es el rol de la fuerza de trabajo y comités

VII. Comunicaciones

A. Cómo son manejadas las comunicaciones al interior del proyecto (verbal y escritas) ?

1. Entre cada organización
2. Verticalmente a través de la línea de mando

3. Entre colegas (mismo cargo)

VIII. Herramientas de administración de proyectos

A. Cuáles son las herramientas de administración y técnicas que más contribuyen a este proyecto ?

B. Cómo pueden ser modificadas para que sean más efectivas ?

C. Hay herramientas que deberían ser usadas y no lo son ?

D. Cómo es medida y estimada la productividad del programador ?

IX. Enfoque técnico

A. Cuáles son los mayores riesgos ?

B. Cuáles son los ambientes de desarrollo y ejecución ?

C. Cuánto esfuerzo es dedicado a las herramientas SW ? Cuáles son y qué tan útiles son ?

D. Cómo es desarrollada la información de usuario y cuál es el rol de la ingeniería del desempeño humano?

E. Cuál es el enfoque del sistema de prueba ?

F. Cuáles proyectos externos realizaste y estás aprendiendo de ellos ?

X. Pronóstico

A. Cuáles son los problemas más críticos que enfrenta el proyecto ?

B. Cómo sugieres que ellos sean resueltos ?

XI. Preguntas adicionales para la entrevista individual

A. Describe tu asignación

B. Cuáles son tus criterios de éxito ?

C. Qué te frustra en tu trabajo ?

D. Qué te satisface en el trabajo ?

E. Qué necesitas hacer para una contribución más significativa en tu trabajo ?

9. Reglas para la dirección exitosa de proyectos.

9.1 Regla número 1:Fije una meta clara

1. Al planear un proyecto se empieza con el resultado final - la meta - y luego se trabaja hacia atrás a partir de ella.
2. Los jefes de proyecto efectivos mantienen siempre los ojos puestos en la meta y se aseguran que todos en el grupo (incluido el usuario final) apunten en la misma dirección.
3. Los jefes de proyecto efectivos desarrollan una meta INTELIGENTE, se aseguran que sea clara, se la comunican a toda su gente, crean compromiso en torno a ella, y confirman que los miembros del grupo constantemente la tengan en cuenta y trabajen por alcanzarla.
4. Propiciar una visión común hace que cada miembro del equipo del proyecto o del contingente de trabajo esté orientado en la misma dirección.

9.2 Regla número 2: Precise los objetivos

1. Los objetivos que se establecen a través del diálogo descomponen la meta del proyecto en tareas específicas y comprometen a sus miembros con cada una de ellas.
2. Los objetivos le ayudan a identificar quién debe estar en el grupo del proyecto y contribuyen a crear una sensación de propiedad entre los miembros del grupo.
3. Evite concentrarse en forma demasiado estrecha en los objetivos y perder de vista la meta global del proyecto.
4. Formule recompensas que estén ligadas tanto al éxito global del proyecto como al de los objetivos.
5. Asegúrese de que la responsabilidad por el éxito de un proyecto genera la autoridad correspondiente.

9.3 Regla número 3: Establezca puntos de control, actividades, relaciones y estimaciones de tiempo.

1. Desarrolle el cuerpo de su proyecto definiendo los puntos de control que señalen los progresos, las *actividades* que llevan a la realización del proyecto, las *relaciones* entre actividades y las estimaciones de *tiempo* (costos y otros recursos) para cada una de ellas.

2. Al descomponer los elementos de un proyecto es necesario partir de la meta y retroceder hasta llegar al primer paso que sea necesario dar para alcanzarla.
3. Las personas trabajan mejor cuando saben qué avances están haciendo en la consecución de la meta. Fije los eventos y las actividades para ayudar a que los miembros del proyecto supervisen los progresos del proyecto y para motivarlos.
4. Sea creativo y riguroso al considerar las relaciones entre las actividades del proyecto. Asegúrese de incluir las posibilidades "¿qué pasa si ... ?" y "¿qué podría salir mal?", para tener a la mano planes de contingencia.

9.4 Regla número 4: Ilustre gráficamente el programa de trabajo

1. Dos métodos comunes y eficaces, utilizados para elaborar planes de proyectos y mantenerlos dentro del programa de trabajo, son los gráficos de barras y los diagramas de flujo.
2. Los gráficos de barras proveen una rápida visión general del proyecto y permiten que los miembros del equipo puedan supervisar fácilmente sus avances.
3. Los diagramas de flujos son más complejos que los gráficos de barras. Ayudan a identificar y a manejar el flujo secuencial de las actividades críticas de un proyecto.
4. Trabajando con gráficos de barras y con diagramas de flujo usted puede desarrollar un plan efectivo de proyecto. Si usted continúa empleándolos *durante* el desarrollo del proyecto, facilita la llegada a la meta dentro del tiempo previsto, cumpliendo con el presupuesto y con las normas de calidad exigidas.

Ahora bien, el hecho de que ya se ha elaborado el plan y se le ha dado comienzo no significa que ya se haya hecho todo el trabajo. El plan y el programa de trabajo no son el resultado final del proyecto, solamente son medios para lograrlo. Es necesario tener un jefe que lleve el proyecto hasta la línea final. Hay que llevar el proyecto hasta la meta final, a veces con tropiezos y utilizando métodos rudimentarios y de avanzada. La capacidad y visión requeridas para lograrlo exitosamente se condensan en las seis reglas siguientes.

9.5 Regla número 5: Capacite a las personas, individualmente y como equipo

1. Para dirigir con éxito un equipo, es necesario entender el comportamiento humano: primero el suyo, y luego el de los miembros de su equipo.

2. La mala comunicación es a menudo el resultado de un choque de perspectivas. Es necesario aprender cómo perciben los demás sus metas y acciones, y precisar una perspectiva común para el equipo.
3. Es posible que cada miembro del equipo tenga una motivación diferente. Una importante tarea del líder del equipo es reconocer los motivos de cada uno de los miembros y convertirlos en recompensas para ellos.
4. Los equipos no se conforman en un instante. Para llegar al deseado nivel de alta productividad, el equipo debe pasar por las etapas de orientación, insatisfacción y resolución. En la medida en que el proyecto se acerque a su fin, también debe manejarse la etapa de terminación.

9.6 Regla número 6: Refuerce el compromiso y el entusiasmo del personal

1. Desarrollando un sentido de compromiso con el objetivo, así como el entusiasmo por el proyecto, los líderes del equipo pueden animar a los participantes para que se “apropien” de él.
2. Preséntele a la gente una gran imagen y asegúrese de que su proyecto o equipo de trabajo no es visto simplemente “como otro trabajo”, sino como un desafío.
3. Demuestre que los valores e intereses de la gente están sincronizados entre sí y con los objetivos del proyecto. Conviértase en la llama que prende el fuego en los demás.
4. Se requiere compartir la información y la autoridad si es que las personas han de creer que usted considera que sí pueden hacer el trabajo. La responsabilidad sin los recursos y la autoridad necesarios siempre ha sido la receta de los tontos.
5. El compromiso crece cuando las acciones de la gente (y sus realizaciones) se hacen visibles para los demás. Hay que ser el líder que dirige y alienta los vítores de su grupo, porque no existe algo así como un animador de gerentes.

9.7 Regla número 7: Informe a las personas relacionadas con el proyecto

1. Las fallas de comunicación en los proyectos se deben a una serie de obstáculos que pueden superarse. La clave son los bloques organizacionales que intervienen debido a la naturaleza misma del proyecto o del contingente de trabajo: personas de diferentes departamentos que trabajan en una misma tarea, única e interrelacionada.

2. Para transmitir su mensaje más efectivamente, a) toque a los demás allí donde están, b) dígales por qué su mensaje tiene importancia para ellos, c) manténgalos informados regularmente y d) comuníquese afirmativamente.
3. Escuchar es aún más importante que hablar. Los pasos críticos para saber escuchar incluyen concentrarse en el mensaje, escuchar sin hablar, escuchar hasta que el otro termine y fijarse en las señales verbales tanto como en las no verbales.

9.8 Regla número 8: Estimule al personal estableciendo acuerdos

1. Aprender a manejar los conflictos, en lugar de evitarlos, puede ayudar al líder de un equipo a alcanzar sus objetivos. Los conflictos representan un compromiso emotivo con un proyecto y cuando se mantienen bajo control pueden llevar a los miembros del equipo a solucionar los problemas en forma cooperativa.
2. Generalmente, en el ciclo de vida de un proyecto o de un contingente de trabajo surgen diferentes fuentes de conflicto. Saber cuándo y por qué se originan estos conflictos permiten aprender a canalizar las energías de los miembros del equipo.
3. Para resolver estos conflictos se requiere crear un terreno común, ampliar las áreas de acuerdo, recoger información y concentrarse en los problemas y *no* en las personas.
4. Al mediar en un conflicto, recuerde que ambas partes deben quedar satisfechas, no sólo racional sino también emotivamente.

9.9 Regla número 9: Aumente el poder: tanto el suyo como el de los demás

1. Gran parte del poder real que tienen los gerentes de proyecto y los líderes de contingentes de trabajo proviene del poder personal derivado de tres fuentes: poder: de referencia, de experto y de relaciones.
2. Cuando los líderes comparten el poder con los miembros del equipo, puede lograrse un mayor nivel de compromiso y participación.
3. La gente busca en sus líderes cuatro características personales: honestidad, competencia, visión y capacidad para ser fuente de inspiración.

9.10 Regla número 10: Atrévase a acercarse con creatividad a los problemas

1. Contrario a lo que mucha gente cree, es posible enseñar creatividad a las personas.
2. Antes de que las personas puedan permitirse ser creativas, su trabajo es eliminar los obstáculos potenciales para la toma de riesgos.
3. Las metas y las fechas límite pueden ayudar a estimular la creatividad.
4. Cuatro estrategias para alentar la creatividad son la sensibilidad, la fluidez en las ideas, la originalidad y una mayor flexibilidad.

10. Bibliografía.

- Richard Thayer ed., "Software Engineering Project management", IEEE Computer Society, 1998.
- Marcela Varas C., "Apuntes de clases: Gestión de Proyectos de Ingeniería de Software", Universidad de Concepción, 1998 (en colaboración con Ximena Hormazábal, Luis Monsalve, Jorge Muñoz, César Olivares, Rodrigo Oviedo y Carmen Wolff).
- Ricardo Contreras, "Apuntes de clases: Ingeniería de Software", Universidad de Concepción, 1997
- Steve McConnell, "Desarrollo y Gestión de Proyectos Informáticos", McGraw Hill 1996
- W. Alan Randolph, Barry Posner, "Gerencia de Proyectos", Mc Graw Hill 1993
- Watts Humphrey, "Managing the Software Process", Addison Wesley 1990
- Barry Boehm, "Software Engineering Economics", Prentice Hall 1981