UNIDAD IV

ESTIMACION Y PLANIFICACION TEMPORAL DEL PROYECTO DE SOFTWARE

Contenido:

4.1 Introducción

4.2 Factores que influyen en el costo del software

4.3 El proceso de Estimación

4.4 Métricas del Software

4.5 Tecnicas de estimación de costos

- Juicio Experto
- Tecnica DELFI
- Método Histórico
- Modelo de Costos COCOMO
- Ecuación de Primer Orden de Jones
- Técnica Iterativa

4.5 Consejos sobre estimaciones

4.7 Calendarización

4.8 Especificación del Plan del Proyecto

4.1 Introducción

Algunas estimaciones se hacen cuidadosamente, pero otras simplemente se hacen a ojo. La mayoría de los proyectos rebasan sus límites de sus planes estimados de un 25 a un 100%, son muy pocas las organizaciones que han logrado realizar una predicción con una precisión de un 10%.

Una estimación precisa del plan de desarrollo constituye una de las bases para obtener una velocidad máxima de desarrollo.

Ejemplo. Estimación de proyectos a ojo

Carl había sido encargado de la versión 1 del sistema de control de Inventario de Giga Safe (SCI). Tenía ideas generales sobre las funciones deseadas cuando asistió a la primera reunión del comité de supervisión del proyecto. Bill era el responsable principal del comité de supervisión. "Carl, cuánto durará el SCT 1.0?», preguntó.

"Creo que tardará unos 9 meses, pero sólo es una estimación a ojo" dijo Carl.

«No puede tardar tanto», dijo Bill. «Esperaba que dijera 3 o 4 meses. Necesitarnos absolutamente tener el sistema dentro de 6 meses ¿Puede hacerlo en seis meses?»

«No estoy seguro», dijo Carl honestamente. «Tendría que mirar el proyecto con más detalle, pero puedo intentar tenerlo en 6 meses.»

«Entonces considero un plazo de 6 meses», dijo Bill. «De todas maneras, eso es lo que tenía que ser.» El resto del comité asintió.

Pasadas 5 semanas, el trabajo adicional realizado sobre el concepto del producto había convencido a Carl de que el proyecto iba a necesitar más bien los 9 meses iniciales, en vez de en 6 meses, pero pensó que con un poco de suerte aún podría estar finalizado en 6 meses. El no deseaba consideraran como una persona problemática, de forma que decidió apretarse el cinturón.

El equipo de Carl hacía grandes progresos, pero el análisis de requerimientos necesitó más tiempo de lo que se esperaba. Llevaban casi 4 meses de lo que se suponía un proyecto de 6 meses. «No hay forma de realizar el resto del trabajo en 2 meses», le dijo a Bill. Carl le comentó a Bill que llevaban un retraso de dos meses respecto a la planificación y que el proyecto tardaría 8 meses.

Unas semanas más tarde, Carl comprobó que el diseño tampoco se estaba realizando tan rápidamente como se esperaba. «Implementen las partes que se puedan hacer rápidamente», le dijo al equipo. «Nos preocuparemos del resto de las partes cuando lleguemos a ellas,»

Carl se reunió con el comité de supervisión. «Vamos por el séptimo mes de los 8 que dura el proyecto. El diseño detallado está casi finalizado, y se van haciendo grandes progresos. Pero no podemos terminar el proyecto en los 8 meses.» Carl anunció su segundo aplazamiento, esta vez a 10 meses. Bill se quejó y requirió a Carl que buscara formas de volver a la planificación de 8 meses.

Al límite del noveno mes, el equipo había finalizado el diseño detallado, pero la codificación de algunos módulos aún no había comenzado. Estaba claro que Carl tampoco podría cumplir la planificación de los 10 meses. Anunció un tercer aplazamiento (a 12 meses). Cuando Carl le anunció el aplazamiento, la cara de Bill se puso roja, y la presión llegó a ser más intensa. Carl empezó a sentir que su trabajo estaba en juego.

La codificación iba bastante bien, pero unas cuantas partes necesitaban volver a ser diseñadas e implementadas. En esas partes, el equipo no había coordinado bien los detalles del diseño, y algunas de sus implementaciones entraban en conflicto. En la reunión de los 11 meses del comité de supervisión, Carl anunció el cuarto aplazamiento, a 13 meses. Bill se quedó lívido. «¿Tiene alguna idea de lo que está haciendo?», le dijo a gritos. «¡Obviamente no tiene ni idea! ¡No tiene ni idea de cuándo va a terminar el proyecto! ¡Ahora mismo le voy a decir cuándo va a estar terminado! ¡Va a estar terminado en 13 meses, o lo voy a despedir! ¡Estoy cansado de ser manipulado por los tipos del software! ¡Usted y su equipo van a trabajar 60 horas a la semana hasta que terminen!»

Carl sintió cómo subía su presión arterial, especialmente porque Bill insistió en la planificación tan poco realista que habían hecho al principio. Pero él sabía que con cuatro aplazamientos en su haber, había perdido toda credibilidad. Sabía que tendría que hacer horas extras a la fuerza o le echarían del trabajo.

Carl le comentó a su equipo lo que sucedió en la reunión. Ellos trabajaron duro y consiguieron entregar el software en 13 meses. La implementación adicional no cubrió los flecos del diseño adicional, pero trabajando todo el mundo 60 horas a la semana, a sangre y fuego, entregaron el producto.

Naturaleza de la estimación del software

La estimación de software es difícil. Los jefes, directivos, clientes y desarrolladores no parecen entender porqué la estimación es tan difícil.

El argumento básico de la estimación de software es que es un PROCESO DE REFINAMIENTO GRADUAL.

Se comienza con una imagen borrosa de lo que se desea construir y se pasa el resto del proyecto intentando aclarar esa imagen. La estimación del tiempo y el esfuerzo necesarios para desarrollar el software es también borrosa.

¿Cuánto cuesta desarrollar un sistema de facturación ?. Depende de cómo sea.

No se puede estimar con precisión el costo de un programa hasta que se comprendan con detalle cada una de las funciones que realizará el sistema. La incertidumbre sobre la naturaleza del producto aporta incertidumbre a la estimación.

4.2. FACTORES QUE INFLUYEN EN EL COSTO DEL SOFTWARE

La estimación de lo que costará el desarrollo de un software es una de las actividades de planeación que reviste especial importancia, ya que una de las características que debe tener un producto de software es que su costo sea adecuado, de lo contrario el proyecto puede fracasar.

Muchas organizaciones utilizan una serie de estimaciones de costos parciales antes de emitir el costo definitivo. Se hace una estimación durante el estudio preliminar del problema y se revisa durante el análisis de factibilidad del proyecto. Una estimación mejorada se presenta durante las especificaciones del software y la estimación final durante la revisión del diseño.

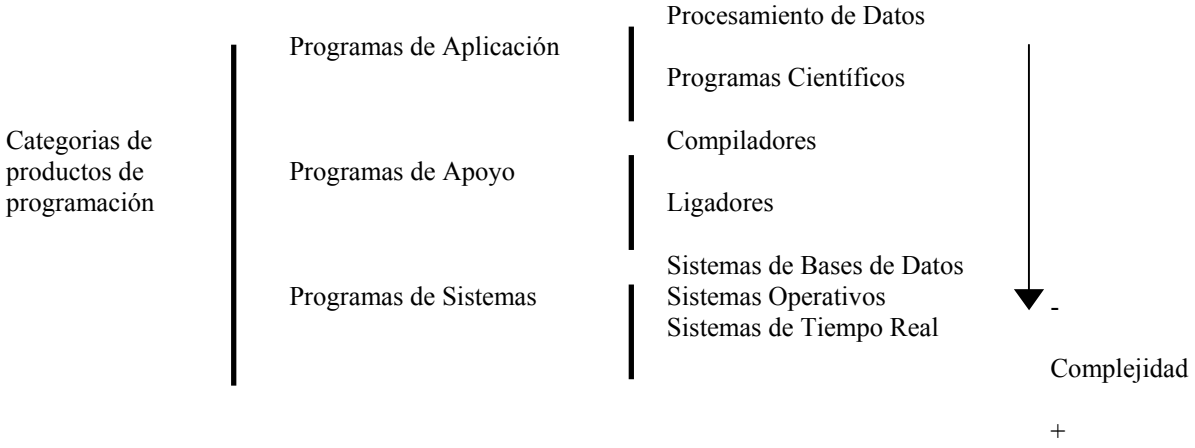
Los factores que principalmente afectan el costo del software son:

1. Capacidad del programador
2. Complejidad del producto
3. Tamaño del programa
4. Tiempo disponible
5. Confiabilidad requerida

CAPACIDAD DEL PROGRAMADOR

La producción de productos de programación son tareas laboriosas, por lo que la productividad es función directa de la capacidad individual. Los programadores que se muestran competentes en el procesamiento de datos, suelen no serlo en áreas científicas y de igual forma, un buen programador científico no es, forzosamente, un buen programador de sistemas. Por tanto, la falta de familiaridad con el área de aplicación puede implicar baja productividad y por ende mayor costo y esfuerzo.

COMPLEJIDAD DEL PRODUCTO



TAMAÑO DEL PRODUCTO

TIEMPO DISPONIBLE

El esfuerzo total del proyecto se relaciona con el calendario c Tamaño del Producto asignado para la terminación del proyecto. Los proyectos de programación requieren más esfuerzo si el tier arrollo se reduce.

NIVEL DE CONFIABILIDAD REQUERIDO

La confiabilidad puede expresarse en términos de exactitud, firmeza, cobertura y consistencia del código fuente. Existe un costo asosciado con el aumento del nivel de análisis, diseño, codificación y esfuerzo de verificación y validación que debe aportarse para asegurar alta confiabilidad.

El nivel de confiabilidad debe establecerse durante la fase de planeación al considerar el costo de las fallas del programa, en algunos casos, las fallas pueden causar al usuario pequeñas inconveniencias, mientras que en otros tipos de productos pueden generarse gran pérdida financiera e incluso poner una vida en peligro.

4.3 El Proceso de Estimación

El proceso para crear una planificación de desarrollo exacta consta de tres pasos :

1. *Estimar el tamaño del producto (en número de líneas de código fuente o puntos de función).*
   Algunos proyectos saltan directamente a estimar el plan, pero una estimación efectiva necesita estimar primero el tamaño del software para poder construirlo. Este paso es sin duda el más difícil intelectualmente.

2. *Estimar el esfuerzo (personas - mes).*
   Si tiene una estimación exacta del tamaño y los datos previos de la organización en proyectos similares, es fácil realizar la estimación del esfuerzo.

3. *Estimar el plan (meses).*
   Una vez que ha estimado el tamaño y el esfuerzo, estimar la duración del plan resulta casi trivial.

Estos tres pasos se pueden englobar dentro de un paso más general, un metapaso :

Dar un intervalo de estimación e ir refinando periódicamente ese intervalo, para ir aumentando la precisión a medida que avanza el proyecto.

4.3.1 Estilos de Presentación de Estimaciones

La forma en que inicialmente se presente la estimación influye enormemente en lo que suceda cuando posteriormente se necesite modificar esa estimación. La estimación de software generalmente conlleva un gran riesgo e incertidumbre, y una buena estimación capta ese riesgo e incertidumbre.

A continuación se muestran las técnicas para presentar la estimación de un plan de desarrollo.

Modificadores más-o-menos.

Utilice este estilo para indicar tanto la cantidad como la tendencia de la incertidumbre de la estimación. Una estimación de 6 meses, + ½ mes, - ½ mes, indica que la estimación es bastante precisa y que hay bastante probabilidad de conseguirla. Una estimación de 6 meses, + 3 meses, - 2 meses, indica que la estimación no es muy precisa y que es menos probable de conseguirla.

Rangos.

Uno de los problemas de la estimación con estilo más o menos es que solo se difunde la parte nominal de la estimación. Se eliminan los factores más o menos, cuyo resultado pierde información significativa. La alternativa es utilizar un intervalo de estimación, si la estimación es de 6 meses, + 3 mese, -1 mes, se podría presentar la estimación como de 5 a 9 meses.

Casos.

Con este estilo se presenta la estimación para el mejor y peor caso, así como el caso más real. Hay que estar preparado para explicar a los clientes lo que tendría que haber ocurrido para conseguir el “mejor caso” o caer en el “peor caso” ; los clientes desearán tener la información de las dos posibilidades.

Caso	Estimación
Mejor caso	1 abril
Caso más real	30 mayo
Peor caso	15 Julio

Factores de Confianza

Una de las preguntas que la gente se suele hacer sobre su plan de desarrollo es "¿ Qué probabilidad hay de terminar en esta fecha ?". Si se utiliza un enfoque con factores de confianza, se puede responder a la pregunta proporcionando una estimación como la siguiente :

Fecha de entrega	Probabilidad de entregar en la fecha planeada
1 de abril	5 %
1 de mayo	50%
1 junio	95%

4.4 METRICAS DEL SOFTWARE

4.4.1 Métricas orientadas al tamaño

La principal métrica para considerar el tamaño del producto de software es la cantidad de Líneas de Código Fuente (LCF) con las que fue construido.

Si una organización de software mantiene registros sencillos, se puede crear una tabla de datos orientados al tamaño, como la siguiente:

Proyecto	LDC	Esfuerzo	$ (miles)	Pag. Doc.	Errores	Defectos	Personas
Alfa	12,100	24	168	365	134	29	3
Beta	27,200	62	440	1,224	321	86	5
Gama	20,200	43	314	1,050	256	64	6

La tabla indica que para el proyecto Alfa se escribieron 12,100 LDC, con un esfuerzo de 24 personas-mes y un costo de $ 168 mil, se escribieron 365 Páginas de Documentación, se registraron 134 errores antes de que el SW se entregara al cliente y se encontraron 29 errores después de entregárselo al cliente durante el primer año de utilización.

Debe tenerse en cuenta que el esfuerzo y el costo registrado en la tabla incluye todas las actividades de ingeniería de software (Análisis, Diseño, Codificación y Prueba) y no solo la codificación.

Con los datos contenidos en la tabla se puede obtener un conjunto de métricas simples adicionales:

- No. De Errores por LDC 134 / 12,100 0.01 Errores / LDC
- No. De Defectos por LDC 29/12,100 0.002 Defect / LDC
- Costo por LDC 168,000 / 12,100 $ 13.88 / LDC
- LDC por persona-mes 12,100 / 24 904 LDC/p-m

La última métrica se puede usar como una medida de la productividad del equipo de desarrollo en el proyecto Alfa, es decir, nos dice que en promedio un programador escribía (producía) 904 líneas de código al mes.

Cuando se va a estimar el costo de un proyecto nuevo, se puede comparar sus características con las de los proyectos registrados en nuestra tabla histórica y de ahí obtener las LDC requeridas para un proyecto de ese tipo. Es lógico pensar que para el proyecto nuevo se requiera escribir un número APROXIMADO de LDC al que se requirió en un proyecto anterior parecido.

La métrica de LDC tiene ventajas y desventajas.

Ventajas:

- Es una métrica fácil de comprender.
- Muchos modelos, herramientas automáticas y literatura de estimación se basan en LDC.

Desventajas:

- Las LDC son dependientes del lenguaje. Escribir el mismo programa en lenguajes diferentes puede arrojar una diferencia en LCF bastante grande.
- Resulta difícil que el planificador estime las LCF a producirse mucho antes de que se complete el análisis y el diseño, más aún si no tiene datos históricos.

4.4.2 Métricas orientadas a la Función

Los puntos de función es una métrica con la cual se puede medir también el tamaño de un proyecto. Consiste en evaluar las características de cada funcionalidad que debe proporcionar el software.

Los puntos de función son más fáciles de determinar a partir de la especificación de requisitos que las LDC y proporcionan una medida más exacta del tamaño del programa.

El número de puntos de función en un programa se basa en el número y complejidad de los siguientes elementos:

Entradas.- Pantallas, formularios de captura, cuadros de diálogo, controles o mensajes, a través de los cuales el usuario final pueda añadir, borrar, modificar datos del programa.

Salidas.- Pantallas, informes, gráficas o mensajes que el programa genera para el usuario final. Esto incluye cualquier salida que tenga un formato diferente a otros tipos de salida.

Peticiones.- Normalmente entendido como Consultas que son una E/S interactiva, simple e inmediata. Las consultas recuperan datos de una base de datos y muestra solo el formato elemental.

Archivos Lógicos Internos.- Archivos que están bajo el control completo del programa a desarrollar. Puede ser un único archivo plano (de texto o ASCII) o de una única tabla en una base de datos relacional.

Archivos de Interfaz Externos.- Archivos controlados por otros programas, con los que el programa va a interactuar. Pueden ser cualquier archivos que salga o entre al programa.

Para calcular los puntos de función de un programa, se toma el número de entradas de menos complejidad y se multiplica por 3, el número de salidas de menor complejidad y se multiplica por 4, etc. La suma de estos números nos da el total de puntos de función.

Característica	Baja	Media	Alta
Número de Entradas	x 3	x 4	X 6
Número de Salidas	X 4	X 5	X 7
Peticiones	X 3	X 4	X 6
Archivos Internos	X 7	X 10	X 15
Interfaces Externas	X 5	X 7	X 10

Con los puntos de función obtenidos se puede comparar el tamaño del programa con proyectos anteriores y derivar el costo del nuevo proyecto, o bien se puede usar el método de estimación de primer orden de Jones, estudiado más adelante.

4.5 TECNICAS DE ESTIMACION DE COSTOS

JUICIO EXPERTO

La técnica más utilizada para la estimación de costos es el uso del juicio experto. El juicio experto se basa en la experiencia, en el conocimiento anterior y en el sentido comercial de uno o mas individuos dentro de la organización.

El experto, por ejemplo, puede hacer una estimación de costos razonando de la siguiente manera:

- El sistema que se va a desarrollar es muy similar a uno que se desarrollo anteriormente y que costó $ 8,000.00 y tardó 4 meses.

- Se puede reutilizar la base del proyecto anterior.

- Se utilizará el mismo equipo de cómputo y a muchos de los programadores que participaron en el proyecto anterior, por lo que la estimación se puede reducir en un 20 %.

- Mucho código y rutinas comunes se podrán reusar por lo que el esfuerzo se reduce otro 20%.

- Por lo tanto el nuevo proyecto puede ser un 20% más económico que el anterior.

Ventajas

1. Se obtiene una estimación en corto tiempo.

Desventajas

1. El experto puede confiarse y olvidar algunos factores importantes del nuevo proyecto, creyendo que es casi igual al anterior.
2. El experto puede no tener familiaridad con el área del proyecto.

Para compensar las desventajas se forma un grupo de expertos que en consenso determinen o estimen el costo. La desventaja es que algunos miembros del grupo pueden ser inocentes con respecto a la presencia de una alta autoridad o al dominio de un miembro del grupo con fuerte personalidad. También se pueden suscitar discusiones fuertes por no lograr un acuerdo entre los miembros del grupo.

Para evitar ahora estas nuevas desventajas existe una tecnica de estimacion conocida como tecnica DELFI.

TECNICA DELFI

Se desarrollo en la corporación Rand en 1948, con el fin de lograr un acuerdo de un grupo de expertos sin contar con los efectos negativos de las reuniones de grupos. La técnica se lleva a cabo de la siguiente manera:

1. Un coordinador proporciona a cada experto la documentaciuón con la Definición del Sistema y una papeleta para que escriba su estimación.

2. Cada experto estudia la definición y determina su estimación en forma anónima; los expertos pueden consultar al coordinador, pero no entre ellos.

3. El coordinador prepara y distribuye un resumen de las estimaciones efectuadas, incluyendo cualquier razonamiento extraño efectuado por alguno de los expertos.

4. Los expertos realizan una segunda ronda de estimaciones, otra vez anonimamente, utilizando los resultados de la estimación anterior. En los casos en que una estimación difiera mucho de las demás, se podrá solicitar que también en forma anónima el experto justifique su estimación.

5. El proceso se repite tantas veces como se juzgue necesario, impidiendo una discusion grupal durante el proceso.

Ejemplo de papeleta para la estimación:

Proyecto: Sistema Operativo Fecha: 6/6/84

Estimación en la 3a vuelta:
Su estimación es:

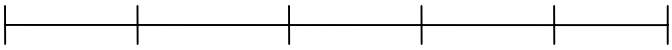
X X

Meses-Programador

0 20 40 60 80 100

Su estimación para la siguiente vuelta es: 35 PM

Razones para su estimación:
º Parece un sistema normal de control de procesos.
º Nuestra gente ha tenido gran experiencia en proyectos similares.
º No se espera que haya problema con este proyecto



METODO HISTORICO

Este método se basa en registros cuidadosos que se mantienen de esfuerzos de desarrollo previos. Los registros indican las características del programa o proyecto, la asignación de tareas, los requerimientos de tiempo del personal y cualquier acontecimiento o problema poco común. Cuando se proponen nuevos proyectos, se comparan con los registros que se llevan en el archivo y se iguala a lo planeado con trabajos anteriores para estimar el tiempo de desarrollo esperado.

MODELO DE COSTOS COCOMO

El modelo de costos COCOMO (Modelo Constructivo de Costos), se basa en el uso de ecuaciones que se utilizan de acuerdo a la complejidad del sistema a desarrollar:

Las ecuaciones son de la forma:

$$PM = a * (KDSI)^{b} ; \qquad TDEV = c * (PM)^{d}$$

donde KDSI es el numero de miles de lineas de codigo fuente esperadas.
PM es el esfuerzo en meses de programador
TDEV es el tiempo de desarrollo en meses

Programas de Aplicación:	$PM = 2.4 * (KDSI)^{1.05};$	$TDEV = 2.5 * (PM)^{0.38}$
Programas de Apoyo:	$PM = 3.0 * (KDSI)^{1.12;}$	$TDEV = 2.5 * (PM)^{0.35}$
Programas de Sistemas:	$PM = 3.6 * (KDSI)^{1.20;}$	$TDEV = 2.5 * (PM)^{0.32}$

Las ecuaciones proporcionan los valores nominales de la estimación de meses de programador (PM) y del calendario de desarrollo (TDEV) para cada unidad de trabajo, basándose en el número de instrucciones de codigo fuente entregadas (DSI) de cada unidad; después se utilizan factores multiplicadores (FM) para ajustar la estimación de acuerdo con los atributos del producto, de la computadora, del personal dedicado y del proyecto. Cada factor multiplicador se evalúa y califica con un valor, ese valor representa el peso que tiene dicho factor en el desarrollo del proyecto, luego se multiplican todos los factores para obtener el factor multiplicador global:

$$FM = fm_1 * fm_2 * fm_3 * ..... * fm_i$$

Las ecuaciones de calculo de esfuerzo aplicando los factores multiplicadores quedarían de la forma:

$$PM = a * (KDSI)^{b} * FM;$$

Tabla de Factores Multiplicadores de Esfuerzo en COCOMO:

Factor Multiplicador	Intervalo de Valores		
Atributos del Producto			
º Confiabilidad requerida	0.75	a	1.40
º Tamaño de la Base de Datos	0.94	a	1.16
º Complejidad del producto	0.70	a	1.65
Caracteristicas de la Computadora			
º Limitantes en tiempo de ejecución	1.00	a	1.66
º Limitantes en memoria principal	1.00	a	1.56

º Tiempo de entrega de programas	0.87	a	1.15
Caracteristicas del Personal			
º Capacidad de los analistas	1.46	a	0.71
º Capacidad de los programadores	1.42	a	0.70
º Experiencia en programas de aplicacion	1.29	a	0.82
º Experiencia en el lenguaje de programacion	1.14	a	0.95
Caracteristicas del proyecto			
º Uso de tecnicas modernas de programacion	1.24	a	0.82
º Uso de herramientas de programacion	1.24	a	0.83
º Tiempo requerido para el desarrollo	1.25	a	1.10

Es muy importante considerar este metodo con sus correspondientes reservas, ya que el estudio realizado para obtener las ecuaciones incorpora ciertas suposiciones importantes, además de que su uso esta basado en estimar primeramente el numero de lineas de código fuente (DSI) de que podrá constar el producto, estimación ciertamente dificil de efectuar.

Las suposiciones que el modelo COCOMO incorpora hace que las ecuaciones por ejemplo, para los programas de aplicacion sae usen solo en las siguientes situaciones:

1. Desde programas pequeños hasta medianos (de 2K a 32 KDSI).
2. En un área de aplicacion conocida
3. Con un sistema operativo y plataforma conocida.

Las estimaciones por esta técnica cubren desde el diseño hasta las pruebas de aceptación del programa.
Las estimaciones excluyen los costos de planeación, análisis, instalación y entrenamiento, así como los costos de secretarias, personal de limpieza y operadores del equipo de cómputo.

ECUACION DE PRIMER ORDEN DE JONES

- Carpers Jones propone una ecuación derivada del análisis de su base de datos de cientos de proyectos.
- Este es también un modelo empírico de estimación.
- Con la suma total de puntos de función , se puede realizar a partir de ellos un cálculo aproximado de la planificación.
- La ecuación de Jones tiene la forma:

$$TDEV = \#PF^{n}$$

Donde TDEV = Tiempo de planificación (meses)
#PF = No. de Puntos de Función
n = Exponente según la tabla siguiente

Clase de Software	Mejor Caso	Media (Nominal)	Peor Caso
Aplicación	0.39	0.42	0.45
Gestión	0.41	0.43	0.46
Sistemas	0.43	0.45	0.48

Ejemplo: Si #PF = 350 en un proyecto de software de Aplicación:

TDEV = $350^{0.42}$ = 12 meses aprox. Estimación Nominal

Si el equipo de desarrollo fuera sumamente organizado, están motivados, conocen y se apegan a las metodologías de ingeniería para el desarrollo, etc. entonces podría considerar lograr un menor tiempo, si aplica el Mejor Caso:

TDEV = $350^{0.39}$ = 10 meses aprox. Estimación Mejor Caso

Este no es el mejor método de estimación de la planificación, pero ayuda o obtener una aproximación, lo cual es mejor que hacerlo a ojo de buen cubero. Si cree que puede desarrollar un proyecto de software de Aplicación de los mismos 350 puntos de función en 7 meses, tendría que pensarlo muy bien. Como se vio anteriormente la planificación en el mejor de los casos es de 10 meses.

TECNICA ITERATIVA

- Se basa en la entrega evolutiva del software (por etapas o iteraciones).
- En cada etapa se construye un pequeño conjunto de funciones del software (miniproyecto), que implica análisis, diseño, codificación, pruebas, documentación y evaluación del cliente.
- La técnica permite determinar el número de iteraciones y su longitud necesaria para desarrollar el proyecto completo.

Consideraciones:

- Suponga que se dispone de desarrolladores que trabajan de tiempo completo en el proyecto (luego se considerará un factor de error).
- Las estimaciones son de los desarrolladores, no de los gerentes.

Procedimiento:

1.- Estimar el esfuerzo de cada función del software.
Asegurarse de que quien haga la estimación sea el desarrollador que posea el mayor conocimiento de la función dada. La estimación puede ser por Juicio Experto, sin embargo el razonamiento del desarrollador puede incluir un breve análisis de LCF o de Puntos de Función.

Ejemplo:

Funciones del SW	Esfuerzo (semanas-programador)
f_1 = Captura de datos generales de alumno	e_1 = 2.0
f_2 = Generar boleta de calificaciones	e_2 = 1.5
f_3 = Generar constancia de estudios	e_3 = 1.0
.	.
f_n	e_n

Esfuerzo Total E = Suma e_i = 50 semanas-programador

2.- Determine el número de programadores.
Utilizando los registros de proyectos anteriores, busque una aproximación con el Esfuerzo del proyecto actual y deduzca el número de programadores que convendría utilizar.

Ejemplo:
Si en un proyecto anterior de 38 semanas-programador requirió 4 programadores, puede ser razonable que el nuevo proyecto requiera de 5.

3.- Determine el factor de eficiencia de los programadores (productividad).
Aunque los desarrolladores sean de tiempo completo, no dedicarán el 100% de la jornada en trabajo productivo. Investigaciones de cómo usan su tiempo los programadores indican lo siguiente:

Escritura de programadores	13%	
Lectura de manuales y programas	16%	
Comunicaciones de trabajo	32%	
Otros:		
- Usos personales		13%
- Varios (ir al baño, prepararse un café, etc)	15%	
- Entrenamiento	6%	
- Correo	5%	

Esto demuestra que es muy importante considerar un factor de productividad en la estimación. Nuevamente puede recurrir a su base de datos de historia y derivar el nivel de productividad de sus equipos de desarrollo.

Ejemplo: Suponga una productividad del 50% para este ejemplo.

4.- Determine la longitud de la iteración.
Se busca una longitud de iteración fija para todo el proyecto, de modo que se pueda lograr un ritmo regular de entrega del proyecto. Cada iteración debe ser lo suficientemente larga para realizar varias funciones del software.

El lenguaje de programación es un factor para determinar la longitud de iteración, por ejemplo en un lenguaje conocido en el que los programadores estén familiarizados y tengan experiencia puede ser más fácil desarrollar las funciones del software en un tiempo más corto y viceversa.

Puede considerar longitudes de iteración de 2 a 8 semanas.

Ejemplo: Suponer para este ejemplo que se implementará en un lenguaje conocido, por lo que una longitud de

iteración de 3 semanas permitirá implementar de 2 a 4 funciones del software por iteración.

5.- Calcular el esfuerzo por iteración.
El equipo de desarrollo será capaz de desarrollar un determinado nivel de esfuerzo por el tiempo que dure cada iteración. Este esfuerzo depende del número de programadores, la longitud de la iteración y la productividad del equipo.

Esfuerzo por iteración (semanas-programador) = No. Programadores * Long. Iteración * Factor de Productividad

Ejemplo:
Esfuerzo por iteración = 5 * 3 * 0.5 = 7.5 semanas-programador

6.- Calcular el número de iteraciones.
Se divide el esfuerzo total entre el esfuerzo por iteración y el resultado se incrementa en uno (una iteración más como factor de seguridad).

No. Iteraciones = (Esfuerzo Total / Esfuerzo por Iteración) + 1

Ejemplo:
No. Iteraciones = (50 / 7.5) + 1 = 7.6 aprox. 8 iteraciones

7.- Calcular el tiempo de desarrollo.
Teniendo la longitud y el número de iteraciones es fácil determinar el tiempo de desarrollo.

Tdev = Long. Iteración * No. Iteraciones

Ejemplo:
Tdev = 3 * 8 = 24 semanas = 6 meses.

8.- Considerar un factor de contingencia.
Agregue un factor del 10% al 20% del tiempo de construcción, dependiendo de lo arriesgada que parezca la situación. Debe planificar la terminación del proyecto sin considerar el tiempo de contingencia, pero comprométase a entregar al final del tiempo de contingencia.

Ejemplo:
Considerando el factor 40-20-40, es decir, el 40% de esfuerzo es Analisis-Diseño, 20% Codificación y 40% Pruebas, calculamos que el tiempo posible de construcción (codificación) es el 20% de las 24 semanas totales.

Tcodif = 0.2 * 24 = 4.8 aprox. 5 semanas

Suponiendo que nuestro factor de contingencia es del 15% entonces el tiempo de contingencia será

Tconting = 0.15 * 5 = 0.75 semanas o aprox. 4 dias.

Por lo tanto el plan debe ser formulado para desarrollar el producto en 24 semanas, pero el compromiso de entrega final sería para 24.75 semanas, 4 días más.

4.6 CONSEJOS SOBRE ESTIMACIONES

1. Evite estimaciones improvisadas (ojo de buen cubero).
2. Use datos de proyectos anteriores (método histórico).
3. Las estimaciones las debe hacer el desarrollador, no el directivo.
4. Estime por concenso (técnica delfi).
5. Evite estimar el proyecto como un todo.
6. Estime por descomposición, es decir, estime cada función o requisito del software individualmente. Luego la suma de ellos será la estimación del proyecto total.
7. Utilice diferentes técnicas de estimación, compare los resultados y resuelva las diferencias.
8. Si es posible use una herramienta automática de estimación. Con esto tendrá otro punto de comparación.
9. Refine sus estimaciones a medida que conozca más detalles del proyecto.
10. Emita una estimación preliminar después de revisar la Definición del Sistema. Refinela durante el Análisis y de una estimación definitiva después de revisar el Diseño. Si esto no es posible, trate de emitir su estimación definitiva al terminar el Análisis. Si aún esto no es posible, deberá formularla solo con la Definición del Sistema pero considere un factor de seguridad adecuado.
11. Presente sus estimaciones usando rangos o casos (mejor, más probable, peor).
12. Defienda sus estimaciones.
    Finalmente Ud. y su equipo son los que conocen el proyecto y lo desarrollarán. En lo posible evite que un directivo le imponga una estimación que no esté justificada, para esto debe prepararse para defender sus estimaciones.
13. Negocíe con el cliente.

Muchas veces elaborar la Definición del Sistema puede implicar varias semanas o meses de trabajo, por consiguiente su primera estimación tardará lo mismo. Si existe el riesgo de que el cliente diga no a su presupuestación, negocie desde el principio un costo por la actividad de definición o análisis del sistema, aclare al cliente que él recibirá como producto de este trabajo un manual que incluye los documentos de Definición o Análisis del Sistema.

PROCEDIMIENTO SUGERIDO PARA LA ESTIMACION DE COSTOS

1. Identificar todos los sub sistemas y los módulos del producto.
2. Estimar el tamaño o magnitud de cada módulo y luego del sistema en total.
3. Especificar los factores multiplicadores de esfuerzo para cada módulo, tal como la complejidad del producto, la capacidad de los programadores, la experiencia en el lenguaje de programación.
4. Estimar el tiempo de desarrollo de cada modulo y del sistema en total.
5. Determinar la cantidad de personal que integrará el equipo de programación.
6. Calcular el costo de desarrollo tomando en cuenta el tiempo, cantidad de personal y sueldo del personal.
7. Sumar los otros costos por planeación y análisis que no se hayan incluido antes.

4.7 Calendarización

4.7.1 Fundamentos

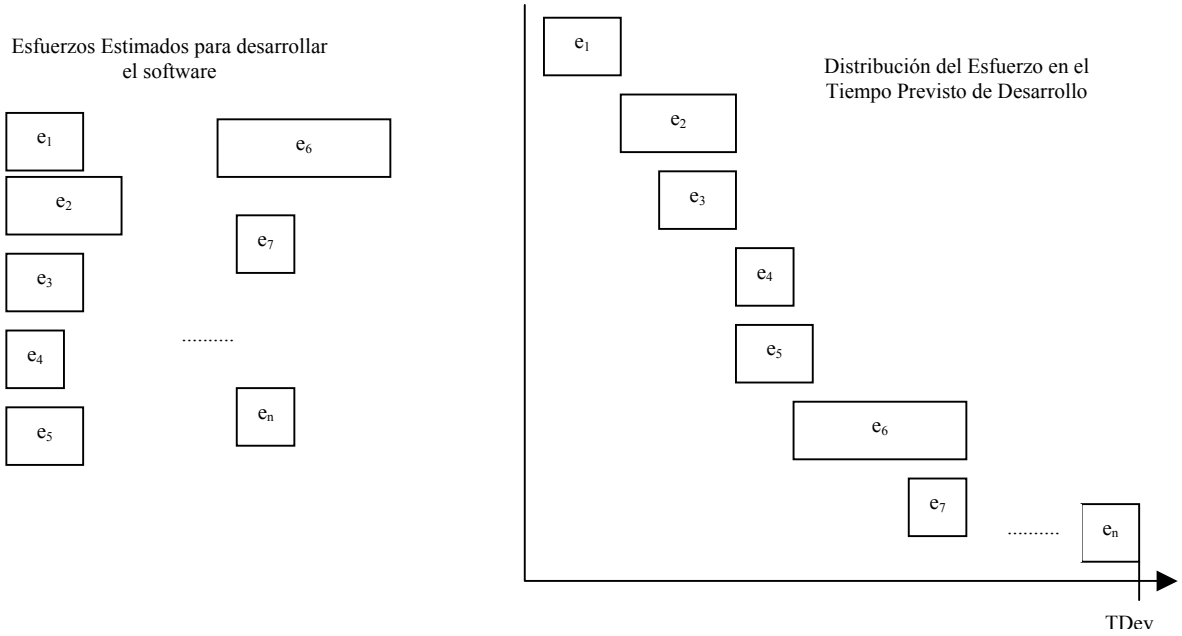
La realidad de un proyecto técnico, tal como uno de software, es que hay que realizar cientos de tareas pequeñas en un orden determinado antes de poder alcanzar la meta final. Las tareas están interrelacionadas en una secuencia lógica en el sentido de que algunas de ellas no pueden empezar hasta que otras se hayan terminado.

Algunas tareas se encontrarán en el camino principal (ruta crítica), de manera que si una de estas tareas críticas se retrasa, la fecha de terminación del proyecto se pone en peligro.

El administrador o líder del proyecto debe definir todas las tareas del proyecto, identificar las que son críticas y hacerles un seguimiento para asegurarse de que un retraso se pueda reconocer de inmediato. Para conseguir esto el administrador debe hacer un plan calendarizado para supervisar y controlar el proyecto.

Elaborar un plan calendarizado de un proyecto de software es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto, asignando el esfuerzo a las tareas específicas de la ingeniería de software.

La fase de planeación del proyecto debe generar un diagrama del proyecto, en forma de calendario o agenda de actividades; en el que a través del tiempo se pueda observar como se ejecutarán las actividades de desarrollo.



Las técnicas de calendarización más utilizadas no solo en proyectos de software, sino en proyectos de cualquier índole son:

1. Método PERT
2. Gráfica GANTT

4.7.2 Método PERT

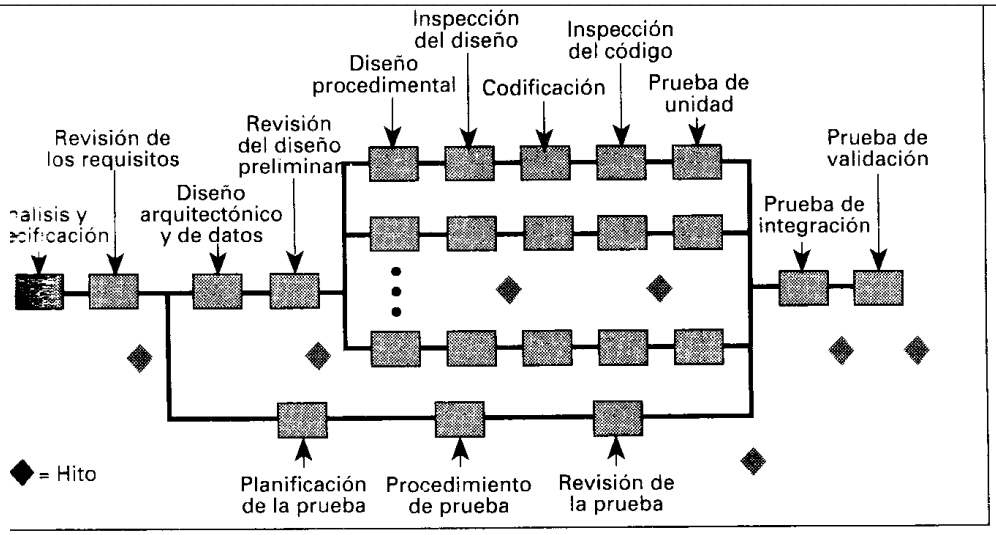
Las tareas y subtareas individuales tienen interdependencias basadas en su secuencia. Además cuando hay más de una persona involucrada en el proyecto, es posible que las actividades de desarrollo y las tareas se realicen en paralelo.

Para representar las tareas, su secuencia y dependencias se utiliza un diagrama llamado *red de tareas*.

El método PERT permite determinar:

- El tiempo de inicio más temprano y tardío de cada tarea.
- El tiempo de terminación más temprano y tardío de cada tarea.
- La ruta crítica – la cadena de tareas que determinan la duración del proyecto.
- El diagrama de tiempo (básicamente una gráfica Gantt).

El punto de partida y base para la aplicación del método es establecer la red de tareas.
La siguiente figura muestra una red de tareas esquemática para un proyecto típico de Ingeniería de Software con varias personas.



4.7.3 Gráfica Gantt

La gráfica de Gantt es un diagrama que muestra gráficamente la relación del tiempo entre los pasos en un proyecto. Cada tarea en un proyecto es representada por una barra horizontal situada en el diagrama en el periodo de tiempo dentro del cual debe ejecutarse. También muestra la secuencia del flujo de las actividades, así como las que se estén llevando a cabo al mismo tiempo.

La gráfica de Gantt es también un subproducto del método PERT (como se menciono anteriormente), sin embargo esto no obliga a tener que desarrollar el método PERT para derivar la gráfica Gantt.

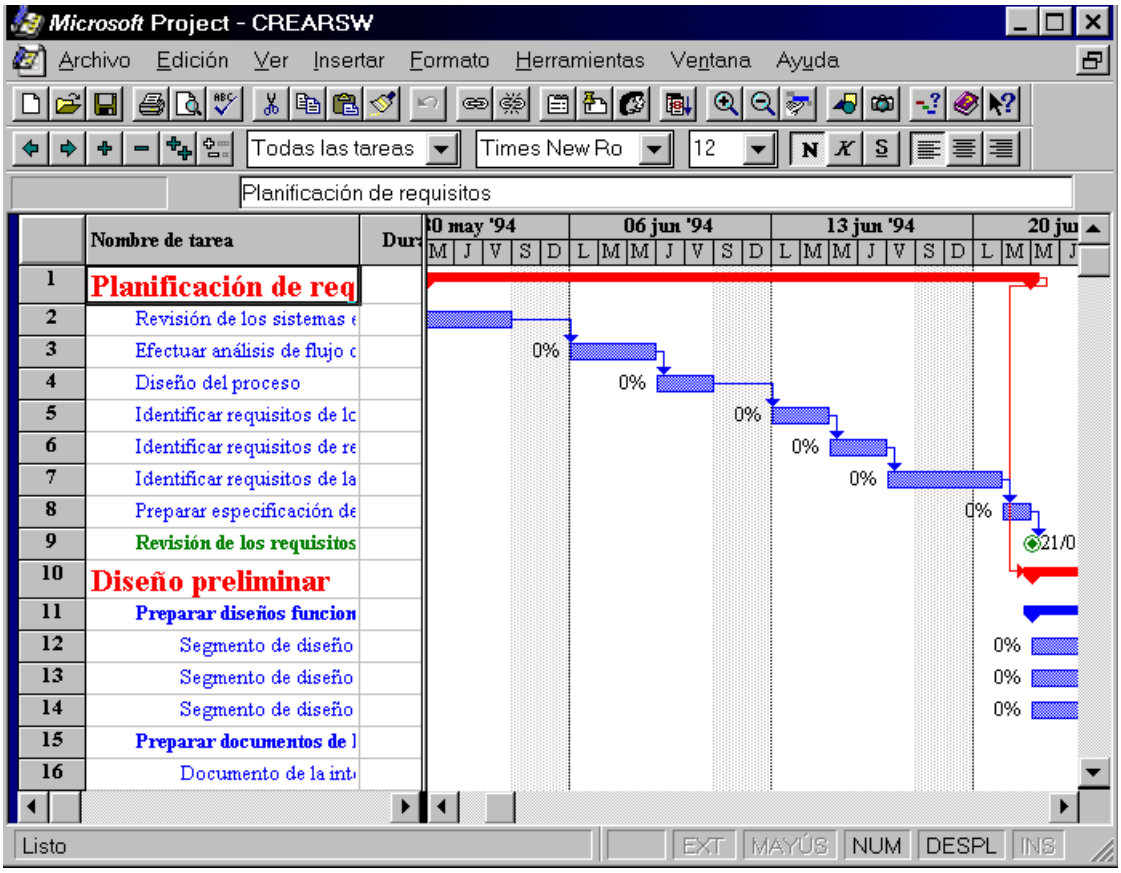
4.7.4 Herramientas Automáticas de Calendarización

Existen en la actualidad diversas herramientas automáticas de calendarización y seguimiento de proyectos, herramientas implementadas en software, tales como Microsoft Project, Turbo Project, etc.

Estas herramientas permiten introducir la red de tareas del proyecto, la fecha de incio y duración de cada tarea.
En consecuencia la herramienta crea automáticamente diferentes vistas de los datos alimentados, típicamente una gráfica de Gantt y el diagrama PERT señalando la ruta crítica.

Además se puede alimentar la asignación de personal a cada una de las tareas y el costo asociado con el uso de cada recurso por unidad de tiempo, los avances y costos en la ejecución real del proyecto.

La herramienta genera informes útiles para el administrador del proyecto en los que puede apreciar entre otras cosas el presupuesto del proyecto, recursos sobreasignados, recursos disponibles, :



4.8 ESPECIFICACION DEL PLAN DEL PROYECTO

El plan del proyecto de software se produce como resultado de la etapa de planificación. El plan es un documento relativamente breve, que debe:

1. Comunicar el ambito y los recursos a los administradores de software, personal técnico y al cliente
2. Definir el costo y la agenda de trabajo.
3. Proporcionar un enfoque global del desarrollo de software para toda la gente involucrada en el proyecto.

El plan ha de establecer de forma general el qué y de forma especifica cuánto y por cuánto tiempo.

FORMATO DEL PLAN DEL PROYECTO

Seccion 1:	Modelo del ciclo de vida
Seccion 2:	Estructura organizacional
Seccion 3:	Requisitos preliminares de personal y recursos.
Seccion 4:	Programación preliminar del desarrollo y grafica Gantt.
Seccion 5:	Estimado preliminar de costos.
Seccion 6:	Mecanismos de supervision y control del proyecto.
Seccion 7:	Herramientas y tecnicas que se emplearán.
Seccion 8:	Lenguajes de Programación.
Seccion 9:	Requisitos de Prueba.
Seccion 10:	Documentos de apoyo necesarios.
Seccion 11:	Formas de demostracion y entrega.
Seccion 12:	Programacion de entrenamiento.
Seccion 13:	Plan de instalacion.
Seccion 14:	Consideraciones de mantenimiento.
Seccion 15:	Metodo y tiempo de la entrega final.
Seccion 16:	Metodo y tiempo de pago.
Sección 17:	Fuentes de Información.

-oOo-