

10-Administración de Proyectos.

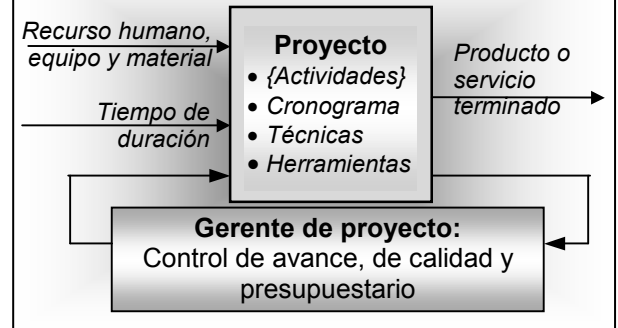
Por: Ing. Carlos Ernesto García, M.Sc
Septiembre 2013.

Una de las herramientas más importantes de que dispone el Gerente Informático para desarrollar las actividades de su UI es el proyecto; sobre todo las de desarrollo de SI y desarrollo de software; aunque también tiene utilidad en actividades de mantenimiento de SI, mantenimiento de software, planes de contingencia y conversión de SI.

Qué es un proyecto. Como se muestra en la Fig. 10.1, un proyecto es un conjunto de actividades, técnicas y herramientas interrelacionadas para lograr un objetivo concreto en un tiempo dado; como construir un edificio, desarrollar un software, auditar un SI u organizar un evento deportivo.

Al objetivo de todo proyecto siempre están asociados tres factores: tiempo de duración, recurso disponible y calidad. El tiempo de duración y el recurso disponible son las entradas al proyecto más importantes. La calidad expresa una medida en que deben cumplirse los objetivos para que el proyecto sea considerado exitoso. Los 3 factores se constituyen desde el inicio en restricciones a la ejecución del proyecto: el objetivo del proyecto debe alcanzarse con el recurso disponible, en el tiempo especificado y con la calidad especificada.

Fig. 10.1. El proyecto.



Administración de proyectos. La administración de un proyecto conlleva cinco macroactividades: formulación, organización, ejecución, control e implementación.

- Formulación.** Es la función de planificación del proyecto y es previa al nombramiento del gerente del proyecto. Consiste en definir los objetivos del proyecto y el tiempo de duración; y en función de éstos, los siguientes elementos: fecha de inicio, actividades [duración y restricciones de ejecución de cada una], relación de secuencia entre actividades, ruta crítica, recursos por actividad [recurso humano, equipo y materiales], calendario de ejecución de las actividades, presupuesto monetario del proyecto y flujo de efectivo semanal, quincenal o mensual.
- Organización.** Ocurre bajo la responsabilidad total del gerente del proyecto. Consiste en definir la organización interna formal para la ejecución del proyecto. Comprende el nombramiento del gerente o director de proyecto.
- Ejecución.** Ocurre bajo la responsabilidad total del gerente del proyecto. Consiste en ejecutar las actividades del proyecto de acuerdo a la secuencia, calendario y recursos predefinidos en la formulación.
- Control.** Ocurre bajo la responsabilidad total del gerente del proyecto. Consiste en monitorear de manera continua la ejecución del proyecto para medir su avance con respecto a lo programado, la calidad del proceso de ejecución, la ejecución del presupuesto y el flujo de efectivo.
- Entrega.** Esta macroactividad es la última del proyecto, y consiste en entregar al *propietario* el producto o servicio terminado, una vez alcanzados los objetivos del proyecto, a su entera satisfacción.

Proyectos de desarrollo de SI. En proyectos de desarrollo de SI se aplican las mismas 5 macroactividades que se aplican en la gestión de cualquier otro proyecto. Sin embargo, hay particularidades que se deben considerar:

- Ciclo de vida del proyecto.** La Gerencia Informática debe establecer un estándar que especifique el ciclo de vida que debe adoptarse para este tipo de proyecto, clásicamente dividido en 6 fases, [*método de desarrollo en cascada*]: análisis contextual, formulación de requerimientos, diseño, plan de prueba, plan de implementación y documentación. Cada fase del proyecto es una *macroactividad* con relación al proyecto y sirve como punto de partida para definir las actividades específicas de un proyecto en particular dentro de cada fase.

Se ha asumido el modelo de *ciclo de vida en cascada* sólo para ilustrar ciertos conceptos importantes. Otros modelos de ciclo de vida son: *modelo evolutivo o en espiral*, *modelo por incrementos* y el *modelo unificado*. Todos ellos dividen de una u otra manera la vida del proyecto en varios ciclos de menor duración, aplicando a cada ciclo el modelo en cascada.

- Duración del proyecto.** La duración del proyecto de desarrollo un SI suele ser una restricción de desarrollo expresada como "*el proyecto no debe durar más de x tiempo*". Dada una duración requerida D para un proyecto, es posible hacer una primera estimación de la duración de cada fase como una fracción de D, como se muestra en el cuadro 10.1.
- Duración de actividades.** En todo proyecto, la estimación de la duración de cada actividad es punto clave para determinar la duración total del proyecto. Las duraciones más

Cuadro 10.1. Estimación duración de proyectos.	
Fase del proyecto	Duración como % de D
Análisis contextual	10 a 15%
Formulación de requerim.	10 a 15%
Diseño	10 a 30%
Plan de prueba	10 a 30%
Plan implementación	10 a 20%
Documentación	10 a 15%

difíciles de estimar son las relacionadas con actividades de análisis y de diseño.. Lo recomendable es que la estimación de tales duraciones se base en métricas generadas por la UI con base en experiencias previas en ambientes similares. Con base en estas experiencias, resulta estadísticamente aceptable trabajar con un *tiempo esperado* (*TE*) de duración para cada actividad; partiendo de una estimación de un *tiempo pesimista* (*TP*), un *tiempo normal* (*TN*) y un *tiempo optimista* (*TO*) de duración para la misma actividad.

Tiempo pesimista es la duración estimada si la actividad se llevara a cabo en *las condiciones más adversas*; tiempo optimista, la duración al realizarla en *las mejores condiciones que puedan ocurrir*; y tiempo normal, la duración si se llevara a cabo en *condiciones típicas*. El tiempo esperado resulta ser entonces:

$$TE = (TP + 4TN + TO) / 6$$

El tiempo esperado calculado de esta manera supone que la duración de una actividad es una variable aleatoria con distribución de probabilidad normal. A partir de este supuesto puede afirmarse que la probabilidad de que la duración de la actividad correspondiente sea *TE* es ligeramente mayor que un 95%.

d. Programación de actividades. Calendarizar las actividades de un proyecto de desarrollo de un SI se facilita si se parte de la calendarización de las fases del ciclo de vida del proyecto, las cuales no se desarrollan en serie normalmente, una tras otra, como se puede asumir erróneamente en principio. Más bien se traslapan entre sí en función de las características del proyecto, del modelo de proceso de desarrollo adoptado y otros factores como: tiempo requerido, acervo de recursos, estilo dirección del gerente del proyecto y nivel de complejidad.

Para ilustrar, la figura 10.2 presenta la programación de fases de desarrollo del ciclo de vida un proyecto X. Las primeras tres fases duran en conjunto 60% del tiempo total, traslapándose entre sí 10%. La fase documentación se desarrolla a lo largo de todo el proyecto. Cada proyecto real tiene su propia configuración de porcentajes.

Figura 10.2. Programa de las fases de un proyecto.

Fases	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Análisis context.										
Requerimientos										
Diseño										
Plan de prueba										
Plan Implement.										
Documentación										

B. Causas de éxito o fracaso en desarrollo de software.

Caper Jones¹ analizó las causas del éxito o fracaso de proyectos de desarrollo de software pequeños, medianos y grandes. La unidad de tamaño que emplea para el software es el *punto funcional* (PF).

Punto funcional. Es una medida propuesta para medir la magnitud de un programa o del esfuerzo de programación de un proyecto. Consiste en un conjunto de instrucciones que realiza una función elemental dentro de un programa. Es equivalente a un procedimiento, función o método en un programa modular. Su tamaño no debería exceder 100 líneas de código (LDC) fuente según Jones. Para este curso el estándar será 50 LDC fuente incluyendo comentarios.

Tamaño de proyecto. Según Jones, un proyecto de desarrollo de software grande tiene más de 5,000 PF, equivalentes a unas 500,000 LDC fuente en un lenguaje orientado a procedimientos como Cobol o Fortran. Un proyecto mediano tiene entre 1,000 y 5,000 PF; y uno pequeño tiene entre 100 y 1,000. Un software de menos de 100 PF puede considerarse un proyecto muy simple, de muy poca complejidad. En términos de duración, un proyecto grande se programa para 24 meses en promedio, uno mediano para 18, uno pequeño para 12 y uno simple para 6 meses o menos.

En proyectos grandes casi uno de cada 4 fallan o son cancelados, (cuadro 10.2); y de los que logran terminar, dos de cada tres exhiben un atraso de al menos seis meses.

Cuadro 10.2. Porcentaje de proyectos por tamaño y resultado final.

Resultado final del proyecto	Tamaño del proyecto (PF)			
	Simple <100	Pequeño 100/1,000	Mediano 1,000/5,000	Grande >5,000
Cancelado	3	7	13	24
Terminó tarde, más de 12 meses	1	10	12	18
Terminó tarde, más de 6 meses	9	24	35	37
Terminó aproximadamente a tiempo	72	53	37	20
Terminó antes de lo esperado	15	6	3	1
	100	100	100	100

Lo que no se aprecia en este cuadro es que ya en el primer año el 100% de tales proyectos exhibe costos arriba de lo presupuestado, muestra escasa confiabilidad y tiene problemas de calidad. Algunos pocos proyectos grandes, uno de cada 100, (cuadro 10.2) logran terminar a tiempo, dentro del presupuesto y prácticamente sin problemas de calidad.

¹ Capers Jones, "Patterns of Large Software Systems: Failure and Success", *Computer*, Vol.28, No.3, March 1995, pp. 86/87.

Tal como es de esperar, a medida que el proyecto decrece en magnitud los porcentajes de éxito aumentan, como se aprecia en los cuadros 10.2 y 10.3. Sin embargo, es de notar que aún los proyectos pequeños tienden a concluir con retraso significativo con respecto a la duración programada.

¿Porqué en proyectos grandes pocos tienen éxito y la mayoría fracasa? Caper Jones responde a esta pregunta citando diez factores de éxito y diez factores de fracaso que él ha observado en su estudio. (Ver cuadro 10.4).

Cuadro 10.3. Duración real vs. programada según tamaño.

Duración (meses)	Tamaño del proyecto (PF)			
	Simple <100	Pequeño 100/1,000	Mediano 1,000/5,000	Grande >5,000
Programada	6	12	18	24
Real	6	16	24	36
Atraso	0	4	6	12

Cuadro 10.4. Factores de éxito o fracaso en el desarrollo de un proyecto.

FRACASO	EXITO	Tipo
1. Sin datos históricos para métrica de software	Medición precisa del software	Administración de proyectos
2. Herramientas de estimación manuales	Herramientas de estimación automáticas	
3. Sin herramientas de planificación	Uso continuo de herramientas de planificación	
4. Sin monitoreo del avance	Control de avance continuo y formal	
5. Sin chequeo de especificaciones de diseño	Revisión formal de especificaciones de diseño	
6. Sin chequeo de programas fuente	Chequeo formal de programas fuente	
7. Uso de inexpertos para tareas críticas	Uso de especialistas para tareas críticas	Administración general
8. Diseño y requerimientos informales	Diseño y requerimientos formales	
9. Sin herramientas de control	Continuo uso de herramientas de control	
10. Cambio de requerimientos mayor del 35%	Cambios de requerimientos menor del 15%	Fuera de control

Los primeros seis factores se asocian a con aspectos específicos en el dominio de la administración de proyectos. Los tres siguientes están relacionados de alguna manera con la práctica de la administración en general. El décimo factor está fuera del control del proyecto, pues aunque los cambios en los requerimientos pueden anticiparse, rara vez pueden ser controlados o revertidos una vez que aparecen.

Desde el punto de vista de la ingeniería de software, la razón más común de desastres en proyectos de desarrollo de software es un pobre control de calidad. Encontrar y reparar fallas en el desarrollo de software es el aspecto más caro y el que más tiempo consume, especialmente en proyectos grandes.

C. Guía para Estimaciones de Proyectos de Software [Metodología Metzger²].

Un *estimado* es un valor de medición acerca de un elemento de un proyecto, generado a partir del *juicio* de una persona. Los estimados de la duración y el costo de un proyecto se expresan en términos de horas-hombre, horas-máquina, días-hábiles, etc. El objetivo final de las estimaciones es enunciar formalmente la cantidad de recursos requeridos en el proyecto y cómo serán utilizarlos. Cuando los estimados se expresan en términos de dinero y se calendarizan, arribamos a un estimado que se denomina *presupuesto del proyecto*.

Estimar los costos de un proyecto de desarrollo del software de un SI o de otro tipo es una tarea complicada para quien lo formula. No hay una receta universal para hacer las estimaciones de forma totalmente mecanizada; pero existen métodos recomendados para enfrentar el problema con cierto grado de razonabilidad. La metodología de Metzger sirve para estimaciones relativas a un proyecto de desarrollo de software, comprende 12 pasos:

Paso 1. Elaborar un *macrodiseño* del software. El nivel de detalle queda a juicio de quien hará las estimaciones. El resultado puede ser sólo una o varias páginas de diagramas. Puede tomar una sola tarde o varios días. Lo importante es que el macrodiseño sea suficiente para percibir claramente la magnitud real del problema a resolver y los problemas técnicos más importantes que habrá que enfrentar.

Paso 2. Estimar el *tamaño del software*. Se refiere a los programas *finales* que se entregarán como producto del proyecto, el tamaño expresado en PF. En anexo se presenta un método simplificado de conteo para hacer esta estimación con PFs: el método IFPUG³. Si se tienen definidos módulos del software en el macrodiseño, las estimaciones se hacen al nivel de módulos y luego se suman para totalizar. A mayor detalle en el macrodiseño corresponden estimaciones más precisas.

Paso 3. Estimar la *fuerza de programación*. Se especifica en *meses-programador*. Incluye el esfuerzo para todas las fases del ciclo de vida; no sólo diseño y programación. El estimado se hace en función del tamaño total del software y un estándar de rendimiento, como *PF/mes-hombre* totalmente terminadas. El IFPUG estima que el rendimiento promedio general es 11.5 PF/Mes-hombre.

² Metzger, Phillip W.; Managing a Programming Project, Prentice-Hall, New Jersey, 1973.

³ International Function Point Users' Group. Originalmente definido por Allan Albrecht, en 1979

Cada organización tiene su propio estándar de rendimiento sobre PF/mes-hombre y no necesariamente es lineal. Ejemplo: el std. de IBM es 10 PF/mes-hombre para proyectos de 50 PF; pero es 3.6 PF/mes-hombre para 500 PF. Pese a que existen publicaciones sobre estos estándares de rendimiento, no hay mejores que aquéllos de la organización que va a desarrollar el software. No existen dos ambientes iguales. Por ello, al considerar los estándares de otras fuentes, siempre es recomendable cuestionarlos; aunque sin tomar en cuenta lo que algunos pesimistas expresan: “En resumidas cuentas, estimar la carga de trabajo que es necesario llevar a cabo para la construcción de un software es una tarea que normalmente debe fracasar”.

Paso 4. Estimar la fuerza de soporte. Se trata de recurso humano técnico (ingenieros, técnicos, etc.) que darán apoyo directo al personal que constituye la fuerza de programación en diversas actividades. En el cuadro 10.5 se presenta una lista rubros potenciales.

Paso 5. Estimar el requerimiento de equipo. Equipo para desarrollar el software, para programación, para pruebas, para control, para la base de datos, etc. El cuadro 10.5 presenta una lista rubros potenciales.

Paso 6. Estimar otros recursos. Otros recursos necesarios para desarrollar el software podrían ser, por ejemplo, operadores de equipo, dibujantes, consultores, diseñadores gráficos, mantenimiento de hardware y capacitación.. En el cuadro 10.5 se presenta una lista rubros potenciales.

Paso 7. Convertir todo a dinero. Aplicar salarios promedio en la organización por cada clase de recurso humano considerada. El resto de recursos deben ser a precio de mercado. El resultado es el *presupuesto no ajustado*.

Paso 8. Ajustar estimaciones por factores contingenciales. Estos son eventos que se tiene la certeza de que van a ocurrir en alguna medida y que no han sido tomados en cuenta en las estimaciones como: vacaciones, permisos, renunciaciones y despidos de personal; prestaciones e incentivos; fallas de equipo; interrupciones por mantenimiento; y primas de seguro. Cualquier estimación sujeta a variación por este tipo de factor debe ser ajustada. Por ejemplo, se puede haber concluido que el proyecto requiere 30 meses-programador; pero los programadores se enferman, toman vacaciones, renuncian, son despedidos, etc. Estos factores pueden reducir fácilmente en un 20% la efectividad de la fuerza de programación estimada. Los factores contingenciales tomados en cuenta y la manera en que han afectado las estimaciones iniciales deben documentarse, para evitar que sean considerados de nuevo en el futuro. Se genera así un segundo presupuesto: *presupuesto ajustado por factores contingenciales*.

Paso 9. Ajustar estimaciones por factores ponderables. Son factores propios del medioambiente en que se desarrollará el proyecto que pueden o no ocurrir; pero que si ocurren que pueden afectar significativamente los costos y duraciones estimadas en una medida que hay que ponderar. Metzger presenta una lista de ellas, a manera de sugerencia, que se reproduce en el Cuadro 10.6. Chequear cada factor en la lista que se considere aplicable al proyecto. Para cada ítem chequeado, verificar si ya ha sido tomado en cuenta en los estimados ya hechos. De no ser así, decidir si el efecto del factor es tan relevante como para ajustar los estimados. Un factor puede afectar varios estimados; pero los más relevantes son los que afectan los estimados de recurso humano, duración y tiempo calendario.

Un cheque en un factor indica que algunos estimados deben *incrementarse*. Un factor no chequeado significa *no cambiar* los estimados y en algunos casos *disminuirlos*. Pero debe tenerse en cuenta que en el mundo de desarrollo de software rara vez un estimado está sobrestimado. El impacto de cada factor chequeado suele ser desde un 5% hasta un 15% de incremento del valor estimado. Al igual que los factores contingenciales, es importante documentar los factores ponderables aplicados y la manera en que han afectado a los estimados. Se genera así un tercer presupuesto: *presupuesto ajustado por factores ponderables*.

Paso 10. Agregar los costos indirectos. Los costos hasta ahora estimados son los llamados *costos directos*. Son costos que agregan *directamente* valor al producto a medida que se va desarrollando el sistema. Pero hay otros costos, llamados *costos indirectos*, en que debe incurrirse durante el desarrollo del proyecto: administración, seguridad, energía, teléfono, soporte administrativo, impuestos, multas y utilidades, entre otros. El cuadro 10.5 presenta una lista rubros potenciales. Como resultado se obtiene el *presupuesto final* del costo del proyecto.

Paso 11. Documentar los supuestos. Todo estimado está basado en supuestos. No se debe entregar ni recibir de *nadie* un estimado sin la documentación de los supuestos correspondientes. Un estimado puede ser totalmente inútil sin la documentación de esos supuestos o si está basado en supuestos erróneos. Por ejemplo, puede haber supuestos imposibles de cumplir o que no son del agrado del cliente o usuario.

Paso 12. Actualizar estimaciones. A medida que el proyecto avanza se obtienen mejores indicadores de la dificultad del proyecto, los cuales deben usarse para ajustar las estimaciones preliminares si es necesario. Un momento apropiado para evaluar la necesidad de hacer reestimaciones es al final de la fase de diseño. En ese momento se tiene el diseño completo y pormenorizado del sistema. Como regla general, todo proyecto que dura más de un mes está sujeto a cambios. Por cada cambio significativo deben hacerse las re-estimaciones que sean necesarias para ajustar el presupuesto. Los estimados siempre son imperfectos y normalmente se afinan a medida que avanza el proyecto.

Cuadro 10.5. Rubros de costo potenciales para desarrollo de software.

<p>Fuerza de soporte</p> <ul style="list-style-type: none"> <input type="checkbox"/> Director del proyecto <input type="checkbox"/> Auxiliares de requerimientos <input type="checkbox"/> Auxiliares de análisis <input type="checkbox"/> Auxiliares de diseño <input type="checkbox"/> Auxiliares de programación <input type="checkbox"/> Auxiliares de documentación <input type="checkbox"/> Auxiliares de pruebas <input type="checkbox"/> Auxiliares de comunicación con usuarios <input type="checkbox"/> Analistas de campo <input type="checkbox"/> Técnicos de O&M <input type="checkbox"/> Entrevistadores <input type="checkbox"/> Coordinadores <input type="checkbox"/> Codificadores <input type="checkbox"/> Digitadores <input type="checkbox"/> Etc. <p>Equipo</p> <ul style="list-style-type: none"> <input type="checkbox"/> Computadoras mainframe <input type="checkbox"/> Computadoras desktop <input type="checkbox"/> Líneas de comunicación <input type="checkbox"/> Estaciones de trabajo <input type="checkbox"/> Computadoras móviles <input type="checkbox"/> Impresores <input type="checkbox"/> Módem <input type="checkbox"/> Switches y HUB's <input type="checkbox"/> Discos magnéticos externos <input type="checkbox"/> Reproductores de documentos <input type="checkbox"/> Dispositivos de comunicación inalámbrica <p>Licencias de software</p> <ul style="list-style-type: none"> <input type="checkbox"/> Sistemas operativos <input type="checkbox"/> Compiladores <input type="checkbox"/> Herramientas de productividad <ul style="list-style-type: none"> <input type="checkbox"/> Editores <input type="checkbox"/> Procesadores de palabras <input type="checkbox"/> Procesadores gráficos <input type="checkbox"/> Generadores de reportes <input type="checkbox"/> Manejadores de bases de datos <input type="checkbox"/> Utilitarios 	<p>Otros recursos</p> <p>Recurso humano</p> <ul style="list-style-type: none"> <input type="checkbox"/> Operadores de equipo <input type="checkbox"/> Dibujantes <input type="checkbox"/> Instructores <input type="checkbox"/> Bibliotecarios <input type="checkbox"/> Redactores <input type="checkbox"/> Consultores <input type="checkbox"/> Diseñadores gráficos <input type="checkbox"/> Mantenimiento <p>Entrenamiento y capacitación</p> <ul style="list-style-type: none"> <input type="checkbox"/> Sobre hardware <input type="checkbox"/> Sobre software <input type="checkbox"/> Sobre estándares <input type="checkbox"/> Equipo audiovisual <input type="checkbox"/> Consultores <p>Viajes</p> <ul style="list-style-type: none"> <input type="checkbox"/> Visitas a clientes y consultores <input type="checkbox"/> Reuniones profesionales y simposios <p>Costos indirectos</p> <p>Facilidades físicas</p> <ul style="list-style-type: none"> <input type="checkbox"/> Espacio de oficina <input type="checkbox"/> Mobiliario de oficina <input type="checkbox"/> Agua potable <input type="checkbox"/> Energía eléctrica, teléfono, Internet <input type="checkbox"/> Seguridad <p>Consumibles</p> <ul style="list-style-type: none"> <input type="checkbox"/> Papelería de oficina <input type="checkbox"/> Papel de computadora <input type="checkbox"/> Medios magnéticos <input type="checkbox"/> Cintas y tintas de impresión <p>Otros</p> <ul style="list-style-type: none"> <input type="checkbox"/> Servicios secretariales <input type="checkbox"/> Servicios de reproducción <input type="checkbox"/> Servicios de mensajería <input type="checkbox"/> Automóviles, motoristas <input type="checkbox"/> Alquileres <input type="checkbox"/> Horas extra <input type="checkbox"/> Impuestos, multas
--	--

Cuadro 10.6. Factores ponderables que pueden incrementar los estimados.

<input type="checkbox"/> Requerimientos vagamente definidos <input type="checkbox"/> Es para más de un usuario <input type="checkbox"/> Operará en tiempo real <input type="checkbox"/> Requiere interfaces con otros sistemas <input type="checkbox"/> Interfaces con otros sistemas indefinidas <input type="checkbox"/> Analistas no trabajaron en sistemas similares <input type="checkbox"/> Diseñadores no trabajaron en sistemas similares <input type="checkbox"/> Programadores no trabajaron en sist. similares <input type="checkbox"/> Personal de dirección no trabajó en sist. Similares <input type="checkbox"/> Se carece de experiencia en programación <input type="checkbox"/> Se carece de experiencia en análisis/diseño <input type="checkbox"/> Recursos en disco duro severamente limitados <input type="checkbox"/> Otros recursos de equipo severamente limitados <input type="checkbox"/> Los diseñadores no son programadores expertos <input type="checkbox"/> El ambiente de trabajo promoverá interrupciones <input type="checkbox"/> Analistas sin experiencia en programación <input type="checkbox"/> Programadores carecen experiencia como analistas <input type="checkbox"/> Programadores deben entrenarse en algún software <input type="checkbox"/> Analistas deben ser entrenados en algún software <input type="checkbox"/> Un alto % de analista/progr son recién graduados <input type="checkbox"/> Analista/programadores deben entrenarse en equipo de computación nuevo <input type="checkbox"/> Se esperan muchos cambios durante el desarrollo: requerimientos, diseño o usuarios <input type="checkbox"/> El SW a desarrollar contiene un gran número de funciones	<input type="checkbox"/> Requiere mucha innovación <input type="checkbox"/> Se instalará en más de un lugar <input type="checkbox"/> Controlará equipo sofisticado <input type="checkbox"/> Requiere modificar programas hechos por otros <input type="checkbox"/> Es más grande que los trabajados hechos antes <input type="checkbox"/> Se compartirá recursos con otros proyectos <input type="checkbox"/> Control de recursos de computación incompleto <input type="checkbox"/> Obligado a adoptar stds. diferentes a los propios <input type="checkbox"/> Bases de datos complejas y aún no definidas <input type="checkbox"/> Las bases de datos son confidenciales <input type="checkbox"/> Se debe proveer programas utilitarios <input type="checkbox"/> Baja perspectiva de continuidad del personal <input type="checkbox"/> El <i>cliente</i> proporcionará las bases de datos <input type="checkbox"/> El <i>cliente</i> proporcionará los datos de prueba <input type="checkbox"/> El <i>cliente</i> aprobará especificaciones de diseño <input type="checkbox"/> Terceros aprobarán especificaciones de diseño <input type="checkbox"/> El <i>cliente</i> tiene experiencia en SI y en computación <input type="checkbox"/> Hay que trabajar bajo supuestos formulados por el <i>cliente</i> <input type="checkbox"/> El SW se desarrollará en una plataforma desconocida <input type="checkbox"/> Computador para desarrollo es diferente al de operación <input type="checkbox"/> El computador para pruebas es diferente al de operación <input type="checkbox"/> El director del proyecto está involucrado en más de un proyecto <input type="checkbox"/> La selección de personal está fuera de control del director del proyecto
--	--

Por cada factor marcado, el costo puede incrementarse desde un 5% y hasta un 15% del costo estimado.