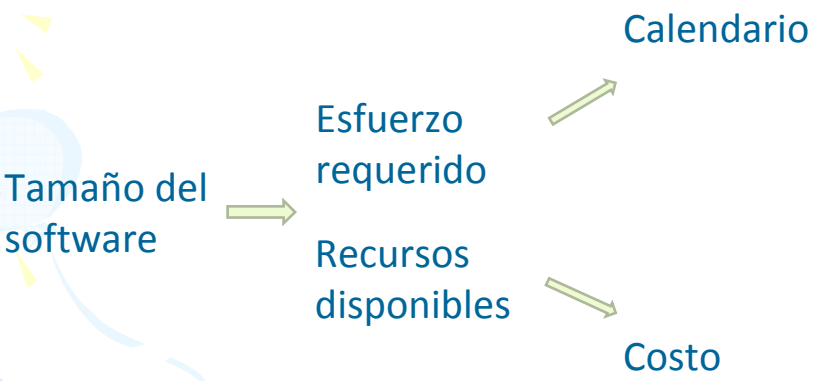
Administración de Proyectos

2. Estimación

Preguntas fundamentales del cliente

- ¿Cuánto me va a costar?
- ¿Cuándo me lo entregas?

Elementos relacionados



Estimación e Incertidumbre(1/3)

- No existe una forma simple de hacer una estimación precisa del esfuerzo requerido para desarrollar un sistema de software.
- Las estimaciones iniciales se hacen bajo la base de la definición de requerimientos de usuario de alto nivel.
- En la primera etapa del proyecto es difícil producir una estimación precisa de los costos de desarrollo del sistema.

Somerville, Ian (2002) Ingeniería de Software, 23.2

Estimación e Incertidumbre (2/3)

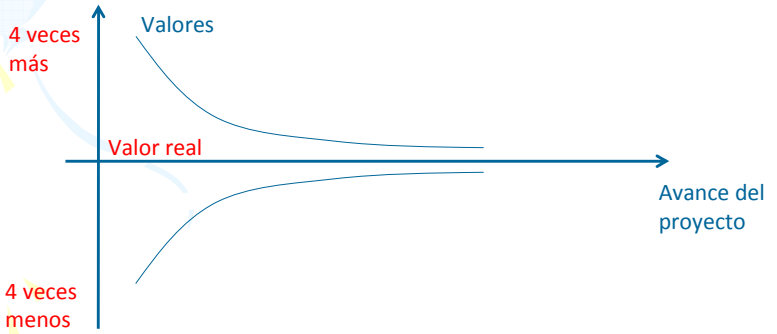
- Por lo general las estimaciones de los costos del proyecto se cumplen por su propia naturaleza.
  - La estimación se utiliza para definir el presupuesto del proyecto y el producto se ajusta para que las cifras del presupuesto se cumplan.

Estimación e Incertidumbre (3/3)

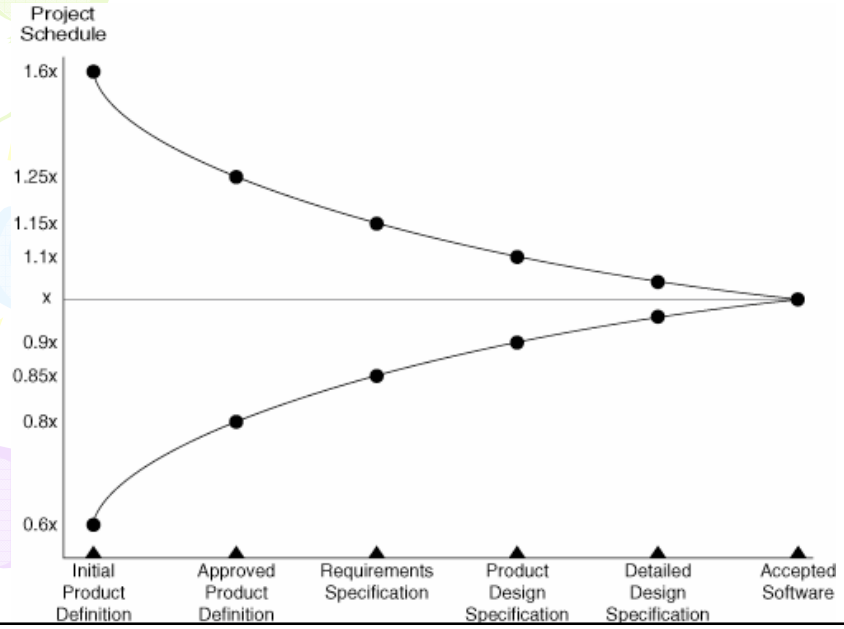
- Las organizaciones necesitan:
  - Hacer esfuerzos de software y
  - Estimaciones de costos
- Para ésto se ayudan de técnicas de estimación.

Cono de incertidumbre

- Si se define al **proyecto** como:
  - un *esfuerzo limitado en el tiempo para lograr un objetivo*.
- El **cono de incertidumbre** es:
  - la herramienta para realizar su **estimación de costo y tiempo.**
- En la estimación de costos y tiempo interviene:
  - el nivel de definición de las fases del proyecto
- La estimación de costo va desde **0.25x** hasta **4x:**
  - siendo **"x"** el **costo real final**,
    - que se conoce con certeza al terminar el proyecto



Ejemplo de un Cono de Incertidumbre [Barry Boehm]



Técnicas de Estimación de descomposición

- Utilizan:
  - Una aproximación de "divide y vencerás" para la estimación del costo y esfuerzo de un proyecto de software.
  - Un enfoque de división del proyecto en:
    - sus funciones principales o
    - en las tareas de ingeniería del software correspondientes donde la estimación del costo y del esfuerzo se hace de una forma escalonada idónea

Pressman, R. (2006) Ingeniería de Software, 23.6

Tamaño del software

- El tamaño del software se refiere a:
  - un resultado cuantificable del proyecto de software
- Se puede medir
  - Directamente: en líneas de código LDC
  - Indirectamente: usando puntos de función PF u otra medida

Pressman, R. (2006): Ingeniería de Software, 23.6

Estimación de tamaño

MÉTRICA DE LÍNEAS DE CÓDIGO (1/5)

- Es una métrica ampliamente usada en la medida de la productividad.
- Se calcula contando el número total de líneas de código que se entrega (o entregará).
- Algunas técnicas de conteo de líneas consideran:
  - Sólo las instrucciones ejecutables;
  - Las instrucciones ejecutables y las de datos;
  - Cada línea que no esté en blanco en el programa, independientemente de lo esté en esa línea.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE LÍNEAS DE CÓDIGO(2/5)

- Ventajas:
  - Fácil de visualizar
- Desventajas:
  - Difícil de definir
  - Varía con lenguajes
  - Varía con costumbres de codificación
  - La afectan los ambientes de desarrollo

MÉTRICA DE LÍNEAS DE CÓDIGO(3/5)

- Comparar la productividad de los diferentes lenguajes de programación da impresiones engañosas.
- Entre más expresivo sea un lenguaje de programación, más baja será la productividad aparente.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE LÍNEAS DE CÓDIGO(4/5)

	ANÁLISIS	DISEÑO	CODIFICA-CIÓN	PRUEBAS	DOCU-MEN-TACIÓN
Código Ensamblador	3 semanas	5 semanas	8 semanas	10 semanas	2 semanas
Lenguaje de Alto nivel	3 semanas	5 semanas	8 semanas	6 semanas	2 semanas

	TAMAÑO	ESFUERZO	PRODUCTIVIDAD
Código Ensamblador	5,000 líneas	28 semanas	714 líneas/mes
Lenguaje de Alto nivel	1500 líneas	20 semanas	300 líneas/mes

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE LÍNEAS DE CÓDIGO (5/5)

- El programador:
  - en ensamblador tiene una productividad de 714 líneas/mes
  - en lenguaje de alto nivel obtiene una productividad de 300 líneas/mes,
  - menos de la mitad de la productividad que nos da usar el lenguaje ensamblador.
- Aparentemente los costos de desarrollo para el sistema en lenguaje de alto nivel son menores y se producen en menos tiempo.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

Medidas indirectas

- Puntos de función
- Puntos de casos de uso
- Puntos objeto
- Puntos de función Cósmicos

Vistos en Ingeniería de Software

Pressman, R. (2006): Ingeniería de Software, 23.6

Estimación de tamaño
Puntos de función

Cálculo de Puntos de Función

1. Calcular Puntos de función sin ajustar
   - Establecer los 5 indicadores y su dificultad
2. Establecer Modificadores con su grado de influencia
3. Aplicar fórmula
   - pf = T * (0.65 + 0.01 * M)
4. Interpretar

Cálculo de Puntos de Función sin ajustar

- Una vez señalado indicadores, su complejidad y modificadores se asignan pesos que ya están dados por el método.

Puntos de Función sin ajustar.

Indicador	simple	mediano	complejo	SUMA
ALI	_ * 7	_ * 10	_ * 15	___
AIE	_ * 5	_ * 7	_ * 10	___
EE	_ * 3	_ * 4	_ * 6	___
SE	_ * 4	_ * 5	_ * 7	___
CE	_ * 3	_ * 4	_ * 6	___
			T =	____

Estimación de cada uno de los factores ponderados de complejidad

Factor	Valor
1.- Respaldo y recuperación	4
2.- Comunicaciones y datos	2
3.- Procesamiento distribuido	0
4.- Desempeño critico	4
5.- Entorno Operativo existente	3
6.- Entrada de datos de línea	4
7.- Transacción de entrada sobre pantallas múltiples	5
8.- ILF actualizado en línea	3
9.- Complejo de valores de dominio de información	5
10.- Complejo de procesamiento interno	5
11.- Código diseñado para reutilización	4
12.- Conversión/instalación en diseño	3
13.- Instalaciones múltiples	5
14.- Aplicación diseñada para cambio	5
∑(Fi) =	52

Conclusión sobre Puntos de Función

- El método de Puntos de Función sirve para:
  - detallar los requerimientos no funcionales o restricciones del sistema
  - estimar el costo, tiempo y personal requerido para el desarrollo de un sistema
- Es un método cuantitativo que apoya a la estimación de tamaño y sus derivados

Puntos de Casos de Uso

Puntos de casos de uso

- Introducida por Gustav Karner en su tesis en 1993 (Universidad de Linkoping).
- Utiliza número de actores y de casos de uso identificados, los clasifica en tres niveles y calcula un tamaño abstracto
- Con este se calcula el esfuerzo que costará desarrollarlos.
- Aún no está estandarizada

Ecuación completa

UCP=UUCP*TCF*ECF

UCP=Puntos de Casos de Uso
UUCP=Puntos de Casos de Uso sin ajustar
TCF= Factor de Complejidad Técnica
ECF= Factor de Complejidad Ambiental

Puntos de casos de uso sin ajustar

- $UUCP = \sum pa_i * A_i + \sum pcu_j * Cu_j$
  - donde:
  - Pa_i del tipo de actor i
  - A_i número de actores de tipo i
  - Pcu_j peso del tipo de caso de uso j
  - Cu_j número de casos de uso de tipo j

Ejemplo de cálculo de casos de uso

Tipo de Caso de Uso	Descripción	Peso	Número de Casos de Uso	Resul-tado
Simple	Transacciones= 3 o menos Clases= Menos de 5	5	7	35
Medio	Transacciones= 4 a 7 Clases= 5 a 10	10	13	130
Complejo	Transacciones= Más de 7 transacciones Clases= Más de 10 clases	15	3	45
			Total UUCW	**210**

Factor de Complejidad Técnica

- Éste se compone de 13 indicadores que evalúan la complejidad de los módulos del sistema que se desarrolla.
- Cada uno tiene un peso definido.
- Indicadores subjetivos, determinado por la percepción del equipo de desarrollo.
  - TCF=0.6+(0.1*Factor Total Técnico)

Elementos de Complejidad Técnica

Factor técnico	Descripción	Peso
T1	Sistema distribuido	2
T2	Rendimiento o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5

Elementos de Complejidad Técnica

Factor técnico	Descripción	Peso
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Características especiales de seguridad	1
T12	Provee acceso directo a terceras partes	1
T13	Se requiere facilidades especiales de entrenamiento a usuario	1

Calculando ECF

- ECF=1.4+(-0.03*Factor Ambiental Total)
- ECF = 1.4 + (-0.03 * 26)
- ECF=0.62

Factor de Complejidad Ambiental

- Establece una medida de la experiencia del equipo de desarrollo.
  - Indicadores relacionados con las habilidades y experiencia del equipo de desarrollo del proyecto.
- También son subjetivos, como los percibe el equipo.
  - ECF = 1.4 +(-0.03*Factor Ambiental Total)

Indicadores de complejidad ambiental

Factor Ambiental	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado Familiaridad con UML	1.5
E2	Personal part-time	-1
E3	Capacidad del analista líder	0.5
E4	Experiencia en la aplicación	0.5
E5	Experiencia en orientación a objetos	1
E6	Motivación	1
E7	Dificultad del lenguaje de programación	-1
E8	Estabilidad de los requerimientos	2

Puntos Objeto

Barry Boehm

MÉTRICA DE PUNTOS DE OBJETO (1/6)

- Es una alternativa para los puntos de función cuando se utilizan 4GL ó leguajes comparables para el desarrollo de software.
- Los puntos de objeto se utilizan en el modelo de estimación COCOMO II
  - que se usa como base para estimar el esfuerzo, calendarización y costos de un nuevo proyecto de software.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE PUNTOS DE OBJETO(2/6)

- El número de puntos de objeto en un programa es una estimación con peso calculado a partir de:

- Plantillas (pueden comprender varias ventanas o vistas) independientes que se despliegan.
  - sencillas cuentan con 1 punto de objeto,
  - moderadamente complejas cuentan como 2 y
  - muy complejas cuentan como 3 puntos objeto.

- Informes (reportes que pueden comprender varias secciones) que se producen.
  - simples cuentan como 2,
  - moderadamente complejos cuentan como 5 y
  - difíciles de producir cuentan como 8.

- Módulos 3GL que deben desarrollarse para complementar el código 4GL.
  - cuenta como 10 puntos objeto.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE PUNTOS DE OBJETO(3/4)

- Para determinar la complejidad de cada objeto se usan las siguientes variables:
  - NOP: Nuevos puntos objeto. Cantidad de puntos objeto ajustados por la reutilización.
  - Srvr: Número de tablas de datos del servidor usadas junto con la plantilla o el informe
  - Clnt: Número de tablas de datos del cliente usadas junto con la plantilla o el informe

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE PUNTOS DE OBJETO (4/6)

Complejidad Asociada a instancias de objeto

Para plantillas				Para informes			
# de vistas	Número y fuente de tablas de datos			# de secciones	Número y fuente de tablas de datos		
	Total < 4 (<2 Srvr <3 Clnt)	Total < 8 (2 a 3 Srvr 3 a 5 Clnt)	Total > 8 (>3 Srvr >5 Clnt)		Total < 4 (<2 Srvr <3 Clnt)	Total < 8 (2 a 3 Srvr 3 a 5 Clnt)	Total > 8 (>3 Srvr >5 Clnt)
< 3	Simple	Simple	Medio	0 ó 1	Simple	Simple	Medio
3 a 7	Simple	Medio	Difícil	2 ó 3	Simple	Medio	Difícil
>= 8	Medio	Difícil	Difícil	>= 4	Medio	Difícil	Difícil

Moreno, A. M. y Capuchino, S. (2003): Estimación de proyectos de Software, Tema 7, Escuela Superior de Ingeniería Informática, Universidad de Vigo, España, *trevinca.ei.uvigo.es/~cfajardo/Nueva_carpeta/presentaciones/**cocomo2**k.pdf*

MÉTRICA DE PUNTOS DE OBJETO(4/5)

- La ventaja de utilizar puntos objeto en lugar de puntos de función es:
  - son más fáciles de estimar a partir de la especificación del software de alto nivel.
- Los puntos objeto sólo se refieren a:
  - pantallas, informes y módulos 3GL.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

MÉTRICA DE PUNTOS DE OBJETO(5/5)

Ponderación de complejidad para tipos de objeto [BOE96].

Tipo de objeto	**Peso de complejidad**		
	Simple	Medio	Difícil
Plantilla	1	2	3
Reporte	2	5	8
Componente 3GL			10

Ejercicio (1/2)

- Para el siguiente problema calcular su tamaño
  - Estime su tamaño en LDC
  - Obtenga su medida en PF
  - Obtenga su medida en PCU
  - Obtenga su medida en PO
  - Comente los resultados

Ejercicio (2/2)

- Una compañía importadora de materiales para vitrales desea desarrollar un sistema para que sus clientes consulten los vidrios, herramientas y otros materiales que ofrece vía Internet.

- El sistema deberá contar con diversas secciones:
  - Sección 1. Registro y consulta:
    - Registro de clientes nuevos;
    - Consulta de vidrios, organizados en familias;
    - Consulta de herramientas (esmeriladoras, cortadoras, pinzas, cautines) y
    - Consulta de insumos (soldadura, aceite, cinta de cobre, etc.)
  - Sección 2. Pedidos, sólo para clientes registrados
  - Sección 3. Actualización de información.
  - Sección 4. Atención de pedidos.
- Se calcula que manejará:
  - 200 clases de vidrio
  - 100 herramientas
  - 100 insumos
  - Entre dos mil y cinco mil clientes
  - 50 pedidos por día, en promedio

ESFUERZO

Unidades

- El esfuerzo es medida abstracta:
  - Promedio de trabajo necesario por persona
  - Horas-persona (la más clara y útil en proyectos pequeños)
  - Semanas-persona (suponiendo 40 horas por semana, pero puede ajustarse)
  - Persona-mes (suponiendo 20 días de 8 horas por día, es decir 160 horas-persona; puede ajustarse)
- Un trabajo puede hacerlo una, dos o más personas.

Técnicas disponibles (Boehm)

- Modelos algorítmicos
- Juicio de expertos
- Analogía
- Parkinson (el esfuerzo disponible es el estimado)*
- Para ganar (mínimo para lograr contrato)
- Descomposición top-down (por tareas componentes)
- Composición bottom-up (se agregan estimados)

* Realmente no es una técnica

Comentarios (1/3)

- Juicio de expertos:
  - es difícil contar con ellos; dependen de experiencia y datos registrados
- Analogía:
  - es difícil establecerla, varia mucho; requiere datos históricos
- Parkinson:
  - limita alcance; útil para trabajo interno de empresa
- Para ganar:
  - sirve para ganar concurso, pero no es bueno

Hughes y Cotterell, capítulo 5

Comentarios (2/3)

- Bottom-up se descompone en tareas, de modo iterativo (esta etapa es top-down) y se va asignando esfuerzo a cada una; se van sumando.
  - Depende de tener estimados históricos para los datos de las actividades.
  - Es bueno en estimaciones detalladas en la etapa de planeación; no son buenas al inicio.
  - Ejemplo basado en proceso, más adelante

Hughes y Cotterell, capítulo 5

Comentarios (3/3)

- Top-down: asociado a modelos paramétricos donde
- esfuerzo = tamaño * productividad
  - Puntos de función y puntos de casos de uso son de este tipo
- Algorítmicos: usan indicadores que guían la duración; un caso es COCOMO II.

Hughes y Cotterell, capítulo 5

Sobreestimación

- Ley de Parkinson: "el trabajo se expande hasta llenar todo el tiempo disponible"
- Ley de Brooks: El esfuerzo crece desproporcionadamente con el número de personal, por coordinación, comunicación, administración
  - "Poner más gente en un proyecto retrasado lo hará retrasarse más"

Estimación muy optimista

- Sufre en la calidad o la funcionalidad
- Ley 0 de fiabilidad de Weinberg: si un sistema no necesita ser fiable, puede alcanzar cualquier otra meta.

Combinar medidas

- Si hay varias mediciones de la estimación (est), se pueden combinar:

$$\frac{\text{est optimista} + 4 * \text{est probable} + \text{est pesimista}}{6}$$

Pressman, R. (2006) Ingeniería de Software.

Descomposición basada en proceso

- Descompone en actividades según modelo de desarrollo (ciclo de vida).
- Depende de datos anteriores y experiencia.

Estimación basada en el proceso (1/2)

- Delimitar las funciones del software.
- Identificar las tareas de ingeniería del software para cada una de las funciones y representarlas en una tabla.
- Estimar el esfuerzo (número de personas/unidad de tiempo) de realización de cada tarea para cada una de las funciones del software.

Pressman, R. (2006) Ingeniería de Software, 23.6

Estimación basada en el proceso (2/2)

- Aplicar las tarifas laborales (costo/unidad de esfuerzo) correspondientes a cada una de las tareas.
- Calcular los costos y el esfuerzo para cada función y cada tarea.
- Es la técnica más utilizada para calcular el costo de un proyecto de software.

Pressman, R. (2006) Ingeniería de Software, 23.6

Ejemplo de estimación basada en el proceso

CC = Comunicación del Cliente EC = Evaluación del Cliente (persona-mes)

Actividad>	CC	Planif	Análisis de riesgo	Ingeniería		Liberación de construcción		EC	Totales
Tarea>				Análisis	Diseño	Código	Prueba		
Función									
FIUC				**0.50**	**2.50**	**0.40**	**5.00**		**8.40**
AG2D				**0.75**	**4.00**	**0.60**	**2.00**		**7.35**
AG3D				**0.50**	**4.00**	**1.00**	**3.00**		**8.50**
FPGC				**0.50**	**3.00**	**1.00**	**1.50**		**6.00**
GBD				**0.50**	**3.00**	**0.75**	**1.50**		**5.75**
FCP				**0.25**	**2.00**	**0.50**	**1.50**		**4.25**
MAD				**0.50**	**2.00**	**0.50**	**2.00**		**5.00**
Totales	**0.3**	**0.25**	**0.25**	**3.50**	**20.50**	**4.75**	**16.50**		**46.00**
% esfuerzo	**0.6**	**0.5**	**0.5**	**7.6**	**44.6**	**10.3**	**35.9**		

Top-down

- Modelos empíricos siguen forma general:
- Esfuerzo = C + A * Tamaño ^{B} * M
  - A: constante según prácticas locales y tipo de software; se obtiene por regresión
  - B: exponente obtenido por regresión; usualmente entre 1.0 y 1.5. (NO LINEAL)
  - M: factor de ajuste según software, experiencia del equipo.
  - C: constante obtenida por regresión

Sommerville 2005, sección 26.3,
Pressman 2005, sección 23.7

Esfuerzo según tamaño algunas relaciones

- Bailey-Basili:
  5.5 + 0.73 * (KLDC)1.16
- Boehm (simple):
  3.2 * (KLDC)1.05
- Doty para KLDC>9:
  5.288 * (KLDC)1.0457
- Albrecht-Gaffney:
  -91.4 + 0.355 * PF
- Kemerer:
  -37 + 0.96 * PF
- Medida persona-mes

Deben ajustarse a cada empresa

Pressman 2005, sección 23.7

Versiones simplificadas

- Usar relaciones conocidas o tablas.
  - PO:
    - esfuerzo = tamaño / productividad
  - PCU:
    - esfuerzo = tamaño * productividad
      - entre 15 (experto) y 30 (inexperto)
  - PF:
    - esfuerzo = tamaño * productividad
      - usar Tablas de Longstreet

PF desde Tablas

PF	Horas-persona		PF	Horas-persona
50	1.3		7000	12.1
100	1.4		8000	16.3
500	1.6		9000	22.1
1000	2.0		10000	29.8
2000	2.7		11000	40.2
3000	3.8		12000	54.3
4000	4.9		13000	73.3
5000	6.6		14000	98.9
6000	9.0		15000	133.6

Ojo: es esfuerzo por cada punto de función

Longstreet

Esfuerzo con tamaño y productividad

PRODUCTIVIDAD

- La productividad en un sistema de manufactura se mide:
  - contando el número de unidades que se producen y dividiendo éste entre el número de personas-hora requeridas para producirlas.
- Los administradores tienen que:
  - estimar la productividad de los ingenieros en el proceso de desarrollo de software.

Somerville, Ian (2002) Ingeniería de Software, sección 23.1

Datos históricos de productividad en LDC

- Una revisión de datos históricos arroja los siguientes valores:
  - Promedio organizacional de productividad para sistemas de un tipo dado es de 620 LDC/pm
- Así:
  - Un sistema de 4900 LDC requiere:

Esfuerzo =(4900 LDC)÷(620 LDC/pm)=7.90 pm

pm=persona-mes

Pressman, R. (2006) Ingeniería de Software, 23.6

Puntos Objeto

- Al aplicar el desarrollo basado en componentes o la reutilización general de software se estima el porcentaje de reutilización (%reut) y se ajusta la cuenta de puntos objeto:

NPO=(puntosobjeto)*[(100-%reut) /100]

Tasa de productividad

- Para obtener una estimación del esfuerzo con base en el valor NPO calculado, se debe calcular una "tasa de productividad".

Tabla 2. Tasa de productividad por puntos objeto [BOE96].

Experiencia/capacidad del desarrollador	Muy baja	Baja	Nominal	Alta	Muy alta
Madurez/capacidad del entorno	Muy baja	Baja	Nominal	Alta	Muy alta
TASA PRODUCTIVIDAD (PROD)	4	7	13	25	50

- PROD = NPO/persona-mes

Algunos datos productividad

- Tiempo real: 40 a 160 LDC/pm
- Sistemas: 150 a 400 LDC/pm
- Aplicaciones comerciales: 200 a 800 LDC/pm
- En puntos objeto: 4 a 50 PO/pm

Esfuerzo estimado

- Una vez determinada la tasa de productividad se puede obtener una estimación del esfuerzo del proyecto:
  - Esfuerzo estimado = NPO/PROD

Esfuerzo con PO

Ejemplo:

- Suponga que se obtuvieron 65 PO y no se reutilizará nada; NPO = 65 PO
- Suponga un equipo promedio (productividad = 13 PO/persona-mes)
- Esfuerzo = 65 PO / 13 PO/persona-mes
- Esfuerzo = 5 personas-mes

COSTO

Cálculo del costo

- Costo principal en el software: tiempo de los desarrolladores
  - Se toma como base el costo de sueldos que es el factor dominante en software
  - Por eso se basa en esfuerzo necesario
- Otros costos:
  - Directos: materiales, equipo, software
  - Indirectos: gastos generales de la empresa, financiamiento, cambios de moneda

Cálculo del costo

- Se debe distinguir costo del precio de venta (este incluye la ganancia)
- FALTA agregar ganancia esperada

Técnicas de estimación

- Basados en el cliente (MALO)
- Las mismas que para el esfuerzo, que se vieron antes

Somerville, Ian (2002) Ingeniería de Software, 23.2

Conociendo esfuerzo

- Costo = esfuerzo * salario promedio
  - Si se desglosaron en actividades, pueden diferenciarse salarios
  - Si se usa persona-mes, se suponen semanas de 40 horas de trabajo

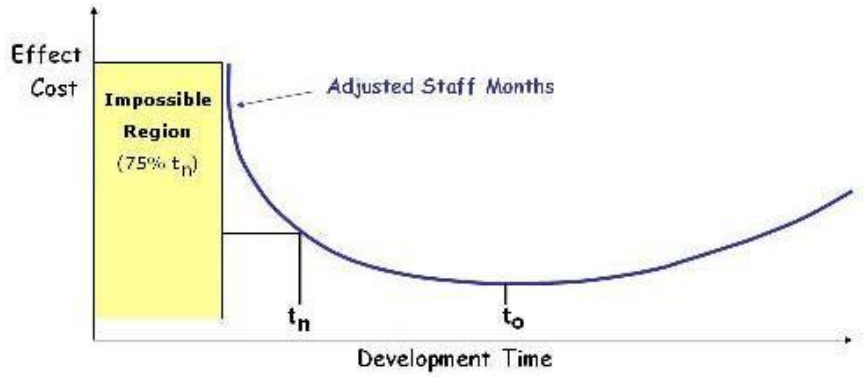
Ejercicio

- Para los valores obtenidos en el ejercicio pasado, calcular costos

Tiempo Calendario

Tiempo conveniente

Putnam Norden Rayleigh (PNR) Curve



t_n = Nominal delivery time

t_o = Optimal delivery time (in terms of cost) = 2t_n

Tiempo conveniente

- $Tn = F * Esfuerzo^{0.33}$
- F = 3.67 (default)
- F = 3.10 Web
- F = 3.20 e-commerce
- F = 4.00 embebido
- Se espera que técnicas mejores y métodos ágiles bajen a 3.0 ó más.
- $To \approx 2\ Tn$
- Esta forma supone esfuerzo medido en meses-persona y produce tiempo en meses.

Enfoque global

- ¿Conviene hacer el sistema desde cero?
- ¿Pueden reutilizarse componentes existentes?
- ¿Se pueden subcontratar partes o todo el sistema?
- Cada una tiene sus riesgos y deben adecuarse

Componentes

- Hay componentes preexistentes
  - De la compañía
  - De software más o menos libre
  - De proveedores (deben pagarse)
- Generalmente deben adaptarse (y eso cuesta)
- Hay otros costos por aprendizaje

Subcontratación (1/3)

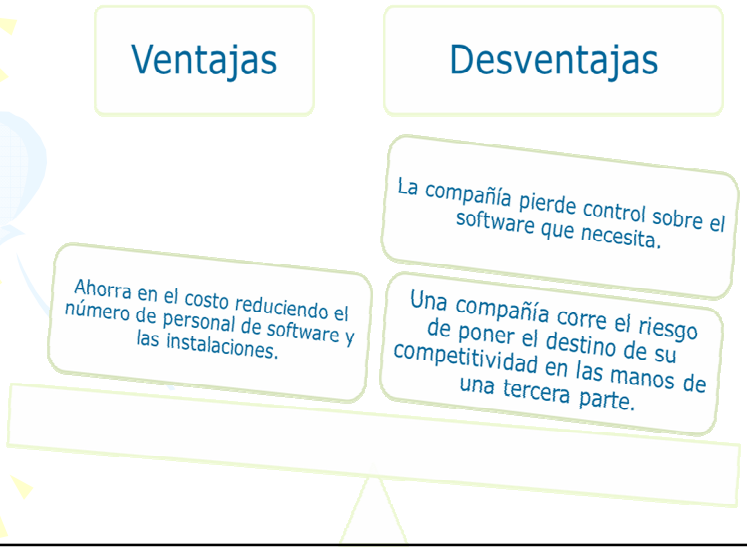
- La subcontratación es cualquier actividad que conduce a la adquisición de software o algunos de sus componentes con una fuente externa a la organización de ingeniería de software.

Subcontratación (2/3)

- La decisión de subcontratar puede ser:
  - Estratégica: Los gestores comerciales consideran si una porción significativa de todo el trabajo de software se puede contratar con otros.
  - Táctica: Un gestor de proyecto determina si parte o todo un proyecto puede lograrse mejor al subcontratar el trabajo de software.

Subcontratación (3/3)

- La decisión de subcontratar usualmente es financiera



La Decisión de Desarrollar o Comprar (1/3)

- En muchas áreas de aplicación es más rentable adquirir que desarrollar software.
  - Los gestores de ingeniería son los encargados de tomar esta decisión.

La Decisión de Desarrollar o Comprar (2/3)

- Existen varias opciones de adquisición:
  - El software se puede comprar (o adquirir la licencia) ya desarrollado.
  - Los componentes de software se pueden adquirir y luego modificar e integrar para satisfacer necesidades.
  - El software se puede construir de manera personalizada por medio de un contratista externo (outsourcing) para satisfacer las necesidades del comprador

La Decisión de Desarrollar o Comprar (3/3)

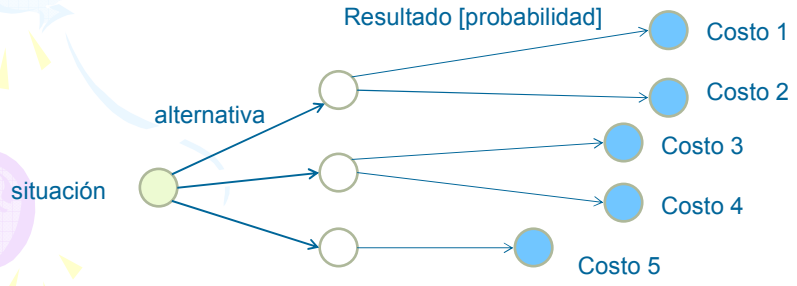
- En el análisis final, la decisión desarrollar-comprar se realiza basándose en las siguientes condiciones:
  - ¿El producto de software estará disponible antes que el software desarrollado de manera interna?
  - El costo de adquisición más el costo de personalización, ¿será menor que el costo de desarrollar el software de manera interna?
  - ¿El costo de soporte externo será menor que el costo de soporte interno?

Creación de un árbol de decisión

- El árbol de decisión es una técnica estadística de apoyo a las decisiones.
- Se deben considerar muchos criterios durante el proceso de toma de decisión:
  - disponibilidad
  - experiencia desarrollador, vendedor o contratista
  - concordancia de los requisitos con "políticas locales" y la probabilidad de cambiar.

Árbol de decisión

- De una situación problemática surgen alternativas de solución que llevan a un nodo cada una
- Cada rama puede tener uno o más resultados, cada uno con cierta probabilidad
- Cada resultado tiene un costo
- El proceso puede repetirse



Ejemplo de creación de un árbol de decisión (1/5)

- Se necesita un sistema basado en un software X. La organización de ingeniería de software puede:
  - Construir el sistema X desde cero.
  - Reutilizar componentes existentes de "experiencia parcial" para construir el sistema.
  - Comprar un producto de software disponible y modificarlo para satisfacer las necesidades locales.
  - Contratar el desarrollo de software con una empresa externa.

Ejemplo de creación de un árbol de decisión (2/5)

Trayectoria: construir

Factor	**Valores**
El sistema se construirá desde cero .	70% de probabilidad de que el trabajo sea difícil.
Estimación del esfuerzo de desarrollo difícil.	450 000 dólares (DLS)
Estimación del esfuerzo de desarrollo fácil.	380 000 DLS

Ejemplo de creación de un árbol de decisión (3/5)

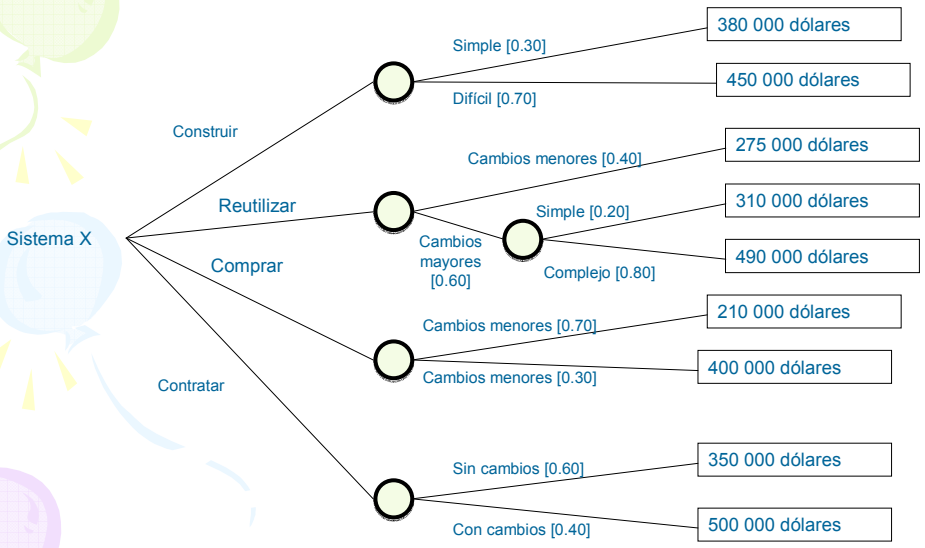
- Valor esperado para el costo calculado a lo largo de cualquier rama del árbol de decisión es:

  costo-esperado=Σ(probabilidad ruta$_i$)*(costo estimado ruta$_i$)

  - Donde i es la trayectoria del árbol de decisión.
- Ejemplo para la trayectoria de construcción:

  costo-esperado=0.30(380K$_{DLS}$)+0.70(450K$_{DLS}$)= 29K$_{DLS}$

Ejemplo de creación de un árbol de decisión (4/5)



Árbol de decisión para apoyar la decisión desarrollar- comprar

Ejemplo de creación de un árbol de decisión (5/5)

- Costos esperados para las otras trayectorias del árbol:
  - Reutilizar

costo-esperado=$0.40(275K_{DLS})=0.60(310K_{DLS})+0.80(490K_{DLS})=382K_{DLS}$

  - Comprar

costo-esperado=$0.70(210K_{DLS})+0.30(400K_{DLS})=267K_{DLS}$

  - Contratar

costo-esperado=$0.60(350K_{DLS}) + 0.40(500K_{DLS}) = 410K_{DLS}$

- El costo esperado más bajo es la opción "comprar".

¿Estimación fallida?

- Tom de Marco (2011) dice que todos los proyectos que terminan tarde lo hacen por haber comenzado tarde
  - Por miedo al costo se pospone inicio
  - Por desconocimiento se pospone inicio
  - Si tiene estimado y no lo usa para comenzar a tiempo, terminará tarde