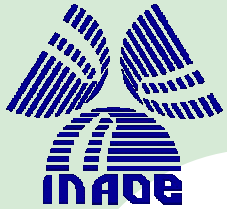


Coordinación de Ciencias Computacionales

TEMA 2.
LA ADMINISTRACIÓN DE PROYECTOS DE SOFTWARE

NOTAS DEL CURSO
C121. INGENIERÍA DE SOFTWARE

DRA. MA. DEL PILAR GÓMEZ GIL

pgomez@acm.org

VERSIÓN: 25-08-09

ADMINISTRACIÓN DEL DESARROLLO DE SOFTWARE

- Para poder realizar exitosamente un proyecto de software debemos entender:
  - El **alcance** del trabajo a realizarse
  - Los **riesgos** que estamos enfrentando
  - Los **recursos** requeridos
  - Las **tareas** a realizarse
  - Los **eventos** importantes a observar
  - El **esfuerzo** (costo) a invertir
  - El **calendario** a seguir

- La **ADMINISTRACIÓN** comienza antes que el trabajo técnico empiece, continúa con la transformación del SW desde que es una idea hasta que es una realidad, y termina cuando el SW es discontinuado.

Elementos Clave en la Administración del Proyecto de Software

1. Definición de objetivos y alcances.

2. Medidas y métricas

3. Estimación, que incluye:
   - -Esfuerzo humano (en meses-persona)
   - -Duración (en unidades de tiempo)
   - -Costo (en unidades monetarias $$$)

Elementos Clave en la Administración del Proyecto de Software (cont.)

4. Análisis de Riesgo, que implica:
   - Identificación del riesgo
   - Evaluación del riesgo
   - Determinación de prioridades
   - Administración del riesgo
   - Monitoreo del riesgo

5. Calendarización

6. Seguimiento y Control

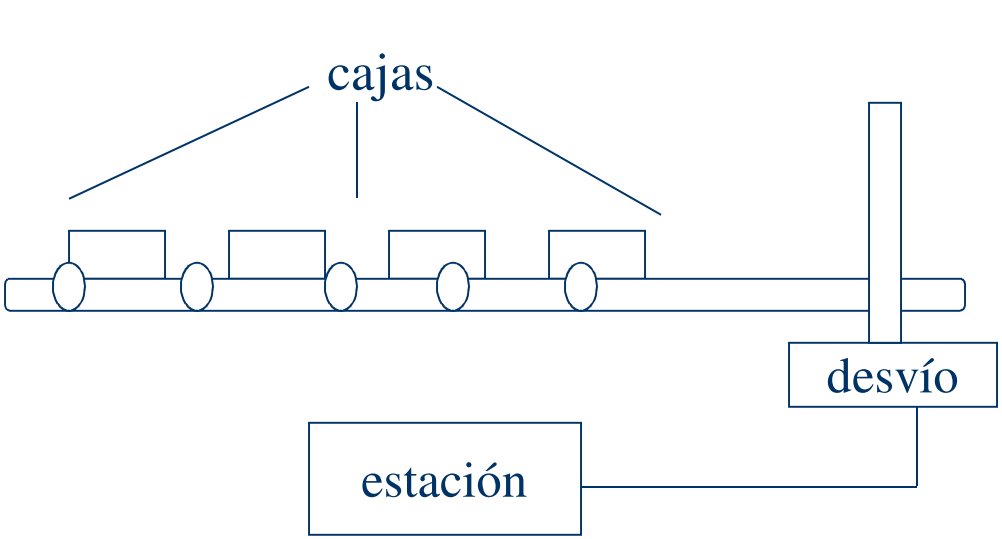
Ejemplo de una definición de Objetivos y Alcances

El sistema ordenador para la línea de transporte se encargará de ordenar cajas que se desplazan a través de una línea. Cada caja estará definida por un código de barras que contiene el número de parte y se colocará en alguno de los 4 estantes que están al final de la línea, siguiendo un orden de acuerdo a su número de parte. La estación de ordenamiento contiene un lector óptico y una PC. La PC estará conectada a un mecanismo de desvío que coloca las cajas en el estante que les corresponde. Las cajas se desplazan al azar y están separadas de una manera uniforme. La línea se mueve a una velocidad de 2 metros por minuto.

Ejemplo de Definición de Objetivos y Alcances (continuación)

El software recibirá información de entrada del lector óptico la cual se decodificará de manera que se identifique la caja. El número de parte se consultará en una base de datos a fin de determinar la localización de la caja. La base de datos contendrá a los más 2, 500 entradas. Se llevará un registro de la posición de colocación de cada caja. El sistema también recibirá información de un tacómetro que se te utilizará para sincronizar la señal de control que va al mecanismo de desvío. [Pressman 92]

Definición de Objetivos y Alcances (continuación)



[Pressman 92]

CONCEPTOS SOBRE GESTION DE PROYECTOS

... Las 4 P´s

- Personal
- Producto
- Proceso
- Proyecto

Personal de Software

- El factor humano es fundamental para el éxito del proyecto

- Participantes en el proceso de software: *Stakeholders*
  1. Administradores superiores
  2. Administradores técnicos del proyecto
  3. Profesionales
  4. Clientes
  5. Usuarios Finales

Personal de Software: Equipos de trabajo

- Características de los líderes de equipo:
  - Resolvedores de problemas
  - Habilidades administrativas
  - Incentivos por logros
  - Influencia y construcción de espíritu de equipo.

Personal de Software: Equipos de trabajo (cont.)

- Hay varias opciones para organizar personas. La mejor estructura depende de muchas cosas . Normalmente la organización de un equipo formal es la más productiva.
- El rendimiento de un equipo es inversamente proporcional a la cantidad de comunicación que se debe entablar.
- Toxinas para los equipos de trabajo:
  - Atmósfera de trabajo frenética
  - Frustración causada por factores tecnológicos, de negocio o personales
  - Procedimientos pobremente coordinados
  - Definición confusa de los papeles
  - Exposición repetida al fallo.

Los Diez Mandamientos (más 1) para un Código de Cooperación Efectivo

Tomado de : "A Learning Centered Approach to Engineering Education for the 21th. Century:UDLA", by L. Bellamy, B. McNeill & S. Foster ©2001

1. Ayuda a los demás a estar bien, no a estar mal.
2. Busca formas de que las nuevas ideas funcionen, no razones para que no funcionen.
3. Si tienes dudas, pregunta! No hagas conjeturas negativas acerca de los demás.
4. Ayuda a los otros a ganar, y siéntete orgulloso(a) del triunfo del otro.
5. Habla positivamente de los demás en el equipo y de tu organización cada vez que puedas.
6. Mantén una actitud mental positiva, no importa las circunstancias.
7. Actúa con iniciativa y coraje, como si todo dependiera de ti.
8. Hazlo todo con entusiasmo, éste es contagioso.
9. Cualquier cosa que quieras, déjala pasar.
10. En cualquier circunstancia, mantén la fe.
11. Diviértete!

*Ford Motor Company

MEDIDAS Y MÉTRICAS

¿Por qué medimos software?

1. Para conocer la calidad del producto
2. Para evaluar la productividad de las personas que lo producen
3. Para evaluar los beneficios de nuevas herramientas de software
4. Para formar una base a fin de estimar
5. Para justificar nuevas herramientas o entrenamientos adicionales.

Actividades del proceso de medición

- **Formulación.** Diseñar las medidas y métricas apropiadas para el software en consideración
- **Recolección.** Determinar y aplicar el mecanismo para acumular los datos requeridos
- **Análisis**: Cálculo de las métricas y aplicación
- **Interpretación:** Evaluación de las métricas a fin de obtener información sobre la calidad
- **Retroalimentación.** Recomendaciones al equipo de trabajo derivadas de la interpretación de las métricas

Tipos de Métricas

- Las medidas puede ser **directas** (ejemplo longitud) o **indirectas** (ejemplo calidad)
- Las métricas de software se pueden clasificar de diferentes maneras:
  - De acuerdo a lo que miden:
    - Medidas técnicas
    - Medidas de calidad
    - Medidas de productividad
  - De acuerdo a la manera como lo miden:
    - Medidas orientadas al tamaño
    - Medidas orientadas a funciones
    - Medidas orientadas a humanos

Métricas Orientadas al Tamaño

Ejemplos de métricas directas:

PROYECTO	ESFUERZO MESES-PERSONA	COSTO miles DLLS	KLOC*	# PAGS. DOCTO.	ERRORES	PERSONAS
SIARP-2.1	27	230	16.3	478	39	4
HUELLAS	54	421	29.4	1351	121	6
PGG-12	34	321	21.3	980	45	7

Métricas Orientadas al Tamaño

Ejemplo de Métricas Indirectas:

PRODUCTIVIDAD =	KLOC / MESES-PERSONA
CALIDAD =	DEFECTOS / KLOC
DOCUMENTACION =	# DE PAGINAS-DOCTO / KLOC

ERROR: Descubierto antes de la entrega al usuario final
DEFECTO: Descubierto después de la entrega

*KLOC = miles de líneas de código

Métricas Orientadas a Funciones

Ver sección 15.3.1 del libro de texto

Propuestas inicialmente por [Albretch,79]. Por cada sistema se calcula un número llamado Punto Funcional (PF) aplicando la siguiente fórmula:

$$\text{Punto Funcional} = \text{Total\_tabla} \times [0.65 + 0.01 \times \sum(F_i)]$$

Donde:

$\sum(F_i)$	es la sumatoria obtenida al aplicar los valores de ajuste de complejidad, descritos a continuación.
Total_tabla	es el total obtenido de la matriz de entradas y salidas descritos enseguida.

Métricas Orientadas a Funciones (cont.)
Matriz de entradas y Salidas

PARAMETRO	CANTIDAD	X	FACTOR DE PESO			TOTAL
			SIMPLE	PROMEDIO	COMPLEJO	
No. de entradas externas	10		3	4	6	40
No. de salidas external			4	5	7	
No. de preguntas hechas usuario			3	4	6	
No. de archivos	5		7	10	15	35
No. de interfaces externas			5	7	10	
TOTAL de la Tabla						

Parámetros en la matriz de entrada-salida

- **Número de entradas externas**: Cada entrada es dada por el usuario o transmitida de otra aplicación. Provee informacion orientada a la aplicación o de control. Normalmente se usa para actualizar Archivos Internos lógicos (ILF´s por sus siglas en inglés). Deben distinguirse de “Preguntas” que se contabilizan aparte
- **Número se salidas externas**. Proveen información al usuario. Pueden referir a reportes, pantallas, mensajes de error etc. Elementos individuales dentro de un reporte no se cuentan separadamente

Parámetros en la matriz de entrada-salida (cont.)

- **Número de preguntas externas**. Se definen como una pregunta en línea que genera una respuesta inmediata del software, en la forma de una salida en línea (frecuentemente resultado de una consulta a un ILF)
- **Numero de ILF internos**. Cada archivo lógico interno es un grupo de datos lógico que reside dentro de la aplicación y se mantiene a través de las entradas externas
- **Número de archivos externos de interfaz (EIF).** Cada EIF es un grupo de datos lógico que reside externo a la aplicación pero que proporciona datos que usará la aplicación

Métricas Orientadas a Funciones (cont.)
CALCULO DE VALORES DE AJUSTE DE COMPLEJIDAD

Rangos:

0 – Sin influencia	1 – Incidental	2 – Moderado
3 – Promedio	4 – Significativo	5 – Esencial

Conceptos (F_i):

1. ¿Requiere el sistema de recuperación y respaldos confiables?
2. ¿Se requiere comunicación de datos?
3. ¿Hay funciones de procesamiento distribuido?
4. ¿Es crítico un buen desempeño?
5. ¿Funcionará el sistema en un ambiente operacional ya existente y altamente utilizado?
6. ¿Requiere el sistema entradas de datos en línea?

Continua…

Métricas Orientadas a Funciones (cont.)
CALCULO DE VALORES DE AJUSTE DE COMPLEJIDAD

Conceptos (F_i) (continuación):

7. ¿Se requiere construir un sistema en línea sobre múltiples pantallas?
8. ¿Se modifican los archivos en línea?
9. ¿Hay entradas, archivos, salidas o preguntas complejas?
10. ¿Es el proceso interno complejo?
11. ¿Se debe diseñar código re-usable?
12. ¿Se incluyen los procesos de conversión e instalación?
13. ¿Está diseñado el sistema para instalaciones múltiples en diferentes organizaciones?
14. ¿Está diseñada la aplicación para facilitar cambios y para ser fácil de utilizar por el usuario?

[Artur 85 según Pressman 92]

Comparación Entre Métricas

- En promedio un **punto funcional** (PF) equivale a:

- 337 **líneas de código** en lenguaje ensamblador
- 90 **líneas de código** en PASCAL
- 77 **líneas de código** en COBOL
- 66 **líneas de código** en C++
- 63 **líneas de código** en Java
- 60 **líneas de código** en Pearl
- 47 **líneas de código** en Visual Basic
- 40 **líneas en de código** en SQL
- 26 **líneas de código** en Small Talk

[Pressman 2006]

Técnicas de descomposición

Ver capítulo 23 libro de texto

- La manera mas simple de estimar está basada en la “descomposición” del software
- Puede descomponerse el software desde dos diferentes punto de vista: descomposición del problema y descomposición de proceso
- La descomposición basada en el problema implica el uso de KLOC y PF.
- La descomposición basada en el proceso incluye división basada en las tareas involucradas, en casos de uso

Ejemplo de Tabla de Estimación de Esfuerzo Orientada al Problema

LCD = LINEAS DE CODIGO **MP = MESES-PERSONA**

FUNCION	OPTIMISTA (KLOC)	PROMEDIO (KLOC)	PESIMISTA (KLOC)	KLOC ESPERADO	COSTO POR LINEA	LINEAS POR MES	COSTO FUNCIÓN	MESES PERSONA (MP)
Control de interfaces con el usuario	1,800	2,400	2,650	2,340	14	315	32,760	7.4
Análisis geométrico de 2 dimensiones	4,100	5,200	7,400	5,380	20	220	107,600	24.4
Análisis geométrico de 3 dimensiones	4,600	6,900	8,600	6,800	20	220	136,000	30.9
Administración de la estructura de datos	2,950	3,400	3,66	3,350	18	240	60,300	13.9
Despliegue de gráficas computacionales	4,050	4,900	6,200	4,950	22	200	108,900	24.7
Control de periféricos	2,000	2,100	2,450	2,140	28	140	59,920	15.2
Análisis del diseño	6,600	8,500	9,800	8,400	18	300	151,200	28.0
TOTAL				**33,360**			**$656,680**	**144.5**

Explicación contenido de columnas de la tabla de estimación de esfuerzos

- *Función:* Incluye cada una de las principales funciones en que se desglosa el proyecto. El desglose se realiza con el detalle necesario que permita estimar miles de líneas de código (KLOC)
- *Optimista:* Valor mas optimista dado por expertos del KLOC de la función
- *Promedio:* Valor medio del KLOC de la función dado por expertos
- *Pesimista:* Valor mas alto dado por expertos del KLOC de la función
- *KLOC esperado:* (OPTIMISTA + 4*PROMEDIO + PESIMISTA)/6
- *Costo por línea*: Valor promedio monetario de cada línea para esa función
- *Líneas por mes:* Cantidad promedio de producción de líneas
- *Costo Función:* KLOC esperado * costo por línea
- *Meses Persona:* KLOC esperado/ líneas por mes

Tabla de Estimación de Esfuerzo Orientada al Proceso

FUNCIONES	ANÁLISIS DE REQUERIM. MP	DISEÑO MP	CODIFICACIÓN MP	PRUEBAS MP	TOTAL MESES-PERSONA
CIU	1.0	2.0	0.5	3.5	7
AG2D	2.0	10.0	4.5	9.5	26
AG3D	2.5	12.0	6.0	11.0	31.5
AED	2.0	6.0	3.0	4.0	15
DGC	1.5	11.0	4.0	10.5	27
CP	1.5	6	3.5	5	16
AD	4	14	5	7	30
TOTAL	14.5	61	26.5	50.5	152.5
COSTO/UNIDAD	$ 5,200	$ 4,800	$ 4,250	$ 4,500	
COSTO TOTAL	$ 75,400	$ 292,800	$ 112,625	$ 227,250	$ 708,075

Factores Que Afectan el Precio del Software

FACTOR	DESCRIPCION
Oportunidad de Mercado	Una organización en desarrollo puede escoger un precio bajo debido a que se está cambiando al área del mercado de sw. Pocas ganancias ahora puede significar mejores ganancias después
Incertidumbre en la estimación del software	Si una organización no está segurá de su estimación de costos, puede incrementar sus precios
Términos contractuales	Un cliente puede permitir al desarrollador que mantenga la posesión del código fuente para re-usarlo en otros proyectos. En este caso el precio cargado sería menor que si el cliente mantiene la posesión del software
Volatilidad en los requerimientos	Si es probable que cambien los requerimientos, la empresa podría cargar precios bajos a fin de ganar un contrato, y compensar después en los cambios.
Salud Financiera de la empresa	Si el desarrollador tiene problemas financieros puede bajar sus precios a fin de ganar la venta. Es mejor ganar poco o no ganar que salir del mercado

[Sommerville 96]

Técnicas de Estimación de Costos

TÉCNICA	DESCRIPCIÓN
Modelado algorítmico del costo	Se desarrolla un modelo usando información histórica relacionada a alguna métrica de software (usualmente tamaño) al costo del proyecto. Se estima la métrica y el modelo predice el esfuerzo requerido.
Juicio Experto	Se consultan varios expertos en el dominio de la aplicación y en la técnica de desarrollo de software escogida. La estimación puede realizarse varias veces hasta que llegan de acuerdo.
Estimación por analogía	Está técnica es útil si se han realizado otros proyectos en el mismo dominio de la aplicación.
La ley de Parkinson	Esta ley establece que el trabajo se expande hasta llenar el tiempo disponible. El costo se determina entonces de acuerdo a los recuros disponibles, más que a los objetivos establecidos. si el software ha de entregarse en 12 meses y hay 5 personas involucradas entonces en esfuerzo requerido se estima como 60 personas-mes
Precio a ganar	El costo se estima de acuerdo a lo que el consumidor está dispuesto a gastar. El esfuerzo sestimado depende del presupuesto del consumidor y no de la funcionalidad del software.

[Sommerville 96]

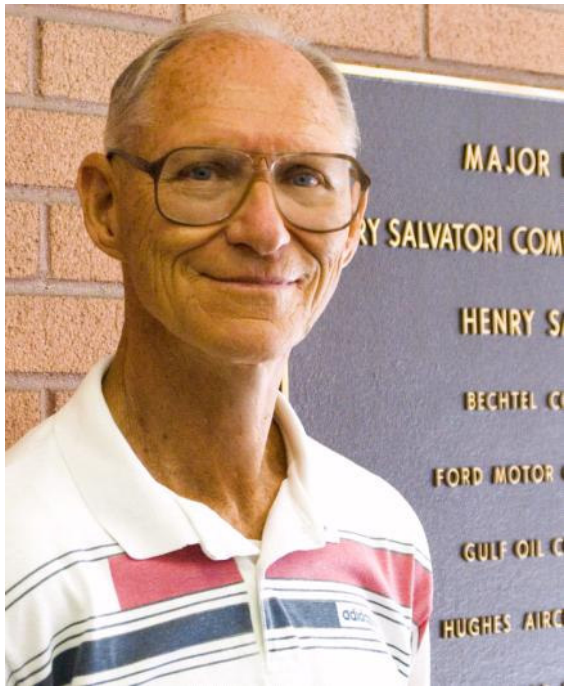
Modelos empíricos de estimación

- Basados en datos estadísticos
- La mayoría tiene una estructura con la forma:

$$E = A + B * (e_v)^C$$

- Donde A,B y C son constantes derivadas empíricamente, E es el esfuerzo en meses persona y e_v es la variable de estimación (LOC o PF)
- Hay varios de estos modelos, uno de los mas populares ha sido el creado por Bohem, COCOMO (*Constructive Cost Model*). Apareció en los años 80, y desde entonces ha sido muy popular

El Dr. Bohem



Tomado de: http://csse.usc.edu/csse/about/people/faculties/BarryBoehm/BarryBoehm.html

Tipos de Modelos COCOMO

- Básico
- Intermedio
- Avanzado
- COCOMO II

Tipos de proyectos en COCOMO

- Dentro de cada modelo COCOMO los proyectos se pueden clasificar de 3 tipos,. Los tipos son:
  - **Orgánico** (Fácil): Proyectos desarrollados con grupos de trabajo pequeños, en un ambiente familiar y construyendo aplicaciones que les son familiares.
  - **Semi-independiente** (Intermedio): Etapa intermedia entre proyectos orgánicos y de modo incorporado.
  - **De modo incorporado** (Avanzado): Proyectos que deben operar dentro de limitaciones estrictas.

- Dependiendo del tipo de proyecto, serán los valores de las constantes que utilizará la fórmula de COCOMO involucrada

Modelo básico de COCOMO

- El modelo calcula 3 valores para estimar el costo del proyecto, esto utilizando como entrada las líneas de código estimadas. Los valores estimados son:
  - MP: Meses-persona
  - TDES: Tiempo de desarrollo
  - N: Número de personas necesarias

- Las fórmulas utilizadas para realizar esta estimación, dependerán del tipo de proyecto en cuestión

MODELO BÁSICO DE COCOMO

PROYECTOS TIPO ORGÁNICO:

MP= [2.4 (KLOC)$^{1.05}$]

TDES= 2.5 (MP) $^{0.38}$

N= MP/TDES

KLOC = Miles de líneas de código

PROYECTOS TIPO SEMI-INDEPENDIENTE:

MP= 3.0 (KLOC)$^{1.12}$

TDES= 2.5 (PM)$^{0.35}$

N= MP/TDES

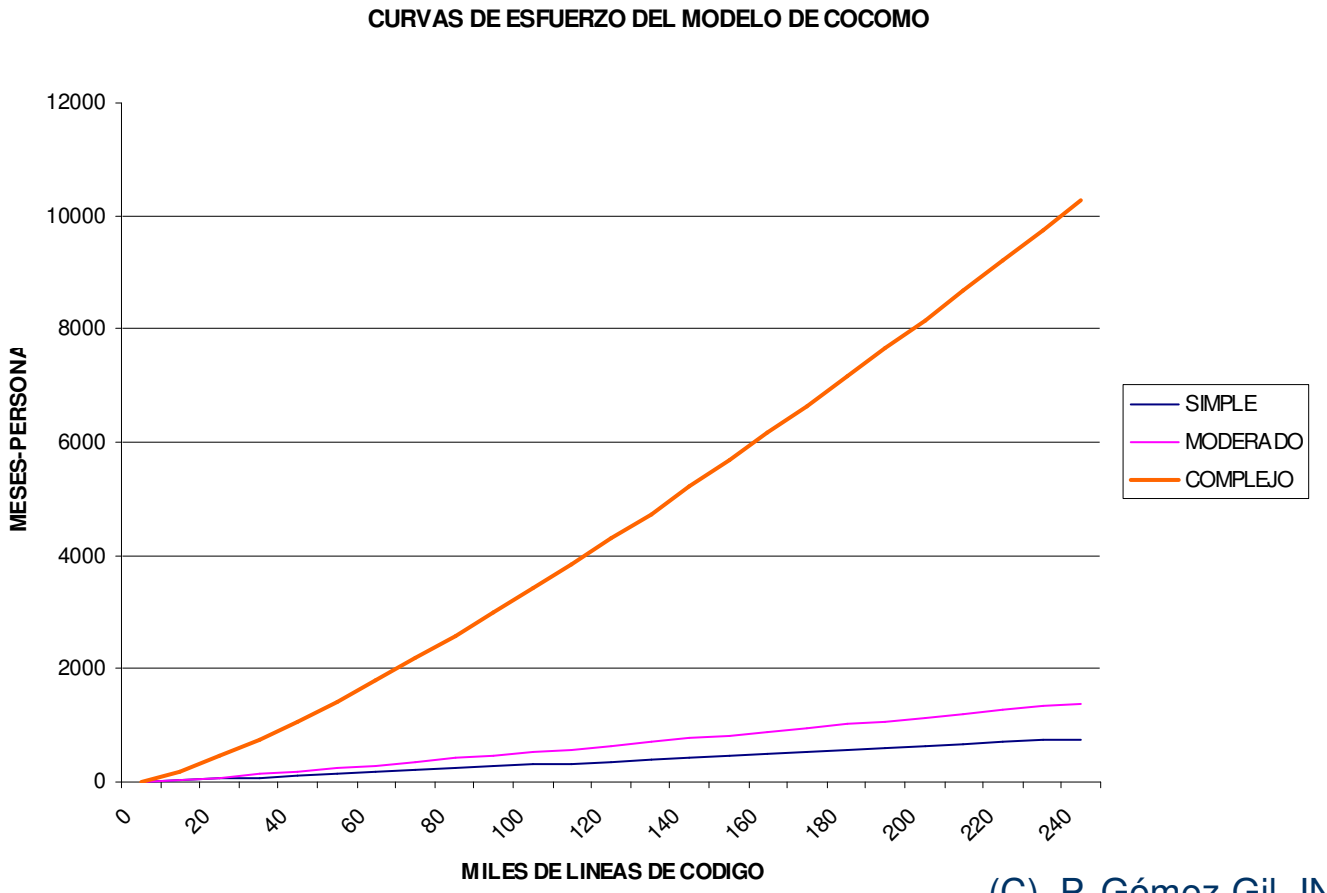
PROYECTOS TIPO INCORPORADO

PM= 3.6 (KLOC)$^{1.20}$

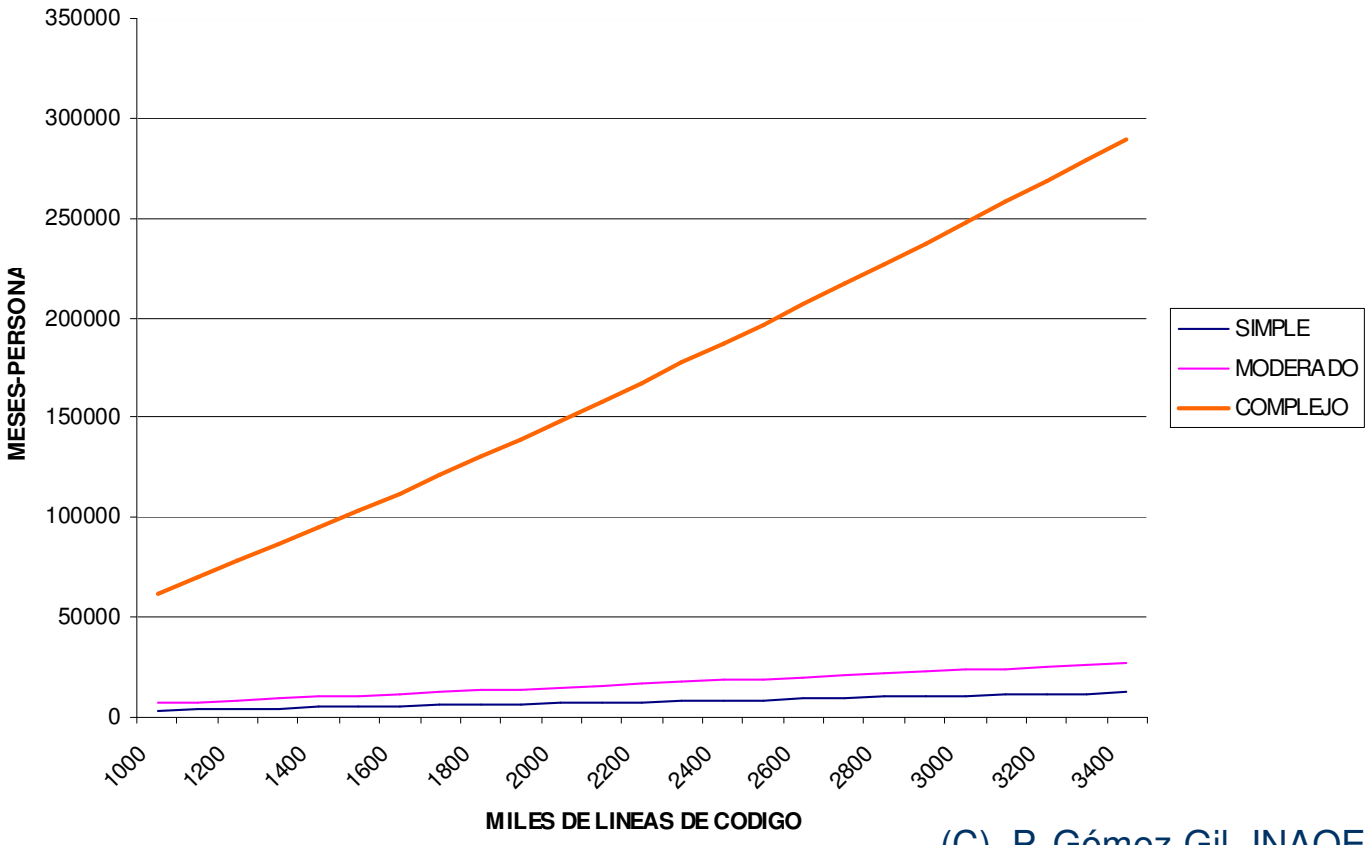
TDES= 2.5 (PM)$^{0.32}$

N= MP / TDES

Curvas de esfuerzo del modelo de COCOMO



Curvas de esfuerzo del modelo de COCOMO



El modelo intermedio de COCOMO

- Modifica las ecuaciones de estimación añadiendo un parámetro multiplicador, el cual será calculado en base a una tabla que evalúa la complejidad añadida debido a otros atributos asociados al proyecto.
- Las formulas entonces quedan de la forma:

$$E = \mathrm{FAE} * B * (e_v)^C$$

- Donde FAE = producto de multiplicadores y es la multiplicación de los valores de la tabla escogidos para cada atributo

Modelo Intermedio De COCOMO. Multiplicadores utilizados

ATRIBUTOS DEL PRODUCTO

1. Confiabilidad requerida en el SW (RELY)
2. Tamaño de base de datos (DATA)
3. Complejidad del producto (CPLX)

ATRIBUTOS COMPUTACIONALES

1. Limitantes del tiempo de ejecución (TIME)
2. Limitantes de almacenamiento (STOR)
3. Volatilidad de la máquina virtual (VIRT)
4. Tiempo de respuesta computacional (TURN)

Modelo Intermedio De COCOMO. Multiplicadores utilizados (continuación)

ATRIBUTOS DEL PERSONAL

1. Capacidad del analista (ACAP)
2. Experiencia en la aplicación (AEXP)
3. Experiencia en la máquina virtual (VEXP)
4. Capacidad del programador (PCAP)
5. Experiencia en el lenguaje de programación (LEXP)

ATRIBUTOS DEL PROYECTO

1. Practicas modernas de programación (MODP)
2. Herramientas de SW (TOOL)
3. Calendario de desarrollo requerido (SCED)

VALORES DE LOS MULTIPLICADORES DEL MODELO INTERMEDIO DE COCOMO

I.D.	muy bajo	bajo	normal	alto	muy alto	extra alto
RELY	0.75	0.88	1	1.15	1.4	---
DATA	---	0.94	1	1.08	1.16	---
CPLX	0.7	0.85	1	1.15	1.3	1.6
TIME	---	---	1	1.11	1.3	1.66
STOR	---	---	1	1.06	1.21	1.56
VIRT	---	0.87	1	1.15	1.3	---
TURN	---	0.87	1	1.07	1.15	---
ACAP	1.46	1.19	1	0.86	0.71	---
AEXP	1.29	1.13	1	0.91	0.82	---
PCAP	1.42	1.17	1	0.86	0.7	---
VEXP	1.21	1.1	1	0.9	---	---
LEXP	1.14	1.07	1	0.95	---	---
MODP	1.24	1.1	1	0.91	0.82	---
TOOL	1.24	1.1	1	0.91	0.83	---
SCED	1.23	1.08	1	1.04	1.01	---

← importancia

PRESSMAN 92

Para el proyecto en clase...

- Utilizar el Modelo Intermedio, con proyecto MODO ORGÁNICO, esto es, usar las fórmulas:

$MP = [2.4 * FAE * (KLOC)^{1.05}]$

KLOC = Miles de líneas de código

$TDES = 2.5 (MP)^{0.38}$

$N = MP/TDES$

COCOMO II

Ver

Center for Systems and Software Engineering – COCOMO II

http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

Calendarización de proyectos de software

- Consiste en establecer detalladamente las fechas en que se realizarán actividades y se entregarán documentos y software, así como la asignación de quienes serán responsables de realizar las actividades

Para el proyecto

- Evaluar la formula del modelo basico, usando el caso del proyecto tipo orgánico

¿PORQUÉ SE ENTREGA TARDE EL SOFTWARE?

- Fechas límite de entrega poco realistas
- Cambios en los requerimientos
- Sub-estimación en la planeación de recursos
- Errores no considerados
- Dificultades técnicas no predecibles
- Dificultades humanas no predecibles
- Falta de comunicación en el equipo
- Falta de reconocimiento por parte de la administración de retrasos y falta de medidas para corregir el problema.

"How does a project get to be a year behind schedule? One day at a time." !.

FRED BROOKS. (Autor de The Mythical Man-Month, 1995)

Principios básicos para calendarizar

- Divide el proyecto en tareas manejables
- Establece interdependencias entre las tareas
- Determina el tiempo de manera realista! (ni mas, ni menos)
- Determina el esfuerzo involucrado en cada tarea
- Determina responsabilidades
- Determina entregables
- Determina puntos clave de inicio/terminación de fases

Herramientas

- Diagramas de Gantt
- Diagramas PERT/CPM
- Redes de tareas y tablas de recursos

- Hay muchos paquetes automáticos para control de proyectos...

Diagramas de Gantt

- Manera simple de definir actividades
- Relaciona actividades, entregables e hitos con tiempo de desarrollo/ejecución
- Puede incluir responsables

Ejemplo de calendario

- Ver el calendario del proyecto...

Consideraciones para la estimación del proyecto de Clase.

- CON RESPECTO A MESES PERSONA DEL PROYECTO
  - Durante el semestre se va a desarrollar básicamente el análisis y diseño de un incremento. Este trabajo equivale aproximadamente al 50% del tiempo total del incremento.
  - Los meses-persona de que se dispone para esta parte del proyecto esta dado por:

    (# de estudiantes en el equipo) *(horas a la semana dedicadas al proyecto)*(15 semanas)/(140 hrs. de trabajo por mes) = 5*3*15/140 = **1.6 meses persona**
  - Esto es el 50% del proyecto, entonces el proyecto a realizarse es de 1.6/50% = **3.2 meses persona.**
  - Si consideramos +/- 20% posibilidad de error entonces el resultado de puntos funcionales debe estar entre **2.56 y 3.84 meses persona**

Consideraciones para la estimación del proyecto de Clase. (cont.)

- Supongamos un salario de **$7,100 pesos al mes** tiempo completo, y que dicha persona produce alrededor de 500 líneas de código al mes. (esto, como es una métrica que incluye no solo la programación, sino todo el proceso.).

  Entonces en 3.2 MP se pueden producir

  3.2 * 500 = **1,600 líneas de código en total aproximadamente (+ / - 20%)**

- 1,600 líneas de código escritas en C, equivalen a 1600*66 = **24.2 Puntos funcionales como total que debería salir (**+/- 20% posibilidad de error: entre 19.36 y 29.04 PF)

Calidad de Software

- Se define como la “congruencia con requerimientos funcionales y de desempeño explícitamente establecidos, estándares de desarrollo explícitamente documentados y características implícitas que se esperan de todo software desarrollado profesionalmente”

[Pressman 2005]

Factores de calidad

- Los factores que afectan la calidad pueden dividirse en aquellos que pueden medirse de forma directa y los que solo pueden medirse indirectamente.
- Asimismo, pueden clasificarse de acuerdo a diferentes aspectos del producto.

Factores de Calidad de la ISO 9126

- **Funcionalidad**: Grado al cual el software satisface las necesidades establecidas. Puede identificarse a través de los atributos: *Exactitud, interoperabilidad, seguridad, conformidad*
- **Confiabilidad:** Cantidad de tiempo en que el software está disponible para usarse, según lo indicado por los siguientes atributos: *madurez, tolerancia a fallas y capacidad de recuperación*
- **Usabilidad:** Grado al cual es software es fácil de usar según lo indicado por los siguientes atributos: *capacidad de entenderse, capacidad de aprenderse, capacidad de operarse*.

Continúa…

Factores de Calidad de la ISO 9126 (2)

- **Eficiencia**: grado al cual el software hace uso óptimo de los recursos del sistema, según indicado por los atributos: *comportamiento de tiempo, comportamiento de recursos*
- **Mantenibilidad**: Facilidad con la que el software puede repararse, según lo indicado por los atributos: *facilidad de análisis, facilidad de cambio, estabilidad, facilidad de pruebas*
- **Portabilidad**: Facilidad con la que el software puede transportarse de un medio ambiente a otro, según lo indicado por los atributos: *adaptabiidad, capacidad de instalación, capacidad de adecuación y capacidad de reemplazo*

Planeación de Software: Estimación de recursos

Ver capitulo 23 libro de texto

Los recursos involucrados son:

- Hardware, software y Personal

Para cada recurso se debe especificar:

- Descripción del recurso
- Disponibilidad del recurso
- Tiempo cronológico en el que se utilizará
- Tiempo de uso del recurso

Planeación de Software: Estimación de recursos (cont.)

Especificar además:

Con respecto a **Personal:**

- Habilidades requeridas

Con respecto a **Hardware:**

- Máquina de desarrollo
- Máquina final
- Hardware adicional requerido

Con respecto a S**oftware:**

- Software a re-utilizarse
- Herramientas CASE* a utilizarse

*CASE= Ing. de Software Asistida por Computadora

Contenido de un Documento de Planeación de Proyectos de Software

1. Introducción
    - 1.1 Propósito del documento
    - 1.2 Identificación del problema
    - 1.3 Objetivos Generales
    - 1.4 Funciones principales el proyecto
    - 1.5 Limitantes técnicas y administrativas
2. Estimación de recursos
    - 2.1 Datos históricos utilizados para estimar
    - 2.2 Técnicas de estimación
    - 2.3 Estimaciones
        - 2.3.1 Esfuerzo
        - 2.3.2 Costos
3. Calendarización
    - 3.1 Diagrama PERT
    - 3.2 Diagrama de Gantt
    - 3.3 Tabla de recursos

Para el proyecto en clase:
2.3.1 incluye Puntos Funcionales, las tablas de estimación orientadas a procesos y KLOC
2.3.2 incluye el modelo intermedio de COCOMO
3.1 y 3.3 no se incluyen

Contenido de un Documento de Planeación de Proyectos de Software (continuación)

4. Recursos del proyecto
    - 4.1 Personal Involucrado
    - 4.2 Contratos externos
    - 4.3 Hardware
    - 4.4 Software
    - 4.5 Recursos especiales

5. Organización del equipo de trabajo
    - 5.1 Estructura del equipo
    - 5.2 Reportes de administración
    - 5.3 Mecanismos de control

BIBLIOGRAFIA

- **Jensen, Randall W. and Charles C. Tonies**. Software Engineering. Prentice-Hall. New Jersey 1979
- **Roger S. Pressman**. Ingeniería de Software. Un enfoque práctico Mc Graw Hill, 2002
- **Roger S. Pressman.** Software Engineering, A practitioner's approach. Sixth Edition, 2005
- **Sommerville, Ian**. Software Engineering. Fifth Edition. Addison-Wesley, 1996