ADMINISTRACIÓN DE PROYECTOS INFORMÁTICOS

COCOMO

(COnstructive COst MOdel)

El modelo COCOMO

- Los costos se pueden calcular como una función matemática basada en atributos de productos, proyectos y procesos, cuyos valores son calculados por administradores de proyectos.

- La función se basa en un estudio histórico de datos de costos.

- LOC (tamaño de código) utiliza estimación de costos

El modelo COCOMO

- Fue desarrollado y presentado en 1981 por Barry W. Bohem.

- Se enmarca en el grupo de los modelos algorítmicos que tratan de establecer una relación matemática que permita estimar el esfuerzo (hombre-mes) y tiempo requerido para desarrollar un proyecto.

- Basado en una base de datos de costos (con más de 60 proyectos diferentes).

El modelo COCOMO

- Existen tres niveles:

Nivel Básico

Nivel Intermedio

Nivel Detallado

- Existen tres modelos:

Orgánico

Semilibre

Fuertemente restringido

El modelo COCOMO

Nivel Básico

* Es adecuado para realizar estimaciones de forma rápida aunque sin gran precisión.

* No tiene en cuenta los diferentes atributos que afectan al proyecto como: calidad, experiencia del personal, restricciones de hardware, utilización de técnicas modernas y herramientas de desarrollo.

El modelo COCOMO

Nivel Intermedio

* Los factores antes mencionados se consideran como adicionales al costo total del proyecto.

Nivel Detallado

* Se considera cómo estos factores afectan dentro de las diferentes fases individuales que componen el proyecto.

El modelo COCOMO

- El factor principal sobre el que se basan las estimaciones es el tamaño del producto, es decir, el *número de instrucciones fuente desarrolladas.*

- La cantidad de instrucciones fuente se deben estimar por experiencia, por analogía con otros proyectos semejantes, o por otros datos que se posean.

El modelo COCOMO

- En el modelo de desarrollo de software se planifican solo las fases comprendidas desde el análisis hasta la implantación, (La fase de estudio preliminar no se considera).

- Los parámetros estimados no incluyen los correspondientes a las actividades de formación de los usuarios, planificación de las instalaciones y trabajos de conversión.

El modelo COCOMO

- Los indicadores de planificación que se pueden obtener con este método son:

- **Esfuerzo** (hombre-mes)
- **Tiempo de desarrollo** (meses)
- **Personal necesario** (hombres)
- **Productividad** (inst/hombre-mes)
- **Costo** (pesos)

El modelo COCOMO

La unidad de esfuerzo Hombre-Mes supone un total de 152 horas de trabajo por persona, en base a la experiencia práctica y a consideraciones sobre vacaciones, permisos, enfermedad, etc.

Hombres-Mes x 152 = Hombres-Hora

Hombres-Mes x 19 = Hombres-Día

Hombres-Mes / 12 = Hombres-Año

El modelo COCOMO

Modelos de desarrollo de software:

Modelo Orgánico o Familiar

Modelo Semilibre

Modelo Fuertemente restringido

Modelo Orgánico o Familiar (1)

- El equipo de desarrollo es relativamente pequeño y se desenvuelven en un entorno altamente familiar.
- La gran mayoría de la gente relacionada con el proyecto tiene una amplia experiencia en otros proyectos relacionados con la misma organización
- Tienen un buen conocimiento de cómo el sistema bajo desarrollo, contribuirá a los objetivos de su organización.

Modelo Orgánico o Familiar (2)

- La mayoría de las personas pueden contribuir de forma efectiva a la terminación puntual de cada una de las etapas sin generar grandes necesidades de comunicación para determinar con precisión las tareas que cada uno debe desarrollar en el proyecto
- El equipo de trabajo puede negociar con facilidad la modificación de algunas de las especificaciones para hacer más fácil este desarrollo.

Modelo Orgánico o Familiar (3)

- Entorno de desarrollo estable, con poco desarrollo concurrente de nuevo Hardware asociado.
- Mínimas necesidades de introducir algoritmos innovadores o nuevas arquitecturas de proceso.

- Un trabajo de proyecto relativamente pequeño. Muy pocos proyectos desarrollados de modo orgánico sobrepasan los 50 MF (50 000 instrucciones fuente).

- Proyectos en Modo Orgánico de mayor tamaño pueden desarrollarse utilizando software ya existente.

Modelo Semilibre (1)

- Representa un estado intermedio entre el modo orgánico y el modo fuertemente restringido
- Todos los miembros del equipo de diseño tienen un nivel medio de experiencia en sistemas relacionados con el proyecto
- El equipo de desarrollo esta formado por una mezcla de gente experta e inexperta.

Modelo Fuertemente restringido (1)

- Debe desarrollarse sometido a fuertes restricciones.
- El producto debe operar en entornos de software y hardware fuertemente acoplados.
- En estos proyectos no existe la posibilidad de negociar fácilmente cambios en el software y en tal caso precisará un mayor tiempo para acomodar o asegurar que los cambios cumplan las especificaciones (mayor costo de verificación, validación y de gestión de la configuración).

Comparación del Ciclo de Vida

Bohem	Nuestro
Planificación y Requisitos	Estudio Preliminar
Diseño	Análisis
Diseño detallado	Diseño
Codificación y Prueba	Desarrollo
Integración y Prueba	Prueba e implantación

COCOMO Básico

$E = a_b(KLOC)^b b$

E= esfuerzo (hombre/mes)

KLOC= número (miles) estimado de líneas de código del proyecto.

COCOMO Básico

Proyecto de software	a_b	b_b	c_b	d_b
Orgánico	2.4	1.05	2.5	0.38
Semi-acoplado	3.0	1.12	2.5	0.35
Fuertemente restringido	3.6	1.20	2.5	0.32

COCOMO Básico

Supongamos que una empresa cualquiera desea diseñar un proyecto que gestione sus inventarios y decide desarrollarlo mediante su propio equipo de analista y programadores que anteriormente y durante muchos años, vienen desarrollando aplicaciones similares en la misma empresa.

Si un estudio inicial determina que el tamaño del producto en alrededor de 32 000 líneas de programa fuente (32 KLOC). Cuales serán las características del proyecto?.

COCOMO Básico

- Esfuerzo:

$E = a_b (KLOC)^{b} b$

$E = 2.4\ (32)^{1.05}$ $E = 91$ hombres-mes

- Tiempo de desarrollo:

$D = c_b (E)^{d} b$ $D = 2.5\ (91)^{0.38} = 14$ meses

COCOMO Básico

- Número de personas trabajando en el proyecto:

$$N = 91/14 = 6.5 \text{ hombres}$$

La cantidad de hombres nos da una medida del número equivalente de personas trabajando a tiempo completo en el proyecto.

Estimación por Puntos de Función

Puntos de Función (1)

•Establecer el esfuerzo que conlleva el desarrollo de un producto de software, ha sido sin duda una necesidad creciente de la industria informática.

• Muchas métricas se han visto en el transcurso de los años, pero la que ha logrado resultados más relevantes ha sido el conteo de los Puntos de Función.

• Las bases de la métrica Punto de Función, fueron desarrollados en un período superior a los 5 años comenzando en 1974 por el Departamento de Servicios Computacionales de IBM.

Puntos de Función (2)

•Allan J. Albrecht (IBM, White Plains) fue el primero en publicar la técnica de Punto de Función en 1979.

• Desde fines del año 1982 el Punto de Función, ha sido usado en los Estados Unidos, Inglaterra, Nueva Zelandia, Australia y Canadá.

• Algunas Empresa que utilizan la técnica son: IBM, UNISYS, Bank of America, Bell, ITT, Xerox, General Motors, otros.

Puntos de Función (3)

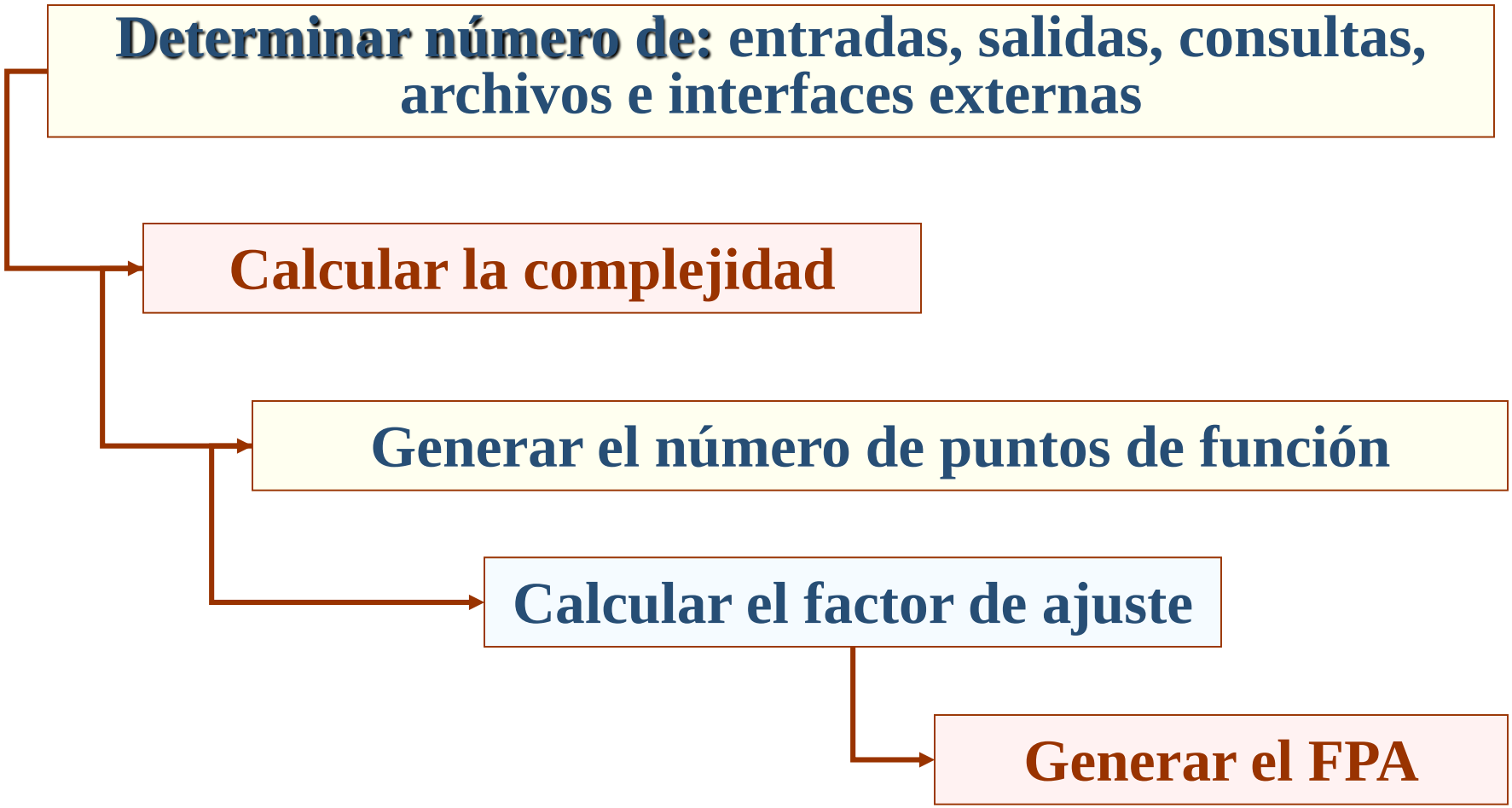
- Los objetivos de los puntos de función son:
  - Medir lo que el usuario pide y lo que el usuario recibe.
  - Medir independientemente de la tecnología utilizada en la implantación del sistema.
  - Proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y la productividad.
  - Proporcionar un medio para la estimación del software.

Puntos de Función (4)

- El análisis de los puntos de función se desarrolla considerando cinco parámetros:
  - Número de Entradas de usuario
  - Número de Salidas de usuario
  - Número de Consultas (peticiones del usuario)
  - Número de archivos
  - Número de interfaces externas

Puntos de Función (5)

- Cálculo:



Puntos de Función (6)

- Entradas de usuario: Se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación.

  - Transacciones : datos introducidos para mantener archivos lógicos internos.
  - Pantallas de entrada: Hay que añadir una unidad por cada función (añadir, cambiar, borrar) que mantiene un archivo lógico interno

Puntos de Función (7)

- Salidas de usuario: Se cuenta cada salida que proporcione al usuario información orientada a la aplicación.

  - Informes
  - Pantallas
  - Mensajes de error/configuración
  - La transferencia de datos a otras aplicaciones
  - Cada gráfico distinto (tabla, diagrama de barras: se cuenta como 2 salidas)

Puntos de Función (8)

- Consultas: Una entrada interactiva que produce la generación de alguna respuesta del software inmediata en forma de salida interactiva.

  - Búsqueda inmediata de datos
  - Tutoriales
  - Las Ayudas
  - Las pantallas de logon que proporcionarían seguridad
  - Las pantallas de menú que proporcionan una selección de pantallas y entradas para la búsqueda de datos para la pantalla llamada.

Puntos de Función (9)

•Número de archivos: se cuenta cada archivo maestro lógico (un grupo lógico de datos que puede ser una parte de una gran base de datos o un archivo independiente).

- Archivos maestros
- Mensajes Help actualizados por la aplicación
- Mensajes de error actualizados por la aplicación
- Archivos lógicos internos mantenidos por más de una aplicación.

Puntos de Función (10)

•Interfaces externas: Se cuentan todas las interfaces legibles por la máquina (archivos de datos de cinta o disco) que se utilizan para transmitir información a otro sistema.

- Bases de datos compartidas
- Archivos lógicos internos utilizados por otra aplicación
- Lista de parámetros compartidos

Líneas de Código y Puntos de Función

La relación entre las líneas de código y los puntos de función depende del lenguaje de programación que se utilice para implementar el software y de la calidad del diseño.

LOC → - Errores por LOC
- $ por LOC
- LOC por persona-mes

PF → - Errores por PF
- $ por PF
- PF por persona-mes

Líneas de Código y Puntos de Función

Lenguaje de programación	LDC/PF (media)
Ensamblador	320
C	128
Cobol	105
Fortran	105
Pascal	90
Ada	70
Java y C++	53
...	...

Ejemplo de Puntos de Función

La Especialización en Ingeniería de Software requiere de un Sistema de Información que permita llevar el control escolar de sus alumnos:

- Se necesita un módulo de Actualizaciones (alumnos, materias y calificaciones) el cual permitirá realizar altas, bajas y cambios para cada uno de ellos.

- Otro módulo es el de consultas (alumnos, materias y calificaciones).

- El último módulo será el de reportes: solo se emitirán los reportes de “alumnos inscritos”, “catálogo de materias” y “alumnos con sus calificaciones”.

Ejemplo de Puntos de Función

- Los archivos que se utilizarán son: alumnos, materias y cardex.

- No. entradas:
  - 3 altas (alumnos, materias y calificaciones)
  - 3 bajas (alumnos, materias y calificaciones)
  - 3 cambios (alumnos, materias y calificaciones
- No. salidas:
  - 3 reportes: “alumnos inscritos”, “catálogo de materias” y “alumnos con sus calificaciones”.

Ejemplo de Puntos de Función

- No. consultas: 3 alumnos, materias y calificaciones

  4 pantalla de menú de selección (pantalla principal, actualizaciones, consultas y reportes)

- No. archivos: 3 alumnos, materias y cardex
- No. interfaces externas: 0

Se considera que el sistema tiene una complejidad media en cada unos de los factores antes mencionados.

Factor de ponderación según complejidad

Tabla 10.1: Factor de ponderación según complejidad

	Simple	Media	Compleja
Entrada externa	3	4	6
Salida externa	4	5	7
Consultas	3	4	6
Ficheros externos	7	10	15
Ficheros internos	5	7	10

Puntos de función
Complejidad media

entradas	9 x 4=36
salidas	7 x 5=35
consultas	3 x 4=12
Archivos lógicos	3 x 10=30
Archivos de interfaz	0 x 7=0

Total		113
Multiplicador	En este ejemplo el multiplicador se da por default. No se calcula.	1.13
Total de PF ajustados		128 PF

Factores de Complejidad Técnica

1- **5** Requiere el sistema copias de seguridad y de recuperación fiables ?
2- **2** Se requiere comunicación de datos ?
3- **0** Existen funciones de procesamiento distribuido ?
4- **4** Es crítico el rendimiento?
5- **4** Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?
6- **4** Requiere el sistema entrada de datos interactiva ?
7- **4** Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?
8- **5** Se actualizan los archivos maestros de forma interactiva ?
9- **5** Son completas las entradas, las salidas, los archivos o las peticiones ?

10- **1** Es complejo el procesamiento interno ?
11- **3** Se ha diseñado el código para ser reutilizable ?
12- **3** Están incluidas en el diseño la conversión y la instalación?
13- **3** Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones ?
14- **5** Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario ?

TOTAL: 48

Ponderación:

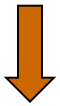
0 No influencia 1 Incidental 2 Moderado 3 Medio

4 significativo 5 Esencial

Esta fórmula utiliza los factores de complejidad técnica para calcular los puntos de función Ajustados

PFA = cuenta-total x (0.65 + 0.01 (f1+f2+…+f14))

PFA = 113 x (0.65 + 0.01 (48))



PFA = 128

Si el lenguaje de programación utilizado es Java, entonces

LOC = 53*PFAjustados = 53 * 128 = 6784

y esfuerzo = PFA/ productividad

Tomando en cuenta la productividad = 12 PFA/ personas mes

Esfuerzo = 128 / 12 = 10.66 personas/mes