Planificación y Gestión de Sistemas de Información

Tema 8

GESTIÓN DE COSTES EN PROYECTOS SOFTWARE

Objetivos

- Ampliar los conocimientos elementales de la gestión de costes ya estudiados en la introducción a la gestión de proyectos.
- Conocer las aspectos principales que se deben tener en cuenta al planificar los recursos necesarios en un proyecto software.
- Estudiar las principales técnicas utilizadas en ingeniería del software para estimar los costes de un proyecto.

Indice

Bibliografía básica:

- Boehm, B. y otros. "*An Overview of the COCOMO 2.0 Software Cost Model*", Software Technology Conference, 1995. En file://sunset.usc.edu/research/COCOMOII/Docs/stc.pdf, recuperado el 15-3-2000.

- Center for Software Engineering, University of Southern California. *COCOMO II Model Definition Manual*. En ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo98.0/modelman.pdf, recuperado el 15-3-2000.

- Connell, S. *Desarrollo y Gestión de Proyectos Informáticos*. McGraw-Hill Interamericana. España 1997.

  Cap. 8.

- Project Management Institute, *A Guide to the Project Management Body of Knowledge*, PMI Communications, USA 1996.

  Cap. 7.

Introducción

- conceptos y técnicas que vamos a ver, lugar que ocupan en el marco PMI de Gestión de Proyectos.

Área	**Proceso**	**Grupo**	**Conceptos, técnicas y herramientas** **C=conceptos, T=técnicas y herramientas, O=salidas**
Gestión de los Costes	Planificación de Recursos	Planificación	
	Estimación de Costes	Planificación	C: Costes Unitarios
	Realizar presupuesto de costes	Planificación	T: Estimación por Analogía (top-down) T: Estimación Bottom-up T: Modelos paramétricos (COCOMO) O: Línea base de costes (cost baseline)
	Control de Costes	Control	O: Estimación de costes a la conclusión

  - C: conceptos que amplian y extienden lo comentado del proceso en el tema IV, pero ahora particularizando en proyectos informáticos y especialmente proyectos software (PS).
  - T: técnicas y herramientas útiles en el proceso.
  - O: salidas (outputs), es decir, resultados del proceso.

Gestión de Costes:Introducción

- *Objetivo*:

  asegurar que el proyecto es completado dentro del presupuesto previsto

- *Procesos*:
  - **Planificación de recursos**: determinar los recursos (personas, equipos, materiales) y las cantidades de cada uno necesarias para realizar las actividades.
  - **Estimación de costes**: realizar una aproximación (estimación) de los costes de los recursos necesarios para completar las actividades del proyecto.
  - **Realizar presupuesto de costes**: calcular el coste global estimado de cada tarea individual.
  - **Control de costes**: controlar los cambios en el presupuesto del proyecto.
- En algunos *proyectos pequeños*, la planificación de recursos, la estimación de costes, y la realización del presupuesto se pueden ver como un único proceso, realizado por una única persona en un período de tiempo reducido.

Planificación de Recursos

- Entradas:
  - *WBS* (diagramas de descomposición de trabajos), DFT (diagramas de flujos de trabajo): para identificar los elementos del proyecto que necesitan recursos.
  - *Información histórica*: tipos de recursos requeridos en proyectos anteriores similares.
  - *Declaración del alcance*: es importante tener en cuenta la justificación y los objetivos básicos del proyecto.
  - *Relación de recursos disponibles*: permite conocer los recursos que están potencialmente disponibles para su utilización en el proyecto.
  - *Políticas organizacionales*: referidas a temas de personal y de adquisición de suministros y equipamiento.
- Herramientas y Técnicas:
  - Juicio de expertos,
  - Identificación de alternativas.
- Salidas:
  - *Recursos Necesarios*: descripción de los tipos de recursos requeridos y en qué cantidad para cada elemento del WBS.

Estimación de Costes

- Entradas:
  - WBS, DFT, ...
  - Recursos Necesarios: obtenido en el proceso anterior.
  - **Costes unitarios**: tasas de coste por unidad de recurso utilizado (coste de una hora de programador, de un litro de combustible, coste mensual de amortización de un PC, ...).
  - Estimación de la duración de las actividades.
  - Información Histórica: referida a los costes de las diversas categorías de recursos.
- Técnicas de Estimación:
  - Estimación por analogía.
  - Modelos paramétricos.
  - Estimación Bottom-up.

Técnicas de estimación de costes (1)

- Estimación de costes por **Analogía** (o *top-down*):
  - Se calcula el coste actual de un proyecto a partir del coste de otro similar.
  - Suele emplearse cuando no se dispone de información suficientemente detallada del proyecto.
  - Es una forma de juicio de experto.
  - Es menos fiable que otras técnicas.
  - Son necesarias dos condiciones:
    - Los proyectos previos deben ser similares de verdad y no en apariencia, y
    - Las personas que realizan la estimación deben ser experimentados.
- Estimación de costes ***Bottom-up***:
  - Se estima el coste de paquetes de trabajo individuales, para a continuación, mediante agregación estimar el total del proyecto.
  - A menor tamaño de los paquetes de trabajo, mayor dificultad de la estimación pero también se obtiene una mayor exactitud.

Técnicas de estimación de costes (2)

Modelos **paramétricos**:

- Utilizan determinadas características del proyecto (**parámetros**) para predecir los costes del proyecto mediante un modelo matemático.
- Dependiendo de la naturaleza del proyecto, el modelo matemático será sencillo o complejo. Ejemplos:
  - *Sencillo*: coste de una vivienda nueva = superficie en m2 x 140000 pts/m2
  - *Complejos*: los modelos de estimación del software pueden utilizar docenas de factores y parámetros diferentes (ejemplo: COCOMO).
- La dificultad y exactitud de los modelos paramétricos son muy variadas.
- Son mas *fiables* cuando:
  - La información histórica utilizada para desarrollar el modelo era exacta,
  - Los parámetros utilizados son cuantificables sin dificultad,
  - El modelo es escalable (funciona para proyectos de diferentes tamaños y/o complejidades).

Medida de los costes

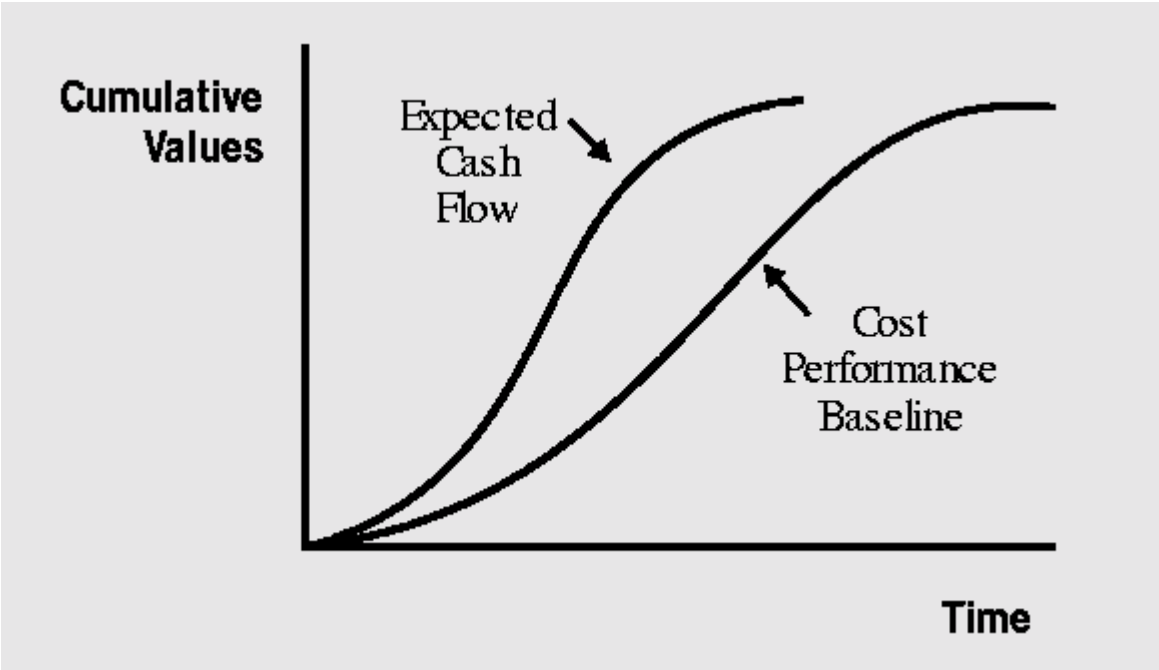
- Se deben estimar los costes de **todos los recursos** que se dedicarán al proyecto:
  - mano de obra, materiales, equipamientos, suministros, servicios, etc.
- **Unidades de medida**:
  - Generalmente se utilizan unidades *monetarias* para permitir comparaciones.
  - También se pueden utilizar otras unidades de *tipo horario* (horas de trabajo x persona), aunque tienen el riesgo de no estandarizar los costes y dificultar las comparaciones (una hora de administrativo no cuesta igual que una hora de ingeniero).
- En algunos campos, es habitual que la estimación de costes tenga *distintos niveles de refinamiento* a lo largo del ciclo de vida del proyecto. Por ejemplo, en la ingeniería civil, se hacen cinco estimaciones: orden de magnitud, conceptual, preliminar, definitiva y control.
- Es frecuente que los costes se indican con un *intervalo de variación*: 6000$ ± 500$

Línea Base de Costes (*cost baseline*) (1)

- Representa el reparto del presupuesto a lo largo del tiempo de duración del proyecto.
- Sirve para medir y supervisar la evolución de los costes a lo largo de la realización del proyecto.
- Se calcula con:
  - los datos de estimación de costes de todos los paquetes de trabajo,
  - el WBS/DFT, y
  - el calendario del proyecto (con las fechas de inicio y fin de todas las actividades).
- Permite resumir gráficamente los costes estimados en cada periodo.
- Se puede construir una línea base de costes para cada categoría de costes (personal, servicios, etc.) o para un recurso individual.

Línea Base de Costes (*cost baseline*) (2)

- Suele tener forma de ‘S’:



Control de Gastos

- Incluye las siguientes acciones:
  - Supervisar la realización de gastos para detectar variaciones respecto de lo previsto.
  - Asegurar que todos los cambios son registrados con exactitud.
  - Prevenir que cambios incorrectos, inapropiados, o no autorizados, sean incluidos.
  - Informar adecuadamente a los clientes de los cambios autorizados.
  - Los "porqués" de las variaciones, positivas o negativas.
- Las **salidas** (*outputs*) de este proceso son:
  - *Revisiones de las estimaciones* de costes.
  - *Modificaciones del presupuesto*.
  - *Acciones correctivas*.
  - *Lecciones aprendidas*: debe crearse una base de datos histórica documentando las causas de las variaciones, las razones de elegir una determinada acción correctiva, etc.
  - ***Estimaciones de costes a la conclusión***.

Estimaciones de Costes a la Conclusión

- Son previsiones de los costes totales del proyecto basados en los gastos realizados hasta la fecha.
- Existen tres tipos de cálculos:
  - EAC = CA + PRP * FC,

    siendo CA los costes actuales (hasta la fecha), PRP el presupuesto restante del proyecto, y FC un factor de corrección para tener en cuenta las desviaciones producidas hasta el momento (relación entre los gastos actuales y los planificados hasta la fecha).
  - EAC = CA + NEP,

    siendo NEP una nueva estimación realizada para todo el trabajo pendiente.
  - EAC = CA + PRP
- IEEE y PMI han definido un **estándar** para realizar el seguimiento de un proyecto software y poder extrapolar, tanto su coste como su duración. Está basado en el concepto de **Sistema de Valor Conseguido** (*Earned Value System*).

  Ver cap. 10 de Dolado, J. J. y Fernández, L., Medición para la Gestión en la Ingeniería del Software.

Estimación del Software

- El proceso de estimación del software se puede dividir en tres etapas:
  1. Estimar el **tamaño** del producto (en número de líneas de código o en puntos función). Es la etapa más compleja.
  2. Estimar el **esfuerzo** (en personas-día o similar) a partir de la estimación del tamaño y datos previos de la organización en proyectos similares.
  3. Estimar la **planificación** (calendario).

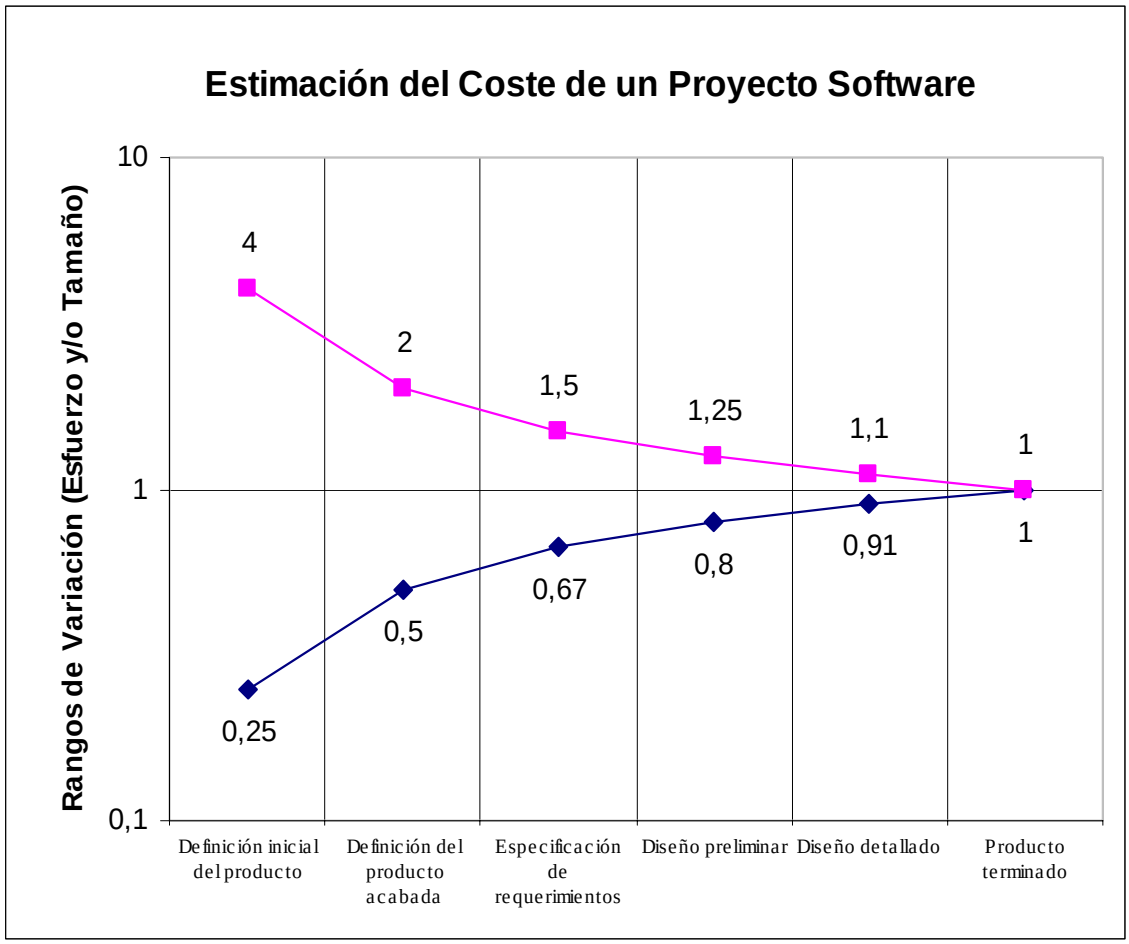
- Estas tres etapas se pueden englobar en una **etapa general**, consistente en: *dar una estimación con un cierto margen de desviación e ir aumentando la precisión (reduciendo el margen) a medida que avanza el proyecto.*
- Por tanto, la estimación del software es un proceso basado en **refinamientos sucesivos**.

Refinamiento en proyectos software (1)

- la **estimación** del software es un proceso basado en **refinamientos sucesivos** porque:
  - No se puede estimar con precisión el coste de un producto software hasta que se comprende con detalle cada una de sus prestaciones.
  - A lo largo del ciclo de vida del desarrollo de un producto software se van tomando decisiones cada vez más detalladas.
  - El concepto del producto se refina en la fase requerimientos, los requerimientos en el diseño preliminar, el diseño preliminar en el diseño detallado y el diseño detallado en el código.
  - En cada una de estas fases se toman decisiones que afectan al coste global del producto.
  - La incertidumbre sobre la naturaleza del producto aporta incertidumbre a la estimación.
  - La incertidumbre sobre una única prestación puede introducir bastante duda sobre la estimación inicial del proyecto.
  - Conforme aumenta el porcentaje de decisiones tomadas, se puede afinar el rango de la estimación.

Refinamiento en proyectos software (2)

- Las estimaciones en proyectos software tienen rangos de precisión previsibles, que se van reduciendo a lo largo de la duración del proyecto:
  - Con la definición inicial del producto la oscilación puede ser de 1 a 16,
  - Después de la especificación de requerimientos la oscilación es de 1 a 2'2, ...



Estimación del Tamaño

- El **tamaño** de un producto software es un **indicador** *de la amplitud y profundidad del conjunto de prestaciones que incorpora, así como de la complejidad y dificultad del programa*.

- Se puede estimar el tamaño de un producto software de varias maneras, utilizando alguno de los tres tipos de técnicas generales de estimación de costes ya comentadas:
  - Estimación por *Analogía*: estimar el tamaño del programa a partir de la descripción de sus características principales. Para ello se suelen emplear herramientas de estimación que localización automáticamente proyectos similares en una base de datos de proyectos que incluyen.
  - Estimación *Bottom-up*: estimar cada una de las partes principales del nuevo sistema como un porcentaje del tamaño de una parte similar de un sistema antiguo. Estimar el tamaño total del sistema nuevo sumando los tamaños estimados de cada una de las partes.
  - *Modelos Paramétricos*: basados en un modelo matemático, utilizan un enfoque algorítmico (**Puntos Función** PF, ...)

Estimación del Esfuerzo

- El esfuerzo es un **indicador** *de la cantidad de trabajo necesario para realizar un proyecto o alguno de los ítems de un proyecto.*
- En productos software podemos considerar **equivalente** estimar el **esfuerzo y** el **coste**, ya que existe una relación directa entre ambos.
- En ingeniería del software se suele medir en unidades del tipo <persona><tiempo>:
  - personas-días, horas de analista, ...
- A partir de la estimación del tamaño, se puede derivar la estimación del esfuerzo utilizando también técnicas de los tipos citados:
  - Estimación por *Analogía*:
    - utilizando datos anteriores de la organización para saber cuanto esfuerzo se realizó en proyectos anteriores de tamaño similar al estimado, o
    - utilizando tablas de estimación para convertir desde líneas de código a esfuerzo (T21).
  - *Modelos Paramétricos*: empleando un método algorítmico para convertir la estimación del tamaño (en LDC o en PF) a una estimación del esfuerzo (Método **COCOMO**).

Tablas de estimación de esfuerzos y duraciones

Tamaño del Programa	Software de Sistemas		Software de Gestión		Software "a medida"	
(LDC)	Duración (meses)	Esfuerzo (personas-mes)	Duración (meses)	Esfuerzo (personas-mes)	Duración (meses)	Esfuerzo (personas-mes)
10.000	10	48	6	9	7	15
15.000	12	76	7	15	8	24
20.000	14	110	8	21	9	34
25.000	15	140	9	27	10	44
30.000	16	185	9	37	11	59
35.000	17	220	10	44	12	71
40.000	18	270	10	54	13	88
45.000	19	310	11	61	13	100
50.000	20	360	11	71	14	115
60.000	21	440	12	88	15	145
70.000	23	540	13	105	16	175
80.000	24	630	14	125	17	210
90.000	25	730	15	140	17	240
100.000	26	820	15	160	18	270
120.000	28	1.000	16	200	20	335
140.000	30	1.200	17	240	21	400
160.000	32	1.400	18	280	22	470
180.000	34	1.600	19	330	23	540
200.000	35	1.900	20	370	24	610
250.000	38	2.400	22	480	26	800
300.000	41	3.000	24	600	29	1.000
400.000	47	4.200	27	840	32	1.400
500.000	51	5.500	29	1.100	35	1.800

Tabla de Estimación de esfuerzo y duración de proyectos software de complejidad media

Estimación de la planificación

- El resultado es una estimación de la **duración** en *unidades temporales*: días, semanas, meses, ...
- Los métodos más habituales para calcular la duración de un proyecto software a partir de la estimación del esfuerzo son:
  - utilización de datos anteriores de la organización, o
  - utilización de tablas de estimación para convertir desde líneas de código a esfuerzo y duración, o
  - utilización de funciones de equivalencia semiempíricas del tipo *duración = función del esfuerzo*, que incluyen diversos parámetros cuyos valores se determinan empíricamente.
    - Por ejemplo,

      Duración en meses = 3'0 x personas-mes ^(1/3)
  - utilización de software de gestión de proyectos que permita realizar una planificación de la duración optimizando la utilización de los recursos disponibles.

Tipos de técnicas para estimación del software

- En la bibliografía se identifican los siguientes grupos de técnicas para estimar un producto software, ya sea su tamaño como su coste (o lo que es lo mismo, su esfuerzo):
  - Estimación **algorítmica**: se construye un *modelo paramétrico* basado en información histórica sobre los costes y, habitualmente, sobre el tamaño. Los modelos empleados pueden ser de dos clases:
    - *Empíricos*: se construyen únicamente a partir de los datos históricos, mediante regresión (ejemplo: COCOMO).
    - *Teóricos*: se derivan de hipótesis teóricas acerca del comportamiento de los proyectos (ejemplo: SLIM).
  - Estimación **heurística**: se incluyen aquí técnicas heurísticas como reglas de inducción, técnicas fuzzy (lógica difusa), redes neuronales, razonamiento basado en casos y, en los últimos años, los algoritmos de computación genética.
  - Estimación por **analogía**.
  - **Juicio de expertos**.

Técnicas para estimar el tamaño del software

- La clásica de los **Puntos Función** (PF), que analizaremos a continuación en detalle.
- Modelos algorítmicos basados en el número de variables y subprogramas para estimar el tamaño en la fase de diseño.
- Estimar el **NLDC** (número de líneas de código) a partir del número de elementos de datos que pertenecen a un determinado proceso elemental (que se obtienen a su vez a partir de los DFD's).
- Métodos basados en estimar el NLDC de cada **componente** según su tipo. Aquí se incluye alguna propuesta que generaliza los PF's.
- Para software orientado a objetos también se han propuesto métodos basados en **Puntos Objeto** (COCOMO II).

Estimación del tamaño mediante Puntos Función

- Es la **técnica algorítmica** de estimación del tamaño de un producto software más conocida. Propuesta por Albercht en 1979 y mejorada en 1983.
- Utiliza un modelo paramétrico (lista de parámetros) **orientado hacia las aplicaciones de gestión**.
- Un punto función (PF) es una medida sintética del tamaño de un programa. La estimación del número de PF de un producto software pretende medir su funcionalidad y no el NLDC.
- Los pasos a seguir son:

  1) Calcular los Puntos Función sin ajustar:

     1.1) Contar el número de **funciones de usuario** (basado en contar el número de elementos de 5 tipos diferentes).

     1.2) Determinar el **nivel de complejidad** (baja, media, alta) de cada función de usuario. Para ello se tienen en cuenta el número de tipos de elementos de datos y el número de tipos de archivos (o de elementos de tipo registro) referenciados.

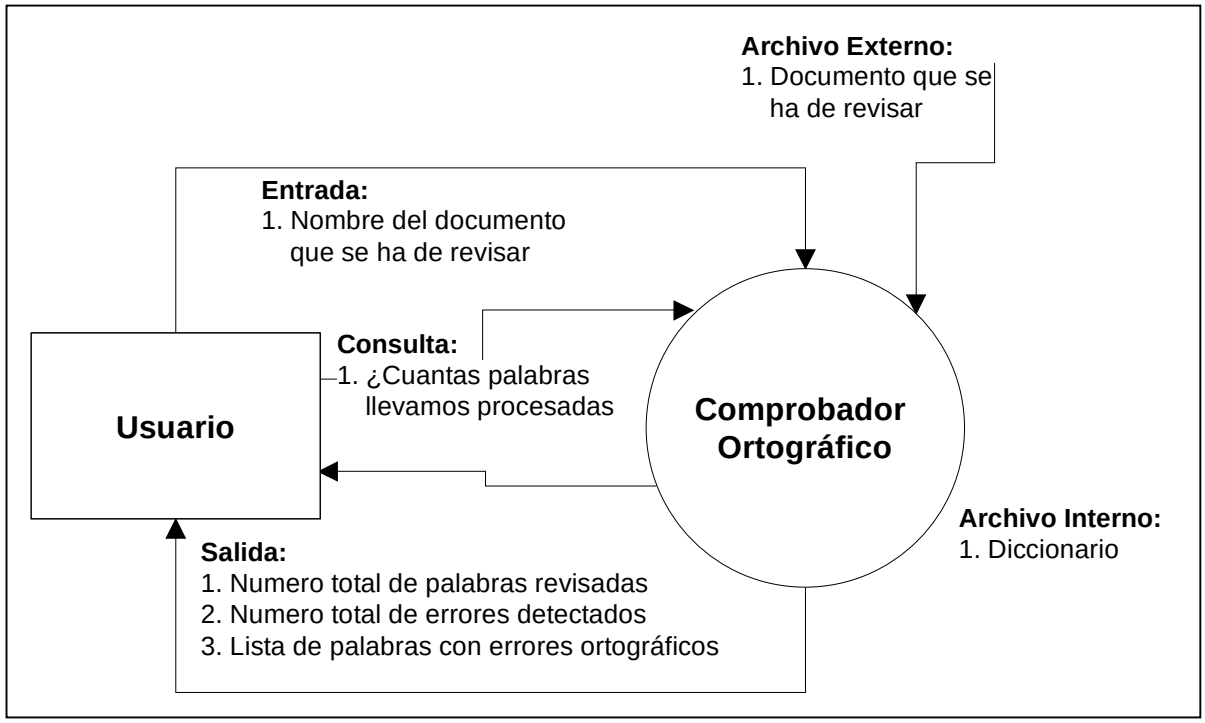
     1.3) Aplicar **pesos de complejidad**. Aplicar pesos según el nivel complejidad. La suma para todas las funciones de usuario equivale al nº de PF sin ajustar.

  2) Ajustar lo anterior para tener en cuenta la "complejidad del proceso". Para ello, se valora el grado de **influencia** de 14 **factores** diferentes.

PF: tipos de funciones de usuario

- En la etapa primera, se definen cinco tipos de funciones de usuario:
  - **Entradas externas** (entradas): cualquier entrada (pantalla, formulario, cuadro de diálogo, control o mensaje) que tenga un formato único o un solo procesamiento, a través de la cual el usuario u otro programa puede añadir, borrar o cambiar datos.
  - **Salidas externas** (salidas): cualquier salida (pantalla, informe, gráfico, mensaje) que tenga un formato diferente o requiera un procesamiento diferente a otros tipos de salida, generada para el usuario u otro programa.
  - **Consultas externas** (consultas): combinaciones de entrada/salida en las que cada entrada genera una salid simple e inmediata.
  - **Archivos lógicos internos** (archivos): principales grupos lógicos de datos de usuarios o de control que están controlados completamente por el programa (una tabla de un SGBDR).
  - **Archivos de interfaz externos** (interfaces): cada uno de los grupos de datos lógicos o información de control que entra o sale del programa.

PF: Ejemplo de cálculo del número de funciones de usuario



- *entradas*: 1 (el nombre del archivo que ha de revisarse),
- *salidas*: 3 (el número total de palabras revisadas, el número total de errores y una lista de las palabras erróneamente escritas),
- *consultas*: 1 (el usuario puede obtener interactivamente el número de palabras procesadas hasta el momento),
- *archivos*: 1 (el diccionario), y
- *interfaces*: 1 (el documento a inspeccionar).

- El **número total de funciones de usuario** es 1+3+1+1+1=7.

PF: Puntos Función sin ajustar

Tipo de función de usuario	Nivel de complejidad	Nº	*	Peso	=	Total
Entradas	Baja			3		
	Media			4		
	Alta			6		
Salidas	Baja			4		
	Media			5		
	Alta			7		
Consultas	Baja			3		
	Media			4		
	Alta			6		
Archivos	Baja			7		
	Media			10		
	Alta			15		
Interfaces	Baja			5		
	Media			7		
	Alta			10		
Número de Puntos Función sin ajustar:						***SUMA***

PF: tablas de niveles de complejidad de las funciones de usuario (1)

- Para hacer menos subjetiva la complejidad, algunas técnicas incluyen **tablas orientativas** para cada tipo de funciones de usuario.

- La tabla de complejidad para las **Entradas** tiene en cuenta dos aspectos:
  - nº de tipos de elementos de datos incluidos, y
  - nº de archivos lógicos internos referenciados.

Nº tipos archivos referenciados	**Nº tipos de elementos de datos incluidos**		
	1-4	5-15	>=16
0-1	Baja	Baja	Media
2-3	Baja	Media	alta
>=4	Media	Alta	Alta
Complejidad de las Entradas			

PF: tablas de niveles de complejidad de las funciones de usuario (2)

- La tabla de complejidad para las **Salidas** y las **Consultas** es:

Nº tipos archivos referenciados	Nº tipos de elementos de datos incluidos		
	1-5	6-19	>=20
0-1	Baja	Baja	Media
2-3	Baja	Media	Alta
>=4	Media	Alta	Alta
Complejidad de las Salidas y Consultas			

- En el caso de **Informes**, algunos autores definen la complejidad de manera diferente:
  - *Baja*: Informes de una o dos columnas. Pocas transformaciones entre datos.
  - *Media*: Informes con múltiples columnas que pueden incluir subtotales. Múltiples transformaciones de datos.
  - *Alta*: Intrincadas transformaciones de datos. Se hacen referencias a archivos múltiples y complejos.
- Algunos autores distinguen entre Consultas de Entrada (con definiciones de los niveles de complejidad similares a las Entradas) y Consultas de Salida (con niveles similares a las Salidas).

PF: tablas de niveles de complejidad de las funciones de usuario (3)

- Los niveles de complejidad para **Archivos Lógicos Internos** y para **Archivos de Interfaz Externos** están determinados por:
  - el nº de tipos de registros (tipos de entidades, tuplas, ...) referenciados, y
  - el nº de tipos de elementos de datos incluidos (campos, atributos, ..)

Nº tipos registros referenciados	**Nº tipos de elementos de datos incluidos**		
	1-19	20-50	>=51
0-1	Baja	Baja	Media
2-5	Baja	Media	Alta
>=6	Media	Alta	Alta
Complejidad de los Archivos y las Interfaces			

PF: cálculo de los PF totales

$$PF=PFSA*(0,65+0.01*SUMA(GI))$$

siendo

- PF los Puntos Función totales,
- PFSA los Puntos Función sin ajustar, y
- SUMA(GI) la suma de los grados de influencia de 14 factores que influyen en la complejidad del proceso.

- A cada **factor de influencia** se le asigna un peso entre 0 y 5 (aceptando decimales) en base al nivel de influencia que tiene sobre el software:
  - 0 => ninguna,
  - 1 => muy poca,
  - 2 => moderada,
  - 3 => media,
  - 4 => significativa, y
  - 5 => esencial (mucha).
- El resultado es que los PF oscilan entre el 65% y el 135% de los PFSA. Con esto se pretende ajustar la *evaluación subjetiva de la dificultad* del sistema.

PF: Factores de influencia en la dificultad del sistema (1)

1. **Comunicaciones de datos**: concerniente a la transmisión de datos o información de control, enviados o recibidos mediante algún sistema de comunicaciones.
2. **Procesamiento distribuido**: concerniente a si una aplicación es monolítica y se ejecuta en un único procesador, o si la aplicación consiste en código independiente ejecutándose en procesadores distintos y persiguiendo un fin común.
3. **Objetivos de rendimiento**: tendrán una puntuación de 0 si el rendimiento de la aplicación no es relevante, o por el contrario la puntuación será 5 si es un factor crítico.
4. **Configuración de uso intensivo**: indica si el sistema se va a implantar en un entorno operativo que será utilizado de manera intensa.
5. **Tasas de transacción rápidas**: tendrá una puntuación de 5 si el volumen de transacciones es suficientemente alto como para requerir un esfuerzo de desarrollo especial para conseguir la productividad deseada.
6. **Entrada de datos en línea**: tendrá una puntuación de 0 si son interactivas menos del 15 por ciento de las transacciones, y tendrá una puntuación de 5 si más del 50 por ciento de las transacciones son interactivas.
7. **Amigabilidad en el diseño**: determina si las entradas de datos interactivas requieren que las transacciones de entrada se lleven a cabo sobre múltiples pantallas o variadas operaciones.

PF: Factores de influencia en la dificultad del sistema (2)

8. **Actualización de datos en línea**: tendrá puntuación máxima si las actualizaciones en línea son obligatorias y especialmente dificultosas, quizá debido a la necesidad de realizar copias de seguridad, o de proteger los datos contra cambios accidentales.
9. **Procesamiento complejo**: se puntuará con 5 si se requieren gran cantidad de decisiones lógicas, complicados procedimientos matemáticos o difícil manejo de excepciones.
10. **Reusabilidad**: indica si gran parte de la funcionalidad del proyecto, está pensada para un uso intensivo por otras aplicaciones.
11. **Facilidad de instalación**: un valor de 5 denota que la instalación del sistema es tan importante que requiere un esfuerzo especial para desarrollar el software necesario para realizarla.
12. **Facilidad operacional**: un valor de 5 indica que el sistema realiza pocas operaciones
13. **Adaptabilidad**: una puntuación máxima indicaría que el sistema se ha diseñado para soportar múltiples instalaciones en diferentes entornos y organizaciones.
14. **Versatilidad**: Determina si la aplicación se ha realizado para facilitar los cambios y para ser utilizada por el usuario.

PF: equivalencia con LDC

Lenguaje (o entorno de programación)	LDC/PF
4GL	40
Ada 83	71
Ada 95	49
APL	32
BASIC - compilado	91
BASIC - interpretado	128
BASIC ANSI/Quick/Turbo	64
C	128
C++	29
Clipper	19
Cobol ANSI 85	91
Delphi 1	29
Ensamblador	320
Ensamblador (Macro)	213
Forth	64
Fortran 77	105
FoxPro 2.5	34
Generador de Informes	80
Hoja de Cálculo	6
Java	53
Modula 2	80
Oracle	40
Oracle 2000	23
Paradox	36
Pascal	91
Pascal Turbo 5	49
Power Builder	16
Prolog	64
Visual Basic 3	32
Visual C++	34
Visual Cobol	20

COCOMO

COnstructive **CO**st **MO**del

(Modelo Constructivo de Costes)

- Desarrollado en 1981 por Barry Boehm (COCOMO 81).
- Es el modelo de estimación de costes del software **más utilizado**.
- En 1995 se publicó la versión COCOMO II, actualmente en vigor.
- Con ello los autores (Center for Software Engineering, University of Southern California) pretenden mejorar, ampliar y adaptar el modelo anterior a las nuevas formas en que se desarrolla el software:
  - Nuevas aproximaciones: desarrollo evolutivo, dirigido a riesgos, colaborativo.
  - Nuevos entornos: 4GL's, generadores de aplicaciones, orientación a objetos, ...
  - Nuevos paradigmas: reusabilidad, madurez, calidad total, ...

COCOMO 81

- Tres modos de desarrollo software diferentes:
  - **Orgánico**: para proyectos de no más de 50 KLDC, sobre áreas muy específicas y bien conocidas por el equipo participante.
  - **Semiempotrado** (semilibre): cuando el nivel de experiencia del equipo de desarrollo se sitúa en niveles intermedios y suelen ser sistemas con interfaces con otros sistemas, siendo su tamaño menor a 300 KLDC.
  - **Empotrado** (restringido): para proyectos de gran envergadura, con una exigencia de altos niveles de fiabilidad y en los que participan muchas personas.
- Presenta una **jerarquía de modelos** de estimación según el nivel de detalle empleado en su utilización:
  - **Básico**: se calcula el esfuerzo de desarrollo como función del tamaño estimado del software en LDC. Es adecuado para realizar estimaciones de forma rápida, aunque sin gran precisión.
  - **Intermedio**: el esfuerzo se calcula como función del tamaño del producto, modificado por la valoración de ciertos atributos que afectan a los costes, incluyendo una valoración subjetiva del producto, del hardware, del personal, etc.
  - **Detallado**: la valoración de los atributos tiene en cuenta su influencia en cada una de las fases de desarrollo del proyecto.

COCOMO 81: Modelo Básico

	Orgánico	Semiempotrado	Empotrado
Esfuerzo de desarrollo	$E_D=2,4(KLDC)^{1,05}$ pm	$E_D=3,0(KLDC)^{1,12}$ pm	$E_D=3,6(KLDC)^{1,20}$ pm
Tiempo de desarrollo	$T_D=2,5(E_D)^{0,38}$ m	$T_D=2,5(E_D)^{0,35}$ m	$T_D=2,5(E_D)^{0,32}$ m
Productividad	$PR = LDC / E_D$		
Nº medio de personas a tiempo completo	FSP (Full-Time equivalent Software Personel) $P_E = E_D / T_D$ p		
Esfuerzo de Mantenimiento	TCA (Tráfico de cambio anual): porción de instrucciones fuente que sufren algún cambio durante un año, bien sea por adición o por modificación. $E_M = TCA \times E_D$ Y por tanto el valor medio del número de personas a tiempo completo, dedicadas a mantenimiento durante 12 meses sería: $P_{EM} = E_M / 12$		

pm=personas-mes, m=meses, p=personas

COCOMO 81: Ejemplo

- Desarrollar un producto software de un tamaño estimado de 32 KLDC.

- Por el tamaño del producto a desarrollar vemos que debemos aplicar las ecuaciones asociadas al modo orgánico, obteniendo lo siguiente:

  $E_D = 2{,}4\ (32)^{1{,}05} = 91$ pm

  $T_D = 2{,}5\ (91)^{0{,}38} = 14$ m

  PR = 32000 / 91 = 352 LDC/pm

  $P_E = 91 / 14 = 6{,}5$ p

- y si al año se cambian, eliminan o adicionan por mantenimiento 2000 LDC:

  TCA = 2000 / 32000 = 0,0625 = 6’25%

  $E_M = 0{,}0625 \times 91 = 5{,}7$ pm

  $P_{EM} = 5{,}7 / 12 = 0{,}5$ p

COCOMO II: Objetivos

1) Desarrollar un modelo de estimación de costes y tiempos en consonancia con las prácticas actuales de ciclo de vida del software.
2) Construir una base de datos y una herramienta de costes del software que incluya capacidades para la mejora continua del modelo.
3) Proveer un marco analítico cuantitativo, y un conjunto de herramientas y técnicas para evaluar los efectos de las mejoras en la tecnología software sobre los costes y tiempos del ciclo de vida del software.

- Estos objetivos intentan satisfacer las necesidades de los ingenieros del software:

  soportar planificación de proyectos, previsión de personal, estimaciones a la conclusión, preparación de proyectos, replanificación, rastreo de proyectos, negociación de contratos, evaluación de propuestas, nivelación de recursos, evaluación de diseños, etc.

COCOMO II: Sectores de Mercado (1)

<table>
<tr><td colspan="3">Programación de Usuario Final</td></tr>
<tr><td>Generadores de Aplicaciones y Ayudas para Composición</td><td>Composición de Aplicaciones</td><td>Integración de Sistemas</td></tr>
<tr><td colspan="3">Infraestructura (software de base y middleware)</td></tr>
</table>

- Los desarrolladores conocen muy bien la tecnología y poco las aplicaciones:
  - **Infraestructura**: abarca los sistemas operativos, SGBD's, sistemas de gestión de interfaces de usuario, sistemas de redes, middleware para procesamiento distribuido o procesamiento de transacciones, etc.
- El desarrollador (el usuario final) conoce bien el dominio de aplicación y mal la tecnología:
  - **Programación de Usuario Final**: comprende las soluciones de procesamiento de información rápidas y flexibles, realizadas por el propio usuario (con hojas de cálculo, generadores de aplicaciones, generadores de consultas e informes, etc.).

COCOMO II: Sectores de Mercado (2)

<table>
<tr><td colspan="3">Programación de Usuario Final</td></tr>
<tr><td>Generadores de Aplicaciones y Ayudas para Composición</td><td>Composición de Aplicaciones</td><td>Integración de Sistemas</td></tr>
<tr><td colspan="3">Infraestructura (software de base y middleware)</td></tr>
</table>

- Los desarrolladores deben conocer bien la tecnología (del sector Infraestructura) y también uno o más dominios de aplicación:
  - **Generadores de Aplicaciones** y **Ayudas para la Composición de Aplicaciones**: crear capacidades pre-empaquetadas para la programación de usuario final,:
  - **Composición de Aplicaciones**: sector dedicado a las aplicaciones demasiado diversificadas para ser manejadas con soluciones genéricas, pero suficientemente simples para ser construidas integrando componentes horizontales (SBGD, GUI, middleware) y/o verticales (específicos de un dominio).
  - **Integración de Sistemas**: se trabaja a gran escala, con sistemas muy embebidos o sin antecedentes. Suelen requerir una cantidad importante de programación específica.

COCOMO II: Modelos (1)

- *Programación de Usuario Final*: no necesita un modelo COCOMO.

- *Composición de Aplicaciones*:
  - Modelo ACM (**Application Composition Model**).
  - Basado en **Puntos Objeto** (PO).
  - Esta adaptado a la información normalmente conocida al planificar un producto de este sector y al nivel de exactitud requerido.
  - Estas aplicaciones suelen ser desarrolladas por un equipo reducido de personas durante varias semanas o meses.

- *Generadores de Aplicaciones, Integración de Sistemas,* o *Infraestructura*:
  - Las estimaciones combinan, dependiendo de la etapa del ciclo de vida, ACM con dos modelos de estimación incremental detallada:
  - Modelo EDM (**Early Design Model**), y
  - Modelo PAM (**Post-Architecture Model**).

COCOMO II: Modelos (2)

- Modelo **EDM**:
  - Usado en las etapas iniciales cuando se conoce poco sobre el tamaño del producto, la plataforma, el personal o el proceso.
  - Utiliza 7 conductores de coste (*cost drivers*) que afectan multiplicativamente al coste del proyecto (por ejemplo, capacidad del personal).
  - Trabaja con un nivel de detalle consistente con la información disponible y el nivel general de exactitud necesarios en la etapa de ***diseño inicial***.
- Modelo **PAM**:
  - Orientado a las etapas de ***desarrollo y mantenimiento*** de un producto software.
  - Se debe conocer la arquitectura del ciclo de vida para:
    - Proveer información más exacta sobre los generadores de costes, y
    - Permitir una estimación de costes más exacta.
  - Utiliza instrucciones de código fuente (similar a las LDC) y/o PF.
  - Incluye modificadores del tamaño para valorar la reusabilidad y otros aspectos.
  - Incorpora 17 conductores de coste (en vez de los 7 de EDM), y 5 factores que afectan exponencialmente al coste del proyecto.

COCOMO II: Métricas de tamaño (1)

- **Puntos Objeto** (PO):

  contadores de pantallas, informes y módulos 3GL, cada uno afectado de un peso según un factor de complejidad.

- **Puntos Función No Ajustados** (PFNA):

  PF sin tener en cuenta los 14 factores de influencia en la dificultad del sistema porque ya se consideran mediante los conductores de coste de COCOMO. [T33]

- **Líneas de Código Fuente** (LDCF):

  se contabilizan según la propuesta del Software Engineering Institute (SEI).

COCOMO II: Métricas de tamaño (2)

Definition Checklist for Source Statements Counts

Definition name: Logical Source Statements Date: _____

(basic definition) Originator: COCOMO II

Measurement unit		Physical source lines				
		Logical source statements	☑			
Statement type	Definition ☑	Data Array			Includes	Excludes
When a line or statement contains more than one type, classify it as the type with the highest precedence.						
1 Executable		Order of precedence →		1	☑	
2 Nonexecutable						
3 Declarations				2	☑	
4 Compiler directives				3	☑	
5 Comments						
6 On their own lines				4		☑
7 On lines with source code				5		☑
8 Banners and non-blank spacers				6		☑
9 Blank (empty) comments				7		☑
10 Blank lines				8		☑
11						
12						

How produced	Definition ☑	Data array		Includes	Excludes
1 Programmed				☑	
2 Generated with source code generators					☑
3 Converted with automated translators				☑	
4 Copied or reused without change				☑	
5 Modified				☑	
6 Removed					☑
7					
8					

Origin	Definition ☑	Data array		Includes	Excludes
1 New work: no prior existence				☑	
2 Prior work: taken or adapted from					
3 A previous version, build, or release				☑	
4 Commercial, off-the-shelf software (COTS), other than libraries					☑
5 Government furnished software (GFS), other than reuse libraries					☑
6 Another product					☑
7 A vendor-supplied language support library (unmodified)					☑
8 A vendor-supplied operating system or utility (unmodified)					☑
9 A local or modified language support library or operating system					☑
10 Other commercial library					☑
11 A reuse library (software designed for reuse)				☑	
12 Other software component or library				☑	
13					
14					

Estimación del esfuerzo de desarrollo

$$PM_{nominal} = A*(Size)^{B}$$

- Ecuación básica de los modelos EDM y PAM para calcular el esfuerzo en personas-mes (PM) necesario para desarrollar un software.
  - Size = tamaño en KLCDF (miles de LDCF) de la aplicación, igual a la suma total de los tamaños estimados de todos los módulos.
  - Si el tamaño se estima en PFNA, éstos se deben convertir a LDCF con las tablas ya vistas.
  - A = constante de calibración (su valor actual es 2'45).
  - B = factor de escala para tener en cuenta las diversas economías de escala, positivas o negativas, existentes en proyectos software.
    - En el modelo ACM su valor es 1'0 (ajuste lineal entre PM y Size).
    - En los modelos EDM y PAM su valor depende de 5 factores de escala, asignando a cada uno un peso de 0 (muy alto) a 5 (muy bajo).

$$B = 0.91 + 0.01 * \sum_{i=1}^{5} W_i$$

Ajuste del Tamaño (1)

- COCOMO II incorpora ajustes por 4 causas:
  - Breakage (desecho),
  - Reutilización,
  - Reingeniería o conversión, y
  - Mantenimiento.

- **Breakage** (BRAK): indicador del % de código desechado respecto del total desarrollado debido a la volatilidad de los requerimientos.
  - En el modelo ACM es 0%.

$$Size_{BREAK} = \left(1 + \frac{BRAK}{100}\right) * Size$$

Ajuste del Tamaño (2)

- **Efectos de la reutilización**: COCOMO trata esta reutilización del software usando un modelo de estimación no lineal para calcular las LDCF equivalentes a nuevo desarrollo (ESLOC):
  - si AAF ≤ 0.5

$$ESLOC = ASLOC * \frac{(AA + AAF * (1 + 0.02 * SU * UNFM))}{100}$$

  - si AAF > 0.5

$$ESLOC = ASLOC * \frac{(AA + AAF + SU * UNFM)}{100}$$

Con:

ASLOC = nº LDCF adaptadas de software existente,

AA (assesment and assimilation) = grado de valoración y asimilación necesarios para decidir cuando un módulo software reutilizado por completo es apropiado para la aplicación,

SU (software understanding) = % de esfuerzo de reutilización debido a la comprensión del software,

UNFM (programmer unfamiliarity) = indicador de la familiaridad del programador con el software, y

AAF (adaptation adjustment factor) = factor de ajuste de la adaptación:

$$AAF = 0.4 * DM + 0.3 * CM + 0.3 * IM$$

DM = % de modificación del diseño,

CM = % de modificación del código,

IM = % del esfuerzo de integración original requerido para integrar el software reutilizado.

Ajuste del Tamaño (3)

- Ajustes por **reingeniería o conversión**: El ajuste anterior por reutilización tiene un refinamiento adicional para contemplar los efectos de la reingeniería y/o conversión debidos a la eficiencia de las herramientas automáticas para reestructuración del software.

$$PM_{nominal} = A*(Size)^{B} + \left[\frac{ASLOC*(AT/100)}{ATPROD}\right]$$

siendo

  - AT: % de código que es sometido a reingeniería mediante traslación automática, y
  - ATPROD: productividad de las herramientas en LDCF/PM (actualmente se estima en 2400).

- **Mantenimiento de Aplicaciones**: cuando el % de código existente que cambia es mayor del 20%, COCOMO utiliza el "*Tamaño de Mantenimiento*" en vez de la reusabilidad.

$$SizeM = (Size_{añadido} + Size_{\mathrm{mod}ificado})*\left[1+\left(\frac{SU}{100}\right)*UNFM\right]$$

siendo

  - SizeM: el tamaño de mantenimiento (en LDCF o PF),
  - Size añadido: las LDCF/PF a añadir,
  - Size modificado: las LDCF/PF a modificar,

Multiplicadores de Esfuerzo (1)

- Son conductores de costes, utilizados en los modelos EDM y PAM para ajustar el esfuerzo nominal de manera multiplicativa:

$$PM = PM_{nominal} * \prod_{i=1}^{N} EM_i$$

  - siendo EM los multiplicadores de esfuerzo.
  - A cada EM se le asigna un ratio entre 1 y 5-7 (según el multiplicador).

- En el modelo **EDM** son 7:
  - RCPX: Fiabilidad y complejidad del producto.
  - RUSE: Reutilización requerida.
  - PDIF: Dificultad de la plataforma.
  - PERS: Capacidad del personal.
  - PREX: Experiencia del personal.
  - FCIL: Medios (*facilities*).
  - SCED: Calendario.

- En el modelo **PAM** son 17, obtenidos al desglosar los 7 anteriores. Se agrupan en 4 categorías:
  - del Producto, de la Plataforma, del Personal, y del Proyecto.

Multiplicadores de Esfuerzo (2)

- Factores del **Producto**: equivalentes a RCPX -los tres primeros- y RUSE -el último- en el modelo EDM.
  - RELY: Fiabilidad del producto requerida.
  - DATA: Tamaño de la base de datos.
  - CPLX: Complejidad del producto.
  - DOCU: Adecuación de la documentación a las necesidades del ciclo de vida.
  - RUSE: Reutilización requerida.
- Factores de la **Plataforma**:equivalentes a PDIF en el modelo EDM.
  - TIME: Limitaciones en el tiempo de ejecución.
  - STOR: Limitaciones en el almacenamiento principal.
  - PVOL: Volatilidad de la plataforma.
- Factores del **Personal**:equivalentes a PERS -los tres primeros- y PREX -los tres últimos- en el modelo EDM.
  - ACAP: Capacidad de los analistas.
  - PCAP: Capacidad del programador.
  - PCON: Continuidad del personal.
  - AEXP: Experiencia en aplicaciones.
  - PEXP: Experiencia en la plataforma.
  - LTEX: Experiencia con el lenguaje y las herramientas.
- Factores del **Proyecto**: equivalentes a FCIL -los dos primeros- y SCED -el último- en el modelo EDM.
  - TOOL: Uso de herramientas software.
  - SITE: Desarrollo en varios sitios.
  - SCED: Calendario de desarrollo requerido.

Factores de Escala

- Afectan al exponente B en la ecuación principal de estimación del esfuerzo:

$$PM_{nominal} = A*(Size)^{B}$$

$$B = 0.91 + 0.01*\sum_{i=1}^{5} W_i$$

- Con Wi:
  - PREC: **Ausencia de Precedentes** (Precedentedness).
  - FLEX: **Flexibilidad del desarrollo**.
  - RESL: **Resolución Arquitectura/Riesgos** (mide una combinación del uso de la gestión de riesgos y de la minuciosidad al diseñar la arquitectura del sistema).
  - TEAM: **Cohesión del equipo** de personas participantes.
  - PMAT: **Madurez del proceso** (basado en utilizar el modelo CMM - Capability Maturity Model- del Software Engineering Institute).

Estimación de la Duración

- La estimación del tiempo de desarrollo TDEV (en meses), conocido el esfuerzo estimado PM (en personas-mes), es:

$$TDEV = \left[3.67 * PM^{(0.28 + 0.2 * (B - 1.01))} \right] * \frac{SCED\%}{100}$$

- siendo
  - PM el esfuerzo de desarrollo excluyendo el multiplicador de esfuerzo de calendario SCED,
  - SCED% el porcentaje de reducción o incremento en el calendario nominal del proyecto (según se determinó al calcular SCED).

- Ejemplo:
- Si PM = 50 personas-mes, con factor de escala lineal (B=1.0) y SCED%=100

TDEV = 3.67 * 50 ^(0.30) * 1.00 = 11.9 meses