



Ingeniería del Software II

Tema 8

Gestión de Costes

Francisco Ruiz, Félix García



Objetivos

- Ampliar los conocimientos elementales de la gestión de costes ya estudiados en la introducción a la gestión de proyectos.
- Conocer los 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 el tamaño y el esfuerzo de un proyecto.



Índice

- 1. Introducción.
- 2. Planificación de recursos.
- 3. Estimación de costes.
- 4. Elaboración de presupuestos y control de gastos.
- 5. Introducción a la estimación del software.
 - 5.1. Refinamiento en proyectos software.
 - 5.2. Estimación del tamaño.
 - 5.3. Estimación del esfuerzo.
 - 5.4. Estimación de la duración.
 - 5.5. Tipos de técnicas para estimación del software.
- 6. Estimación del tamaño mediante Puntos Función.
- 7. Método COCOMO para estimación del software.
 - 7.1. Resumen de COCOMO 81.
 - 7.2. Características de COCOMO II.
 - 7.3. Estimación del esfuerzo.



Bibliografía básica:

- University of Southern California. COCOMO II Model Definition Manual. Version 1.4.
 - Disponible en <http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf>
- 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 2000.
 - Cap. 7.



Bibliografía complementaria:

- Parthasarathy, M. A. Practical Software Estimation. Function Point Methods for Insurced and Outsourced Projects. Addison-Wesley, 2007.
- Garmus, D., Herron, D. Function Point Analysis. Measurement Practices for Successful Software Projects. Addison-Wesley, 2003.



Introducción Mapa

PMBOK 2004			Contenidos del Módulo C=conceptos, T=técnicas y herramientas, S=salidas, N =normas y estándares
Área	Proceso	Grupo	
Gestión de Costes	Estimación de Costes	Planificación	T: Estimación por Analogía (top-down) T: Estimación Bottom-up T: Modelos paramétricos (COCOMO)
	Preparación del Presupuesto de Costes	Planificación	S: Línea Base de Costes
	Control de Costes	Seguimiento y Control	T: Estimación de costes a la conclusión



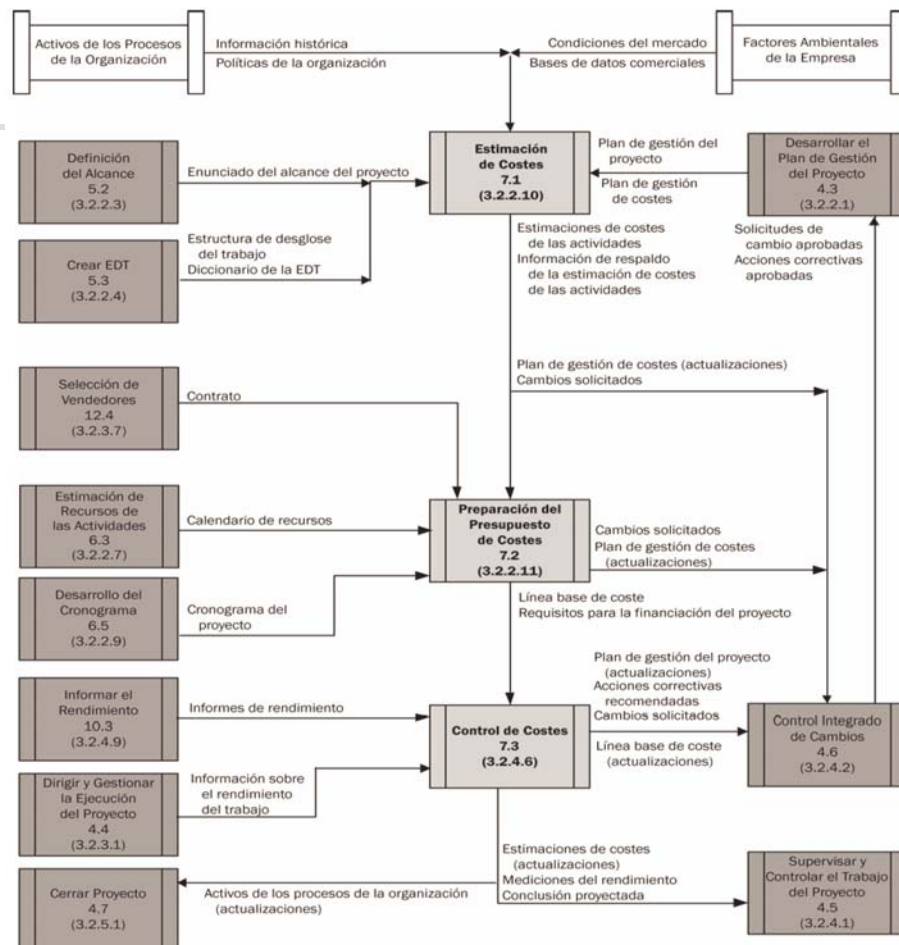
Introducción

La Gestión de Costes en PMBOK

- Objetivo: Completar el proyecto dentro del presupuesto aprobado.
- Procesos:
 - 7.1. Estimación de Costes:
 - Desarrollar una aproximación de los costes de los recursos necesarios para completar las actividades del proyecto.
 - 7.2 Preparación del Presupuesto de Costes:
 - Sumar los costes estimados de actividades individuales o paquetes de trabajo a fin de establecer una línea base de coste.
 - 7.3 Control de Costes:
 - Influir sobre los factores que crean variaciones del coste y controlar los cambios en el presupuesto del proyecto.
- En algunos proyectos pequeños, 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.



Introducción PMBOK





Estimación de costes Técnicas

- 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 Técnicas

- Determinación de Tarifas de Costes de los Recursos
 - Se debe conocer las tarifas de costes unitarios
 - Coste del personal por hora, Coste del material
 - Las bases de datos comerciales y las listas de precios publicadas de los vendedores son una fuente de tarifas de costes.
 - Si no se conocen las tarifas de costes reales, entonces las propias tarifas tendrán que estimarse.
- 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.



Estimación de costes

Técnicas:

- **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*: coste de una vivienda nueva = superficie en m² x 140000 pts/m²
 - *Complejo*: 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 *fiabes* 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).



Preparación del Presupuesto de Costes

Línea Base de Costes (*cost baseline*)

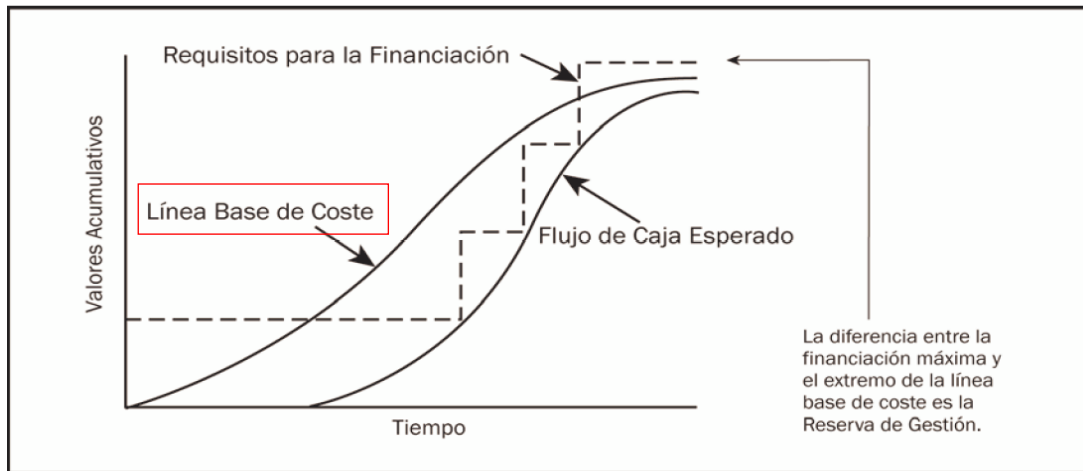
- Representa el reparto del presupuesto a lo largo del tiempo de duración del proyecto.
 - Sirve para medir y supervisar la evolución de los costes a lo largo de la realización del proyecto.
- Se calcula con:
 - los datos de estimación de costes de todos los paquetes de trabajo,
 - el WBS/DFT, y
 - el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



Preparación del Presupuesto de Costes

Base de Costes (*cost baseline*)

- Suele tener forma de S:



Control de Costes

Acciones

- **Influir sobre los factores** que producen **cambios** en la **línea base coste**
- **Asegurarse** de que los **cambios solicitados** sean **acordados**
- **Gestionar** los **cambios reales** cuando y a medida que se produzcan
- **Asegurar** que los posibles **sobrecostes no excedan** la **financiación autorizada** periódica y total para el proyecto
- Realizar el seguimiento del rendimiento del coste para detectar y entender las variaciones con respecto a la línea base de coste
- **Registrar todos los cambios pertinentes con precisión** en la línea base de coste
- **Evitar que se incluyan cambios incorrectos**, inadecuados o no aprobados en el coste o en el uso de recursos informados
- **Informar los cambios aprobados** a los interesados pertinentes
- **Actuar para mantener los sobrecostes esperados** dentro de límites aceptables.



Control de Costes

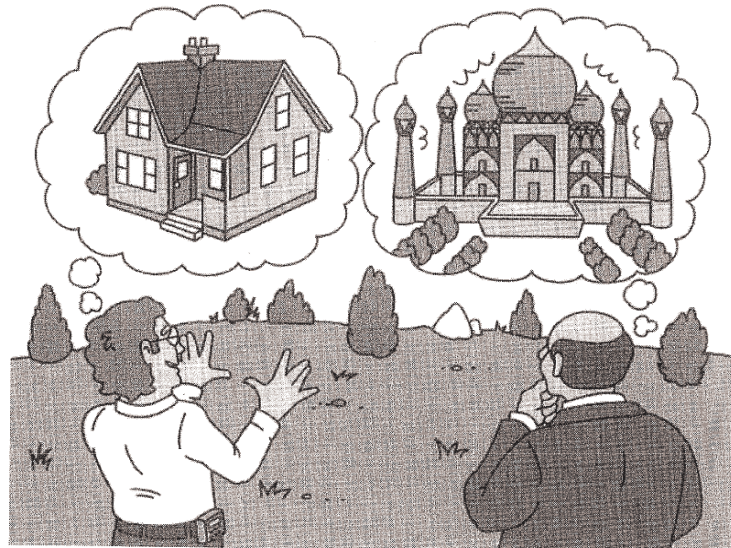
Estimación de Costes a la Conclusión

- 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 → *Earned Value Technique*



Estimación del Software

- Estimación vs Medición
- La naturaleza de la estimación Software



«¿Un año para construir una casa aquí? No hay ningún problema.»

«Bien. Vamos a empezar. Me corre prisa.»



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).
- **Etapa general:** *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**.



Estimación del Software Refinamientos

- La **estimación SW** está basada 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 requisitos, los requisitos 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.



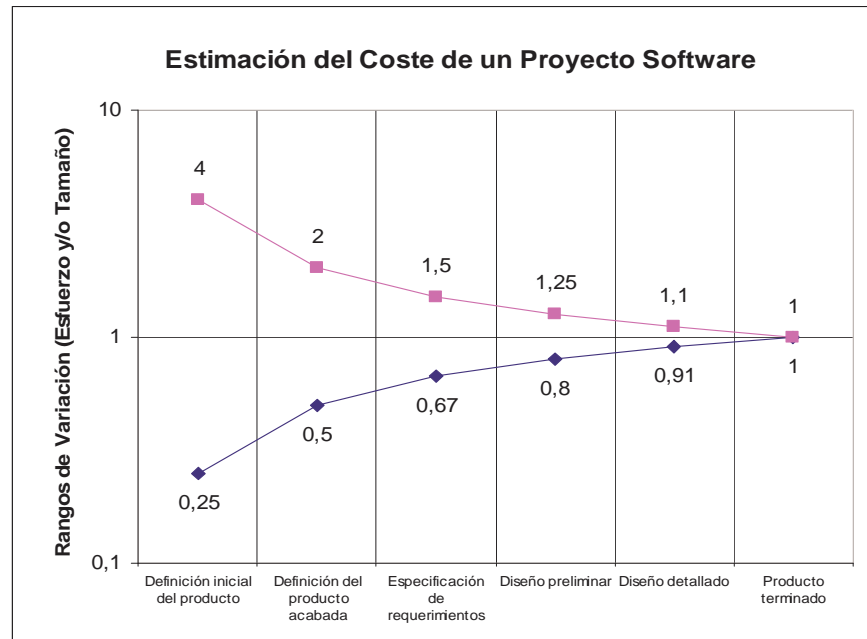
Estimación del Software

Refinamientos

- 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 requisitos** la **oscilación es de 1 a 2', ...**



Estimación del Software

Etapas: 1. 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.
 - 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 Software

Etapas: 2. 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, ...



Estimación del Software

Etapas: 2. Estimación del Esfuerzo

- 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 (Tr23).
 - **Modelos Paramétricos:**
 - empleando un método algorítmico para convertir la estimación del tamaño (en LDC o en PF) a una estimación del esfuerzo (Método **COCOMO**).



Estimación del Software

Tablas de Estimación de **esfuerzos y duraciones**

Tamaño del Programa (LDC)	Software de Sistemas		Software de Gestión		Software "a medida"	
	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

Ruiz & García. G. Costes

8.23



Estimación del Software

Etapas: 3. Estimación de la Duración

- 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,

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

Ruiz & García. G. Costes

8.24



Estimación del Software

Resumen de tipos de técnicas para estimar 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

- **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). **Problemas:**
 - la falta de una definición universal de línea de código,
 - su dependencia con el lenguaje de desarrollo y
 - la dificultad de estimar en fases tempranas del desarrollo la cantidad de líneas de código que tendrá una aplicación.
- Para solucionar estos problemas surge la medida clásica **Puntos Función (PF)**.
- **Modelos algorítmicos** basados en el número de variables y subprogramas para estimar el tamaño en la fase de diseño.
- Para software orientado a objetos también se han propuesto métodos basados en **Puntos Objeto** (COCOMO II).



Puntos Función

- Es la **técnica algorítmica** de estimación del tamaño de un producto software más conocida. Propuesta por Albrecht 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.
 - Su valor es proporcional a la funcionalidad proporcionada por el programa
 - Se centra en la **perspectiva del sistema por parte del usuario**



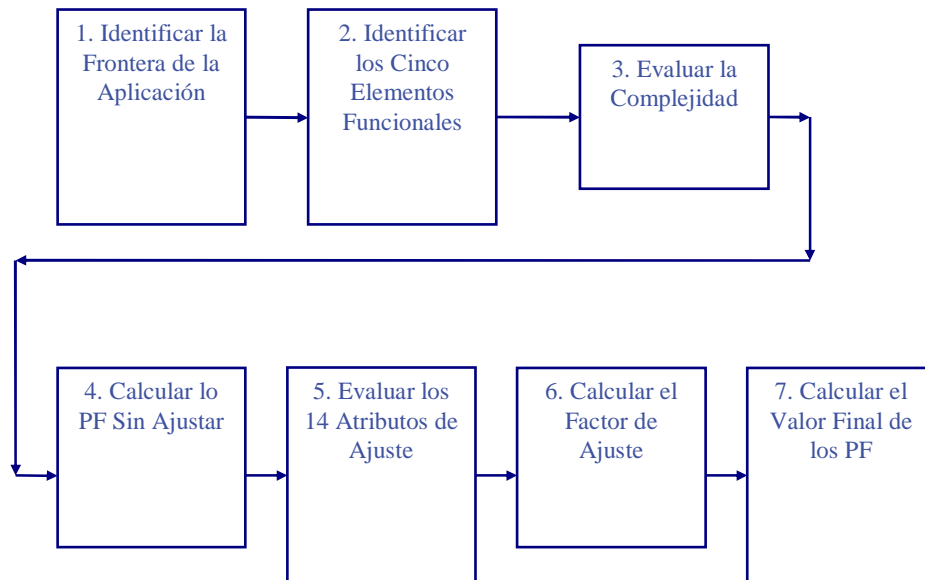
Puntos Función Proceso de Cálculo

- Cálculo de Puntos Función para **nuevos desarrollos**:
Pasos Generales
 - 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.



Puntos Función Proceso de Cálculo

- Cálculo de Puntos Función para nuevos desarrollos:
 - Proceso definido por el IFPUG:



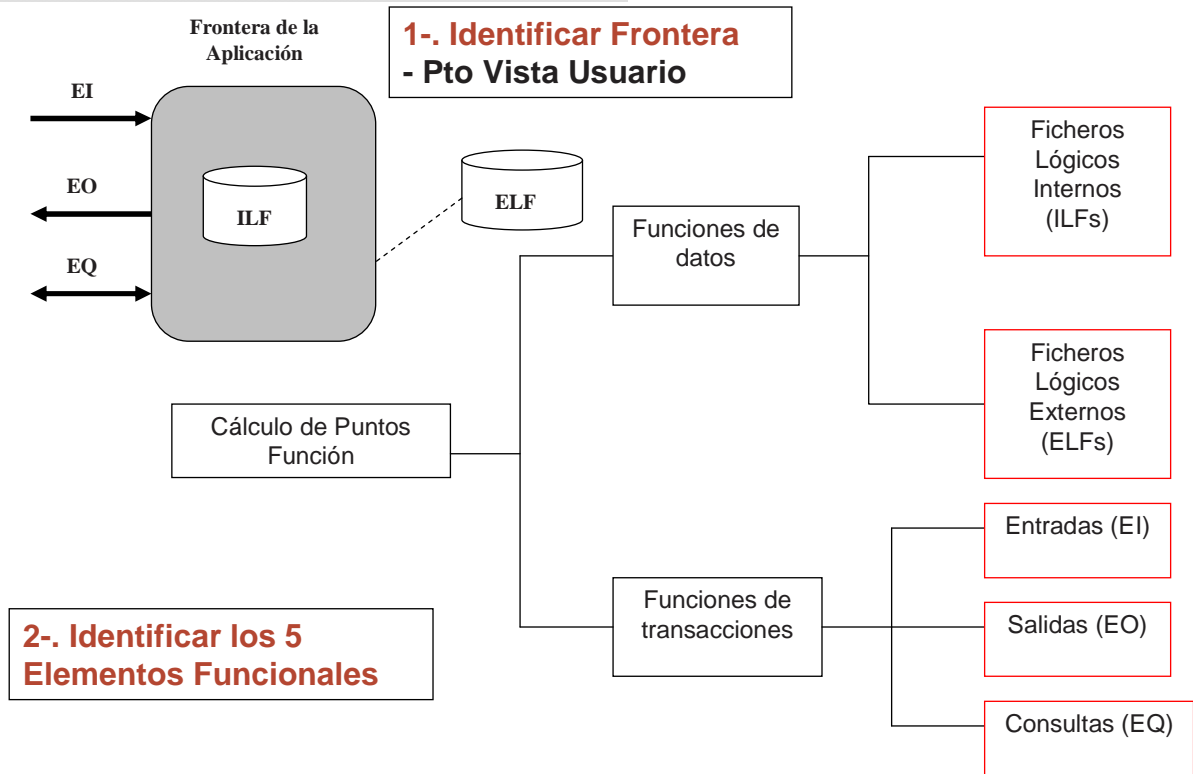
Puntos Función Proceso de Cálculo

- Requisitos para la estimación de los PF en etapas tempranas:
 - En estas fases no se dispone de un documento detallado de especificación de requisitos por lo que se parte de:
 - Descripción de requisitos expresada en el Pliego de Prescripciones Técnicas (PPT).
 - Descripción informal de requisitos expresada en el lenguaje del dominio de la aplicación por parte del cliente
 - En estos documentos se debe distinguir:
 - Los datos que utilizará la aplicación
 - **Datos** almacenados por la propia aplicación (Ficheros Lógicos internos, ILF) o datos que se deberán utilizar en la aplicación pero que son administrados y almacenados por otra aplicación (Ficheros Lógicos Externos, ELF).
 - **La funcionalidad** de la aplicación, distinguiendo los procesos elementales [entradas (EI), salidas (EO) y consultas (EQ)].
 - Para que un proceso sea considerado elemental, el proceso debe ser la menor unidad de actividad que tiene sentido para el usuario.
 - Cuanto más precisos sean los requisitos mejor se podrá realizar la estimación de los PF.



Puntos Función

Proceso de Cálculo



Ruiz & García. G. Costes

8.31



Puntos Función

Tipos de Funciones de Usuario

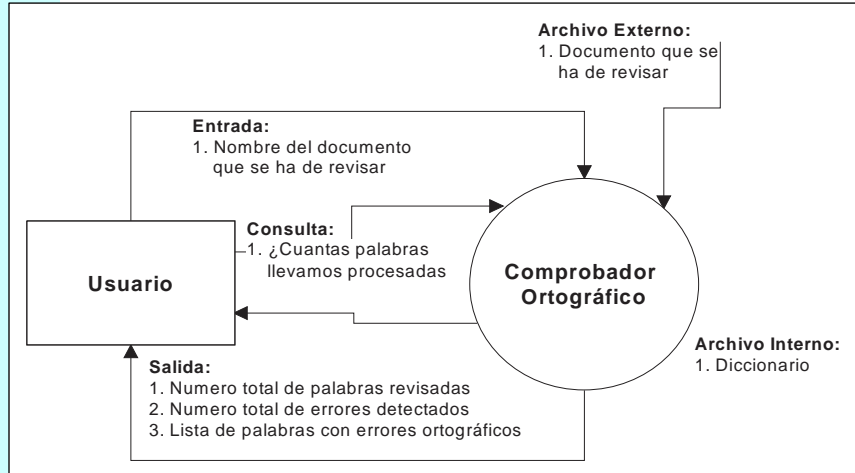
- **1ª Etapa.** Se definen cinco tipos de **Funciones de Usuario**:
 - **Entradas externas (entradas):**
 - cualquier entrada (pantalla, formulario, cuadro de diálogo, control o mensaje) que tenga un formato único o un solo procesamiento, a través de la cual el usuario u otro programa puede añadir, borrar 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 salida simple e inmediata.
 - **Archivos lógicos internos (archivos):**
 - principales grupos lógicos de datos de usuarios o de control que están controlados completamente por el programa (ej: tabla de un SGBDR).
 - **Archivos de interfaz externos (consultas):**
 - cada uno de los grupos de datos lógicos o información de control que entra o sale del programa.

Ruiz & García. G. Costes

8.32



Puntos Función Ejemplo básico



- **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 $1+3+1+1+1=7$.



Puntos Función Tipos de Funciones de Usuario

- **Archivos lógicos internos (ILF):**
 - **Grupo de datos** relacionados lógicamente o información de control reconocida por el **usuario** y mantenido dentro de los límites de la aplicación.
 - Se corresponde con las entidades que típicamente aparecen en un diagrama Entidad-Interrelación (ER).
 - Ejemplo: Supongamos que en una aplicación de alquiler de automóviles un usuario requiere la capacidad de "Introducir, consultar y listar información sobre los datos del cada automóvil (matrícula, marca, modelo, año de fabricación) y los datos de las personas que los alquilan (DNI, apellidos, nombres, dirección, localidad)".
 - Dos grupos de datos lógicos (ILFs), dos entidades:
 - 1. Datos de automóviles
 - 2. Datos de las personas



Puntos Función

Tipos de Funciones de Usuario

- **Archivos lógicos externos (ELF):**
 - **Grupo de datos** relacionados **lógicamente o información de control**, identificable por el **usuario**, referenciado por la aplicación, pero mantenido **dentro de los límites de otra aplicación**.
 - Ejemplo: Sistema de Nóminas con la posibilidad de “Realizar una interfaz de comunicación con la aplicación de Recursos Humanos para recuperar la información de datos de los empleados”.
 - Se distingue un ELF, una entidad: **Datos de empleados**
 - Es un ELF porque referencia a datos almacenados por otra aplicación, (Recursos Humanos).



Puntos Función

Tipos de Funciones de Usuario

Elemento Funcional	Sugerencias para Ayudar en la Estimación de los PF
ILF, ELF	¿Los datos forman un grupo lógico que satisface los requisitos específicos de usuario?
	Una aplicación puede utilizar un ILF o un ELF en muchos procesos pero el ILF o ELF se cuenta sólo una vez.
	Un archivo no puede ser contado a la vez como un ILF y un ELF en la misma aplicación.
	Un ELF será contado como un ILF en otra aplicación.
	No asuma que un archivo físico, tabla o clase de objeto, equivale a un archivo lógico, al considerar los datos desde el punto de vista del usuario. Si por ejemplo se identifica la entidad factura, la misma probablemente se registre en dos archivos o tablas: encabezado y líneas, sin embargo, estos dos archivos corresponden a una misma entidad lógica por lo que se cuenta
	¿Dónde se mantienen los datos? ¿Dentro o fuera de los límites de la aplicación?
	Si los datos son mantenidos por la aplicación que se está midiendo entonces lo contaremos como un ILF, en caso contrario un ELF.
	¿Se mantienen los datos de un ILF a través de un proceso elemental de la aplicación?
	Una aplicación puede utilizar un ILF o un ELF varias veces, pero se cuenta el ILF o ELF sólo una vez.
	Un proceso elemental puede mantener más de un ILF.
Considere los ILFs mantenidos por más de una aplicación en cada aplicación que los mantenga cuando ésta se mida.	



Puntos Función

Tipos de Funciones de Usuario

- **Entradas Externas (EI):** Proceso elemental que procesa datos o información de control que proviene de fuera de los límites de la aplicación
 - Para distinguirse de otras entradas se debe cumplir o que su lógica de proceso sea única, o que los datos o ficheros que se manejan son distintos
- Ejemplo:
 - Supongamos que en una aplicación de Recursos Humanos, el usuario requiere la posibilidad de dar de alta la información de trabajo online, generar un mensaje de error y resaltar los campos incorrectos para que el error pueda ser corregido online, guardar la información de trabajo dada de alta
 - Consideramos que estos requisitos pueden ser realizados por un **único proceso elemental**:
 - una entrada externa (EI), que se encargará de introducir los datos por pantalla, generar un mensaje de error y guardar los datos correspondientes en el ILF de trabajos.



Puntos Función

Tipos de Funciones de Usuario

- **Salidas Externas (EO):** Proceso elemental que envía datos o información de control hacia fuera de los límites de la aplicación.
 - Proceso lógico diferente o adicional al recuperación de datos o control (consultas).
- Ejemplo: REQ. "El usuario requiere notificación automática cuando un empleado ha completado 12 meses en su asignación de trabajo, momento en el que debería llevarse a cabo una revisión de su rendimiento".
 - Distinguimos un proceso elemental, un EO, que se encargará de calcular la fecha en la que se cumplen 12 meses, buscando información en el ILF que guarda información sobre la asignación de trabajos, y emitiendo un mensaje automático por pantalla.



Puntos Función

Tipos de Funciones de Usuario

- **Consultas Externas (EQ):** Proceso elemental que envía datos o información de control fuera de los límites de la aplicación.
 - Proporciona la salida de datos o información de control de un ILF o ELF.
 - Combinación E/S a partir de una búsqueda de datos:
 - no actualiza ficheros lógicos y no contiene datos derivados que requieran un proceso <> búsqueda, recuperación o catalogación)
- Ejemplo: REQ. "El usuario requiere que la aplicación imprima el Informe Mensual de Miembros automáticamente cada mes".
 - Se distingue un único proceso elemental, una consulta externa (EQ), cuyo objetivo es presentar la información de los miembros, sin requerir realizar ningún cálculo con los datos, sólo la recuperación de datos.



Puntos Función

Tipos de Funciones de Usuario

Elemento Funcional	Sugerencias para Ayudar en la Estimación de los PF
EI, EO, EQ	¿Los datos se reciben desde fuera de la aplicación?
	Si es así se habrá identificado un EI.
	¿El proceso mantiene algún ILF?
	Si es así se habrá identificado un EI o EO, y en caso contrario un EQ.
	¿Calcula el proceso algún dato derivado?
	Si es así se habrá identificado un EO.
	¿Contiene el proceso al menos un cálculo o fórmula matemática?
	Si es así se habrá identificado un EO.
¿Sólo recupera el proceso datos de uno o más ILFs o ELFs?	
Si es así se habrá identificado un EQ.	



Puntos Función

Complejidad de las Funciones de Usuario

- Para hacer menos subjetiva la **complejidad**, algunas técnicas incluyen **tablas orientativas** para cada tipo de funciones de usuario en las que se tienen en cuenta dos factores fundamentales.
 - **nº de tipos de elementos de datos incluidos** (DET, Data Element Type),
 - Ejemplo: Archivo con los campos:
 - nombre de empleado, dirección y edad → 3 DETs
 - **Nº de tipos de registros referenciados:**
 - **nº de tipos de archivos lógicos referenciados** (sólo para Entradas, Salidas y Consultas)
 - **nº de archivos lógicos internos referenciados** (RET, Record Element Type) (sólo para archivos)
 - Ejemplo: Archivo Empleado con los siguientes subgrupos:
 - Información Personal: DNI, Nombre, Dirección, Teléfono
 - Estudios: Titulación, año de obtención, Universidad
 - Experiencia Previa: Empresa, Rol, Periodo

3 RETS



Puntos Función

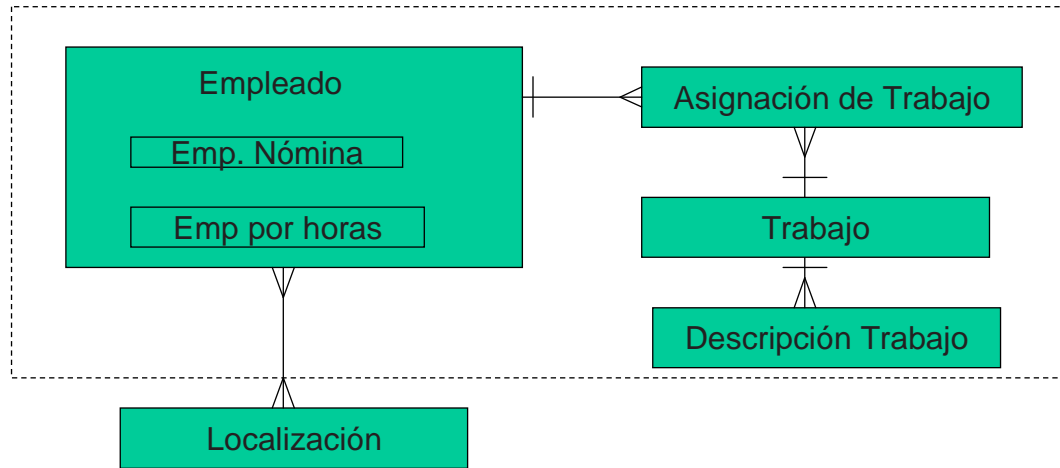
Complejidad de las Funciones de Usuario

- DET. Reglas:
 - 1. "Contar un DET para cada campo único, reconocido por el usuario, no repetido mantenido y/o recuperado de un ILF o ELF a través de un proceso elemental"
 - 2. "Cuando dos aplicaciones mantienen y/o referencian los mismos ILF/ELF pero cada una mantiene diferentes DETs, sólo se cuentan los correspondientes a esa aplicación".
 - Ejemplos:
 - 1. ILF Empleado: Nombre, DNI, Fecha de Nacimiento, Titulación, Universidad, Año, Tipo de Trabajo, Duración, Salario → 9 DETs
 - Si el usuario (no el diseñador) dice que el nombre debe dividirse en: apellidos y nombre, se cuenta como 2 DETs.
 - Aplicación A usa del ILF anterior el nombre, DNI, fecha nacimiento, titulación, universidad y año y la aplicación B el nombre, tipo de trabajo duración y salario.
 - En Aplicación A → 6 DETs
 - En Aplicación B → 4 DETs



Puntos Función Complejidad de las Funciones de Usuario

- RET.
 - 1. "Contar un RET para cada subgrupo en un ILF o ELF opcional u obligatorio".
 - 2. "Si no hay subgrupos contar como 1 RET"
- Ejemplo:



Puntos Función Complejidad de las Funciones de Usuario

Empleado
(ILF) → 8
DETs, 2
RETs

Trabajo
(ILF) →
4 DETs,
1 RETs

Asign.
Trab. (ILF)
→ 5 DETs,
1 RET

Campos	Proceso Conteo	Campos	Proceso Conteo
Empleado		Descripción Trab.	
Nombre		Identificador	
NSS		Número de Línea	
Num. Dependientes		Descripción Línea	
Tipo		Asignación Trabajo	
Nombre_Localización		Fecha_Inicio	
Empleado Nómina		Salario	
Nivel		Ratio Rendimiento	
Empleado Horas		Íd_Trabajo	
Horario estand		NSS_Empleado	
Colectivo Negociador	Localización		
Trabajo	Nombre	Dirección	DET2
Identificador	Salario General	Código_Interoficina	DET3
Nombre			
Salario General			

Localización → 3 DETs, 1 RET



Puntos Función Complejidad de las Funciones de Usuario

Nº tipos registros 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
<i>Complejidad de las Entradas</i>			

Nº tipos registros 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
<i>Complejidad de las Salidas y Consultas</i>			

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
<i>Complejidad de los Archivos y las Interfaces</i>			



Puntos Función Cálculo de los PF sin Ajustar

Tipo de función de usuario	Nivel de complejidad	Nº	*	Peso	=	Total
Entradas	Baja	2		3		6
	Media	4		4		16
	Alta			6		
Salidas	Baja			4		
	Media	1		5		5
	Alta			7		
Consultas	Baja	1		3		3
	Media			4		
	Alta			6		
Archivos	Baja	2		7		14
	Media	1		10		10
	Alta			15		
Interfaces	Baja			5		
	Media			7		
	Alta	1		10		10
Número de Puntos Función sin ajustar:						64



Puntos Función

Cálculo de los PF Totales

$$PF = PFSA * (0,65 + 0.01 * SUMA(GI)) \text{ siendo}$$

- 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 (much).
 - El resultado es que los **PF oscilan entre el 65% y el 135% de los PFSA** (se pretende ajustar la *evaluación subjetiva de la dificultad* del sistema).



Puntos Función

Factores de influencia en la dificultad del sistema

1. Comunicaciones de datos
2. Procesamiento distribuido
3. Objetivos de rendimiento
4. Configuración de uso intensivo
5. Tasas de transacción rápidas
6. Entrada de datos en línea
7. Amigabilidad en el diseño
8. Actualización de datos en línea
9. Procesamiento complejo
10. Reusabilidad
11. Facilidad de instalación
12. Facilidad operacional
13. Adaptabilidad
14. Versatilidad



Puntos Función Equivalencia con LOC

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 II Introducción

CONstructive COSt MOdel (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 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) Proporcionar 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

Programación de Usuario Final		
Generadores de Aplicaciones y Ayudas para Composición	Composición de Aplicaciones	Integración de Sistemas
Infraestructura (software de base y middleware)		

- 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

Programación de Usuario Final		
Generadores de Aplicaciones y Ayudas para Composición	Composición de Aplicaciones	Integración de Sistemas
Infraestructura (software de base y middleware)		

- 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

- *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

- Modelo **EDM**:
 - Usado en las etapas iniciales cuando se conoce poco sobre el tamaño del producto, la plataforma, el personal o el proceso.
 - Utiliza instrucciones de código fuente (similar a las LDC) y/o PF.
 - Utiliza **7** conductores de coste (cost drivers), o **multiplicadores de esfuerzo** (effort multipliers), que afectan multiplicativamente al esfuerzo.
 - Y **5 factores de escala** (scale factors), que afectan exponencialmente al esfuerzo del proyecto.
 - Trabaja con un nivel de detalle consistente con la información disponible y el nivel general de exactitud necesarios en la etapa de **diseño inicial**.



COCOMO II Modelos

- Modelo **PAM**:

Se diferencia de EDM en que:

 - Está orientado a las etapas de **desarrollo y mantenimiento** de un producto software.
 - Se debe conocer la arquitectura del ciclo de vida para:
 - Proporcionar información más exacta sobre los generadores de costes,
 - Permitir una estimación de costes más exacta.
 - Incluye modificadores del tamaño para valorar la reusabilidad y otros aspectos.
 - Incorpora **17 conductores de coste** (en vez de los 7 de EDM).



COCOMO II

Métricas de Tamaño

- **Puntos Objeto (PO):**
 - OP - Object Points
 - contadores de pantallas, informes y módulos 3GL, cada uno afectado de un peso según un factor de complejidad.
- **Puntos Función No Ajustados (PFNA):**
 - UFP - Unadjustment Function Point
 - 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.
- **Líneas de Código Fuente (LDCF):**
 - SLOC – Source Lines Of Code
 - se contabilizan según la propuesta del Software Engineering Institute (SEI).



COCOMO II

Métricas de Tamaño

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	
<i>When a line or statement contains more than one type, classify it as the type with the highest precedence.</i>					
1 Executable	Order of precedence →		1	<input checked="" type="checkbox"/>	
2 Nonexecutable			2	<input checked="" type="checkbox"/>	
3 Declarations			3	<input checked="" type="checkbox"/>	
4 Compiler directives					
5 Comments					
6 On their own lines			4		<input checked="" type="checkbox"/>
7 On lines with source code			5		<input checked="" type="checkbox"/>
8 Banners and non-blank spacers			6		<input checked="" type="checkbox"/>
9 Blank (empty) comments			7		<input checked="" type="checkbox"/>
10 Blank lines			8		<input checked="" type="checkbox"/>
11					
12					
How produced	Definition	Data array	Includes	Excludes	
1 Programmed			<input checked="" type="checkbox"/>		
2 Generated with source code generators				<input checked="" type="checkbox"/>	
3 Converted with automated translators			<input checked="" type="checkbox"/>		
4 Copied or reused without change			<input checked="" type="checkbox"/>		
5 Modified			<input checked="" type="checkbox"/>		
6 Removed				<input checked="" type="checkbox"/>	
7					
8					
Origin	Definition	Data array	Includes	Excludes	
1 New work: no prior existence			<input checked="" type="checkbox"/>		
2 Prior work: taken or adapted from					
3 A previous version, build, or release			<input checked="" type="checkbox"/>		
4 Commercial, off-the-shelf software (COTS), other than libraries				<input checked="" type="checkbox"/>	
5 Government furnished software (GFS), other than reuse libraries				<input checked="" type="checkbox"/>	
6 Another product				<input checked="" type="checkbox"/>	
7 A vendor-supplied language support library (unmodified)				<input checked="" type="checkbox"/>	
8 A vendor-supplied operating system or utility (unmodified)				<input checked="" type="checkbox"/>	
9 A local or modified language support library or operating system				<input checked="" type="checkbox"/>	
10 Other commercial library				<input checked="" type="checkbox"/>	
11 A reuse library (software designed for reuse)			<input checked="" type="checkbox"/>		
12 Other software component or library			<input checked="" type="checkbox"/>		
13					
14					

- Sólo se cuentan LOC que son entregadas como parte del producto (se excluyen "test drivers" y otro tipo de software de soporte)
- LOC sólo creadas por el personal del proyecto (se excluyen las obtenidas mediante generadores de aplicaciones)
- Una instrucción es una LOC
- Las declaraciones se cuentan como instrucciones
- Los comentarios no se cuentan como instrucciones



COCOMO II

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

- 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 (tablas).
 - A = constante de calibración (su valor actual es 2'94).
 - 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).

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



COCOMO II Ajuste del Tamaño

- 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%

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



COCOMO II Ajuste del Tamaño

- **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 $AAF \leq 0.5$
$$ESLOC = ASLOC * \frac{(AA + AAF * (1 + 0.02 * SU * UNFM))}{100}$$

- si $AAF > 0.5$
$$ESLOC = ASLOC * \frac{(AA + AAF + SU * UNFM)}{100}$$

Con:

- **ASLOC** = n° 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 SW
- **UNFM** (*programmer unfamiliarity*) = indicador de la familiaridad del programador con el SW
- **AAF** (*adaptation adjustment factor*) = factor de ajuste de la adaptación:

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

DM = % de modificación diseño,

CM = % de modificación código

IM = % del esfuerzo de integración original requerido para integrar SW reutilizado



COCOMO II Ajuste del Tamaño

- 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 traducción del software.

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

siendo

- AT: % de código que es sometido a reingeniería mediante traducción automática
- ATPROD: productividad de las herramientas en LDCF/PM (actualmente est. 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.

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

siendo

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



COCOMO II Multiplicadores de Esfuerzo

- 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.



COCOMO II Multiplicadores de Esfuerzo

- 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.



COCOMO II Multiplicadores de Esfuerzo

- Factores del **Producto**:

	Factor	Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Producto	RELY	Inconvenientes insignificantes, que afectan solamente a los desarrolladores	Mínimas pérdidas al usuario, fácilmente recuperables	Pérdidas moderadas al usuario recuperables sin grandes inconvenientes	Pérdida financiera elevada o inconveniente humano masivo	Vida humana en riesgo	
	DATA		DB bytes/Pgm SLOC <10	$10 \leq D/P < 100$	$100 \leq D/P < 1000$	$D/P > 1000$	
	CPLX	Ver Tabla 22					
	RUSE		Ningún componente reusable	Reusable dentro del mismo proyecto	Reusable dentro de un mismo programa	Reusable dentro de una misma línea de productos	Reusable dentro de múltiples líneas de producto
	DOCU	Muchas necesidades del ciclo de vida sin cubrir	Algunas necesidades del ciclo de vida sin cubrir	Necesidades del ciclo de vida cubiertas en su justa medida	Necesidades del ciclo de vida cubiertas ampliamente	Necesidades del ciclo de vida cubiertas excesivamente	

	Operaciones de Control	Operaciones computacionales	Operaciones dependientes de los dispositivos	Operaciones de administración de datos	Operaciones de administración de interfaces de usuario
Muy Bajo	Pocas estructuras sin anidamiento: DO, CASE, IF_THEN_ELSE. Composición modular simple por medio de llamadas a procedimientos o simples script	Evaluación de una expresión simple Por ejemplo: $A=B+C*(D-E)$	Sentencias de lectura / escritura con formatos simples	Arreglos simples en memoria principal. Consultas, actualizaciones a COTS-DB	Generadores de reportes, Formularios de entrada simples.

Ruiz & García. G. Coste

8.65



COCOMO II Multiplicadores de Esfuerzo

- Factores del **Personal**:

		Muy Bajo	Bajo	Normal	Alto	Muy Alto
Personal	ACAP	15 percentil	35 percentil	55 percentil	75 percentil	90 percentil
	PCAP	15 percentil	35 percentil	55 percentil	75 percentil	90 percentil
	PCON	48 % por año	24 % por año	12 % por año	6% por año	3 % por año
	AEXP	≤ 2 meses	≤ 6 meses	1 año	3 años	6 años
	PEXP	≤ 2 meses	≤ 6 meses	1 año	3 años	6 años
	LTEX	≤ 2 meses	≤ 6 meses	1 año	3 años	6 años

- Factores de la **Plataforma**:

		Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Plataforma	TIME			Uso de $\leq 50\%$ del tiempo de ejecución disponible	70%	85%	95%
	STOR			Uso de $\leq 50\%$ del porcentaje total de almacenamiento	70%	85%	95%
	PVOL		Un cambio principal cada 12 meses. Un cambio menor todos los meses	Cambio principal cada 6 meses. Cambio menor cada 2 semanas	Cambio principal cada 2 meses. Cambio menor uno por semana	Cambio principal cada 2 semanas. Cambio menor cada 2 días	

Ruiz & García. G. Costes

8.66



COCOMO II Multiplicadores de Esfuerzo

- Factores del **Proyecto**:

		Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Proyecto	TOOL	Herramientas que permiten editar, codificar, depurar	Herramientas simples con escasa integración al proceso de desarrollo	Herramientas básicas, integradas moderadamente	Herramientas robustas y maduras, integradas moderadamente	Herramientas altamente integradas a los procesos, métodos y reuso	
	SITE Ubicación Espacial	Internacional	Multi-ciudad y multi-compañía	Multi-ciudad o multi-compañía	Misma ciudad o área metropolitana	Mismo Edificio o complejo	Completamente Centralizado
	SITE Comunicación	Algún teléfono, mail	Teléfonos individuales, FAX	Email de banda angosta	Comunicaciones electrónicas de banda ancha	Comunicaciones electrónicas de banda ancha, ocasionalmente videoconferencia	Multimedia Interactiva
	SCED	75% del nominal	85% del nominal	100% del nominal	130% del nominal	160% del nominal	



COCOMO II 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 W_i :
 - 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).



COCOMO II Factores de Escala

- Escala General:

Factor de Escala Wj	Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Precedencia PREC	Completamente sin precedentes	Ampliamente sin precedentes	Algún precedente	Generalmente Familiar	Ampliamente Familiar	Completamente Familiar
Flexibilidad en el desarrollo FLEX	Rigurosa	Relajación Ocasional	Alguna Relajación	Conformidad en general	Alguna Conformidad	Metas generales
Arquitectura/ Resolución de riesgo RESL	Poca (20%)	Alguna (40%)	Siempre (60%)	Generalmente 75%	Principalmente (90%)	Completo (100%)
Cohesión de equipo TEAM	Interacciones difíciles	Interacciones con alguna dificultad	Interacciones básicamente cooperativas	Ampliamente Cooperativas	Altamente Cooperativas	Interacciones Sin Fisuras
Madurez del proceso PMAT	Desarrollado más adelante					



COCOMO II Estimación de la Duración

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

$$TDEV = \left[3.67 * PM^{(0.28 + 0.2 * (B - 0.91))} \right] * \frac{SCED \%}{100}$$

- 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



PF: Factores de influencia en la dificultad del sistema

- 1. Comunicaciones de datos:** concerniente a la transmisión de datos o información de control, enviados o recibidos mediante algún sistema de comunicaciones.
 - Cada aplicación al menos tiene dos capas: lado cliente, lado servidor
 - Dependiendo de la complejidad estas capas pueden extenderse a más capas o se pueden introducir protocolos de comunicación hardware y de red

GI	Líneas Guía
0	La aplicación es un proceso por lotes puro o un PC <i>stand alone</i> .
1	Proceso por lotes con impresión remota o entrada remota de datos.
2	Proceso por lotes con impresión remota y entrada remota de datos .
3	El TP es la interfaz para un proceso por lotes.
4	La aplicación está basada en un TP interactivo, pero con un solo protocolo de comunicaciones
5	La aplicación está basada en un TP interactivo, pero con más de un protocolo de



PF: Factores de influencia en la dificultad del sistema

- 2. Procesamiento de datos 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.
 - Tiene cierta relación con el factor de influencia 1

GI	Líneas Guía
0	La aplicación no tiene el objetivo de transferir datos o funciones de procesamiento entre componentes del sistema
1	Datos preparados por la aplicación para su procesamiento por el usuario final sobre otro componente del sistema (hoja de cálculo, base de datos)
2	La aplicación prepara los datos para transferirlos; luego se transfieren y se procesan sobre otro componente/máquina del sistema (no usuario final)
3	Proceso distribuido con transferencia de datos on line en una dirección.
4	Como el anterior, pero con transferencia de datos en ambas direcciones.
5	Las funciones de procesamiento se realizan dinámicamente sobre el componente del sistema más apropiado



PF: Factores de influencia en la dificultad del sistema

3. Rendimiento:

- Las aplicaciones que soportan negocios o misiones críticas tienen importantes restricciones de rendimiento: tiempos de respuesta, productividad en las transacciones, etc..

GI	Líneas Guía
0	No hay requisitos especiales
1	Se establecen requisitos de rendimiento y diseño, pero no se requieren acciones especiales
2	La respuesta del proceso o la productividad es crítica durante las horas punta. No se requiere un diseño especial para el uso de la CPU. El tiempo límite de procesamiento es el siguiente día de negocio.
3	La respuesta del proceso o la productividad es crítica durante todos los días del negocio. No se requiere un diseño especial para el uso de la CPU. Los requisitos de tiempo límite de procesamiento con sistemas de interfaz son críticos.
4	Los requisitos de rendimiento del usuario son suficientemente críticos como para incluir tareas de análisis de rendimiento en la fase de diseño.
5	Además, se emplearán herramientas específicas para el diseño que contemplen estas características.



PF: Factores de influencia en la dificultad del sistema

4. Configuración de Uso Intensivo: describe el grado en que las restricciones de recursos de la máquina influenciarán el desarrollo de la aplicación.

- Es frecuente que el arquitecto software tenga que enfrentarse a situaciones en las que el HW sobre el que residirá el SW tiene ciertas limitaciones (disponibilidad CPU, capacidad almacenamiento, ...)
- En estas situaciones se requieren diseños especiales y técnicas de programación adecuadas.

GI	Líneas Guía
0	No se incluyen restricciones operacionales ni explícitas ni implícitas
1	Existen restricciones operacionales pero son menos restrictivas que una aplicación típica y no requieren por tanto esfuerzo adicional
2	Se incluyen algunas consideraciones sobre tiempo y seguridad
3	Se incluyen requisitos específicos de procesador para una pieza concreta de la aplicación
4	Las limitaciones de operación restringen la ejecución de la aplicación en un procesador central o dedicado
5	Además, hay especiales restricciones sobre la aplicación en los componentes distribuidos del sistema



PF: Factores de influencia en la dificultad del sistema

5. **Tasas de transacción rápidas:** describe el grado en que el ritmo/ratio de transacciones de negocio influenciarán el desarrollo de la aplicación.
- Si es alto afectará al diseño, desarrollo, instalación y soporte

GI	Líneas Guía
0	Las transacciones no están afectadas por picos de tráfico.
1	Se preveen periodos de picos de tráfico mensuales, cuatrimestrales,..
2	Se preveen periodos de picos de tráfico semanales
3	Se preveen periodos de picos de tráfico diarios
4	Los picos de transacción indicados por el usuario en los requisitos o en acuerdos a nivel de servicio son suficientemente altos como para incluir tareas de análisis de rendimiento en la fase de diseño
5	Además, se requieren herramientas de análisis de rendimiento en las fases de diseño, desarrollo y / o instalación.



PF: Factores de influencia en la dificultad del sistema

6. **Entrada de Datos Online:** Describe el grado de entrada de datos de forma interactiva.

GI	Líneas Guía
0	Todas las transacciones son tratadas por lotes..
1	Entre el 1 y el 7% de las funciones son entradas interactivas de datos.
2	Entre el 8 y el 15% de las funciones son entradas interactivas de datos.
3	Entre el 16 y el 23% de las funciones son entradas interactivas de datos.
4	Entre el 24 y el 30% de las funciones son entradas interactivas de datos.
5	Más del 30% de las funciones son entradas interactivas de datos.



PF: Factores de influencia en la dificultad del sistema

7. Eficiencia de Usuario Final: mide la facilidad de uso de la aplicación

GI	Líneas Guía
0	No se han declarado ninguno de los requisitos especiales de usuario.
1	De 1 a 3 de los requisitos de la lista.
2	4 o 5 requisitos de la lista
3	Seis o más requisitos pero no hay requisitos especiales de usuario relacionados con la eficiencia
4	Seis o más requisitos y los requisitos de eficiencia son suficientemente importantes para incluir tareas de diseño para el usuario final (plantillas, minimizar "key strokes", ..)
5	Además se requiere del uso de herramientas y procesos para demostrar que se cumplen los requisitos establecidos

- Ayuda de navegación.
- Menús.
- Ayuda en línea.
- Movimiento automático del cursor.
- Scrolling.
- Impresión remota.
- Teclas de función preestablecidas.
- Procesos por lotes lanzados desde transacciones en línea.

- Selección de datos con el cursor.
- Gran uso de facilidades en el monitor (colores, textos resaltados, etc.).
- Copia impresa de las transacciones en línea.
- Ratón.
- Windows.
- Pantallas reducidas.
- Bilingüismo.
- Multilingüismo.

Ruiz & García. G. Costes

8.79



PF: Factores de influencia en la dificultad del sistema

8. Actualización de Datos Online: 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

GI	Líneas Guía
0	Ninguna.
1	Actualización en línea de uno a tres ficheros de control. El volumen de actualización es bajo y la recuperación fácil.
2	Actualización en línea de cuatro o más ficheros de control. El volumen de actualización es bajo y la recuperación fácil.
3	Actualización en línea de importantes ficheros internos
4	También, se considera esencial la protección contra pérdida de información y ha sido especialmente diseñada y programada en el sistema.
5	Además, grandes volúmenes implican consideraciones de coste en el proceso de recuperación. Se incluyen procedimientos de recuperación que requieren mínima intervención del operador.

Ruiz & García. G. Costes

8.80



PF: Factores de influencia en la dificultad del sistema

9. **Procesamiento Complejo**: se considera complejo cuando hay muchas interacciones, puntos de decisión o gran número de ecuaciones lógicas o matemáticas. Situaciones:

- Extensiones de proceso lógicas.
- Extensiones de proceso matemáticas.
- Muchos procesos de excepción que resultan en muchas funciones incompletas que deben repetirse (ejemplo: validación en un cajero).
- Procesos sensibles de control y / o seguridad.
- Procesos complejos para manejar múltiples posibilidades de Entrada / Salida (multimedia, independencia de dispositivos,...)..

GI	Líneas Guía
0	Ninguno de los anteriores es aplicable.
1	Es aplicable uno de los anteriores.
2	Son aplicables dos de los anteriores.
3	Son aplicables 3 de los anteriores.
4	Son aplicables 4 de los anteriores.
5	Todos ellos son aplicables.

Ruiz & García. G. Costes

8.81



PF: Factores de influencia en la dificultad del sistema

10. **Reutilización**: indica si gran parte de la funcionalidad del proyecto, está pensada para un uso intensivo por otras aplicaciones.

- Es más complejo hacer: "*software development for reuse*"

GI	Líneas Guía
0	No hay que reutilizar el código.
1	Se emplea código reutilizable dentro de la aplicación.
2	Menos del 10% de la aplicación se considera reutilizable.
3	El 10% o más de la aplicación se considera reusable.
4	La aplicación está específicamente preparada y documentada para facilitar la reutilización y se adapta para cumplir las necesidades del usuario específicas sobre el código fuente.
5	La aplicación está específicamente preparada y documentada para facilitar la reutilización y, además, se adapta para su uso mediante el mantenimiento de parámetros de usuario.

Ruiz & García. G. Costes

8.82



PF: Factores de influencia en la dificultad del sistema

11. Facilidad de Instalación: Durante el desarrollo se consideran factores que facilitan la instalación.

GI	Líneas Guía
0	El usuario no ha declarado consideraciones especiales y no se requiere una configuración especial para la instalación
1	El usuario no ha declarado consideraciones especiales pero se requiere una configuración especial para la instalación
2	El usuario ha declarado consideraciones especiales para la conversión e instalación y se requieren guías probadas de conversión e instalación. El impacto de la conversión sobre el proyecto no se considera importante.
3	El usuario ha declarado consideraciones especiales para la conversión e instalación y se requieren guías probadas de conversión e instalación. El impacto de la conversión sobre el proyecto se considera importante.
4	Además del GI 2, se proporcionan y prueban herramientas para la conversión e instalación automática.
5	Además del GI 3, se proporcionan y prueban herramientas para la conversión e instalación automática.



PF: Factores de influencia en la dificultad del sistema

12. Facilidad de Operación: Se han tenido en cuenta factores de operatividad. Se han considerado procedimientos de arranque, de copia de respaldo y de recuperación.

- Algunos de los parámetros están pensados para sistemas heredados y Mainframes. El estimador debe adaptarlos a las nuevas tecnologías

GI	Líneas Guía
0	No hay consideraciones especiales de operación aparte de los procesos normales de back-up establecidos por el usuario.
1-4	Uno, alguno, o todos los elementos siguientes se aplican. Cada elemento vale 1: - Se proporcionan procesos efectivos de arranque, back-up y recuperación pero se requiere la intervención del operador - Lo anterior pero sin requerirse la intervención del operador (cuenta como 2) - La aplicación minimiza la necesidad de montaje de cintas. - La aplicación minimiza la necesidad de manejo de papel.
5	La aplicación debe diseñarse para una operación totalmente automática. La recuperación automática ante errores es una funcionalidad de la aplicación.



PF: Factores de influencia en la dificultad del sistema

13. Múltiples Localizaciones: la aplicación se diseña para soportar múltiples instalaciones en diferentes entornos y organizaciones.

GI	Líneas Guía
0	No hay requisitos de usuario para soporte a la instalación en más de un lugar.
1	Se consideran en el diseño múltiples instalaciones pero para operar con idéntica configuración (tanto HW como SW)
2	Se consideran en el diseño múltiples instalaciones pero para operar con similar configuración HW y/o SW
3	Se consideran en el diseño múltiples instalaciones pero para operar con diferente configuración HW y/o SW
4	Se cumple GI 1 o 2 y se proporcionará documentación y plan de soporte debidamente probados para soportar la aplicación en múltiples sitios.
5	Se cumple GI 3 y se proporcionará documentación y plan de soporte debidamente probados para soportar la aplicación en múltiples sitios.



PF: Factores de influencia en la dificultad del sistema

14. Facilidad de Cambio: Se han tenido en cuenta criterios que facilitarán el posterior mantenimiento. Se aplica:

- Consultas y Generación de Informes Flexible que manejan peticiones simples (ej: sobre un solo archivo interno) → Cuenta como 1
- Lo anterior sobre peticiones de complejidad media (\geq un archivo interno) → Cuenta como 2
- Lo anterior sobre peticiones complejas (combinaciones lógicas and/or sobre uno o más archivo interno) → Cuenta como 3
- Los datos de control de las funciones reside en tablas mantenidas por el usuario mediante procesos interactivos online pero los cambios son efectivos el siguiente día hábil → Cuenta como 1
- Lo anterior pero los cambios tienen efecto inmediato → Cuenta como 2



PF: Factores de influencia en la dificultad del sistema

Ejemplo

- Una empresa de fabricación está desarrollando una aplicación Web para dar soporte técnico a los ingenieros de servicio que están desempeñando las tareas de mantenimiento de la maquinaria. Requisitos:
 - El sistema necesita estar activo las 24h los 7 días de la semana.
 - Actualmente el sistema tiene 3000 usuarios y se calcula que 300 de ellos lo podrían usar de forma concurrente.
 - Turnos de uso del sistema: de 6 a 14 , de 14 a 22 y de 22 a 6. El pico de utilización se alcanza de 9 a 14.
 - El usuario requiere un nivel de rendimiento similar a los otros sistemas, que tienen un tiempo de respuesta ante cargas de trabajo máximas de 2,4 segundos para la mayoría de las transacciones.
 - El sistema debe facilitar el crecimiento fácil de los volúmenes de datos.
 - Se requieren facilidades para el backup y la recuperación
 - El cliente requiere soporte a los navegadores: IE 6.5, Netscape y posiblemente un navegador Mac. Si el sistema soporta múltiples navegadores, el cliente desea especificar la versión que quiere usar en cada momento.



PF: Factores de influencia en la dificultad del sistema

Ejemplo

Requisito no Funcional	GSC	Valor
Tiempo de Funcionamiento	12- Facilidad de Operación	5
Población de Usuarios	5 – Tasas de Transacción Rápidas	5
Requisitos de Rendimiento	3- Rendimiento	4
Crecimiento en Volúmenes de Datos	No aplicable	
Recuperación y Backup de Datos	12 – Facilidad de Operación	5
Entorno	7. Eficiencia del Usuario Final	3-4