

GUÍA DE APRENDIZAJE - TEMA 8 GESTIÓN DE COSTES

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. Estimación de costes.
3. Preparación del presupuesto de costes.
4. Control de costes
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 utilizada:

CON	Connell, S. <i>Desarrollo y Gestión de Proyectos Informáticos</i> . McGraw-Hill Interamericana. España 1997.	Cap. 8
PMI	PMI Guía de los Fundamentos de la Gestión de Proyectos, Tercera Edición, Project Management Institute, 2004.	Cap. 7
USC	University of Southern California. <i>COCOMO II Model Definition Manual</i> . Version 1.4. Disponible en http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf	
PAR	Parthasarathy, M. A. <i>Practical Software Estimation. Function Point Methods for Insurced and Outsourced Projects</i> . Addison-Wesley, 2007.	Cap 3-6

Bibliografía complementaria:

- Boehm, B. y otros. "An Overview of the COCOMO 2.0 Software Cost Model", Software Technology Conference, 1995.
- Dolado, J. J. y Fernández, L. (coordinadores). *Medición para la Gestión en la Ingeniería del Software*. Ra-Ma. España 2000. Cap. 11.
- Gómez, A. y otras. "COCOMO – Un Modelo de estimación de proyectos Software". Trabajo de estudiantes argentinas.

- Garmus, D., Herron, D. Function Point Analysis. Measurement Practices for Successful Software Projects. Addison-Wesley, 2003.

NOTAS DE USO:

- Se indica la referencia de la bibliografía básica (siglas y número de página). Ejemplo: GIL-9/10 significa páginas 9 a 10 de la referencia GIL (ver tabla de bibliografía básica).
- Las transparencias que corresponden se indican entre corchetes. Ejemplo: [T13].
- Se incluyen Ejercicios para realizar por parte de los alumnos. Se identifican con el símbolo ➤.

Contenido:

Introducción

[T5]

- MAPA DEL TEMA RESPECTO A PMBOK.
- Lugar que ocupan en el marco PMI de Gestión de Proyectos los conceptos, técnicas y herramientas del tema:

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

C: conceptos que amplían y extienden lo comentado del proceso en el tema 4, 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.

PMI-157, [T6]

La Gestión de los Costes del Proyecto incluye los procesos involucrados en la planificación, estimación, preparación del presupuesto y control de costes de forma que el proyecto se pueda completar dentro del presupuesto aprobado. Procesos:

- Estimación de Costes: desarrollar una aproximación de los costes de los recursos necesarios para completar las actividades del proyecto.
- 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.
- Control de Costes: influir sobre los factores que crean variaciones del coste y controlar los cambios en el presupuesto del proyecto.

PMI-158

- En algunos proyectos, especialmente los de menor alcance, la estimación de costes y la preparación del presupuesto de costes están tan estrechamente vinculadas que se consideran como un proceso único, que puede ser realizado por una sola persona durante un período de tiempo relativamente corto.

PMI-160, [T7]

Se muestra el diagrama de flujo de los procesos del área de gestión de costes con sus entradas, salidas y relaciones con los procesos de otras Áreas de Conocimiento relacionadas. Como se puede apreciar, para la gestión de costes son entradas fundamentales:

- WBS, DFT, ...
- Recursos Necesarios y Calendarios
- Cronograma del proyecto
- Información Histórica: referida a los costes de las diversas categorías de recursos.

Estimación de Costes

Principales técnicas de Estimación:

- Estimación por analogía.
- Modelos paramétricos.
- Estimación Bottom-up.

PMI-164, [T8]

- 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.

PMI-165, [T9]

- Determinación de Tarifas de Costes de Recursos.
 - La persona que determina las tarifas o el grupo que prepara las estimaciones debe conocer las tarifas de costes unitarios, tales como el coste del personal por hora y el coste del material a granel por yarda o metro cúbico, correspondientes a cada recurso para estimar los costes de la actividad del cronograma.
 - Para los productos, servicios o resultados que deben obtenerse por contrato, se pueden incluir las tarifas estándar con factores de escalamiento en el contrato. Las bases de datos comerciales y las listas de precios publicadas de los vendedores son otra 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* (ascendente):
 - 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.
- PMI-165, [T10]
- 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 más 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).

Elaboración de presupuestos y control de costes

- PMI-170, [T11] y [T12]
- *Línea Base de Costes*:
 - La línea base de costes (cost baseline) es un 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.
 - Suele tener forma de ‘S’, ver [T12].
 - Se puede construir una línea base de costes para cada categoría de costes (personal, servicios, etc.) o para un recurso individual.
- PMI-171, [T13]
- El *Control de Gastos* incluye:
 - Supervisar la realización de gastos para detectar variaciones respecto de lo previsto.
 - Asegurar que todos los cambios son registrados con exactitud en el presupuesto y la línea base de costes.
 - Prevenir que cambios incorrectos, inapropiados, o no autorizados, sean incluidos en el presupuesto y la línea base de costes.
 - Informar adecuadamente a los clientes de los cambios autorizados.
 - También incluye los “porqués” de las variaciones, positivas o negativas.
 - Las salidas (outputs) de este proceso son:
 - *Revisiones de las estimaciones* de costes.
 - *Modificaciones del presupuesto*: son cambios importantes, que afectan a una línea base de costes ya aprobada.
 - *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*: [T14] 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*.
 - Ver PMI 173-176

Introducción a la estimación del software

CON-187, [T15]

Medición vs Estimación del Software

CON-188, [T16]

- 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 duración (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

CON-181, [T17]

- La afirmación última se debe a que:
 - 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.

CON-182, [T18]

- Las estimaciones en proyectos software tienen rangos de precisión previsible, que se van reduciendo a lo largo de la duración del proyecto: ver figura en [T18].
 - 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 2², ...

Estimación del tamaño

CON-190, [T19]

- 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.
- **Ejercicio:** Establecer la diferencia entre tamaño y longitud de un software haciendo un símil con el cuerpo de una persona.

CON-189, [T19]

- 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 localizan 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

[T20]

- 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, ...

CON-197, [T20] y [T21]

- 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. CON-212, tabla en [T22]
- 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 de la duración

CON-198, [T23] y [T24]

- 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 CON-212, tabla en [T22]
 - utilización de funciones de equivalencia semiempíricas del tipo duración = función del esfuerzo, que incluyen diversos parámetros cuyos valores se determinan empíricamente.
 - Por ejemplo, diversos autores han propuesto la siguiente fórmula:
 - Duración en meses = $3 \cdot 0 \times \text{personas-mes}^{(1/3)}$
 - utilización de software de gestión de proyectos que permita realizar una planificación de la duración optimizando la utilización de los recursos disponibles (por ejemplo, MS Project).

Resumen de tipos de técnicas para estimar software

[T24] (ver libro de Dolado et al.)

- 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.

Estimación del Tamaño mediante Puntos Función

[T25] (ver libros de Dolado et al., Parthasarathy et al.)

- Para estimar el tamaño de un producto software se han propuesto diversas técnicas:
- 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).

NOTA: En este tema se explicará el proceso de cálculo de PF según el IFPUG (International Function Points Users Group). En detalle, esta técnica se explica en el libro de Parthasarathy et al. De forma complementaria, se pueden consultar las tablas de factores de complejidad en el documento de Técnicas y Prácticas de la metodología MÉTRICA 3.

En los ejercicios de teoría y prácticas de laboratorio con USC COCOMO II se debe aplicar esta técnica pero empleando las tablas de complejidad indicadas en el modelo COCOMO (que se indican a continuación y se pueden consultar con detalle en los apuntes de Gómez y otros), que no coinciden al 100% con las indicadas en METRICA 3.

CON-189 [T26]

- 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
 - 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.

[T28]

En la figura se observa el proceso de cálculo de PF de acuerdo al IFPUG.

[T29]

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:
 - o Descripción de requisitos expresada en el Pliego de Prescripciones Técnicas (PPT).
 - o 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
 - o 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).
 - o 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.

[T30]

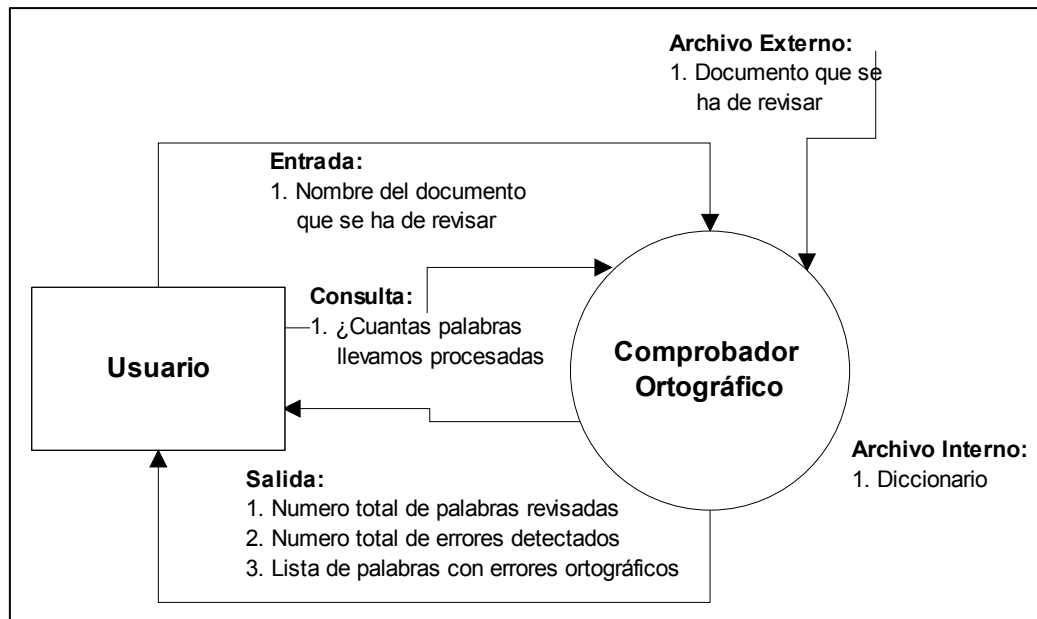
En la figura se ilustran las dos primeras etapas del proceso de cálculo definidas por el IFPUG.

CON-189, USC-24 [T31]

- En la segunda 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 (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.

[T32]

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



PAR-76, 80-81, [T33]

Archivos Lógicos Internos (ILF)

- Un ILF es un grupo de datos relacionados lógicamente o información de control reconocida por el usuario y mantenido dentro de los límites de la aplicación.
- El objetivo fundamental de un ILF es almacenar datos mantenidos a través de uno o más procesos elementales (o acciones) de la aplicación que se está midiendo. Se corresponde con las entidades que típicamente aparecen en un diagrama Entidad-Interrelación (ER).
- La información de control se refiere a datos que influyen en un proceso elemental de la aplicación que se está midiendo. Especifica qué, cuándo, o cómo los datos van a ser procesados.
- Por ejemplo las preferencias de usuario que almacena una aplicación se pueden considerar información de control.

Reglas de Identificación:

- R1: El grupo de datos o información de control es un grupo de datos lógico identificable por el usuario.
- R2: El grupo de datos es mantenido por un proceso elemental, dentro de los límites de la aplicación que se está midiendo.

Ejemplo: Supongamos que en una aplicación de alquiler de automóviles un usuario requiere la capacidad de "Introducir, consultar y listar información sobre los datos de cada automóvil (matrícula, marca, modelo, año de fabricación) y los datos de las personas que los alquilan (DNI, apellidos, nombres, dirección, localidad)".

- En este caso se distinguen dos grupos de datos lógicos (ILFs), dos entidades:
 1. Datos de automóviles
 2. Datos de las personas

PAR 76, 81 [T34]

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.
- La principal diferencia entre ILF y un terno ELF es que el ILF es mantenido dentro de los límites de la aplicación mientras que el ELF es mantenido por otra aplicación. Esto significa que un ELF medido para una aplicación debe ser un ILF en otra aplicación.

Reglas de Identificación:

- R1: El grupo de datos o información de control es un grupo de datos lógico identificable por el usuario.
- R2: El grupo de datos es referenciado por la aplicación que se está midiendo, y es externo a ella.
- R3: El grupo de datos no es mantenido por la aplicación que se está midiendo.
- R4: El grupo de datos es mantenido en un ILF 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).

[T35]

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.	

PAR 103-108, [T36]

Entradas Externas (EI):

- Proceso elemental que procesa datos o información de control que proviene de fuera de los límites de la aplicación

- Mantiene uno o más ILFs y/o modifica el comportamiento del Sistema

Reglas de los procesos elementales por cada entrada externa:

- R1: Los datos o información de control se reciben desde fuera
- R2: Al menos se actualiza un ILF si los datos introducidos en la aplicación no contienen información de control
- R3: El proceso es la unidad más pequeña de actividad que es significativa para el negocio del usuario final.
- R4: El proceso identificado debe verificar alguna de estas reglas: Su lógica de proceso es única respecto de otras entradas externas de la aplicación. Los elementos de datos identificados son distintos a los de las otras entradas externas de la aplicación. Los ILFs y ELFs utilizados son diferentes de los archivos utilizados por otras entradas externas de la aplicación.

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.

PAR 103-108 [T37]

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).

Reglas de los procesos elementales por cada salida externa:

- R1: La función envía información de control o datos fuera de los límites.
- R2: Una de las tres siguientes reglas deben aplicarse al proceso identificado:
 - El proceso lógico es único para el tratamiento lógico realizado para otras EO.
 - El conjunto de datos elementales identificado es diferente de los conjuntos identificados por otras EO.
 - Los ILFs y ELFs utilizados son diferentes de los archivos utilizados por otras EO
- Debe aplicarse, además una de las siguientes reglas:
 - R3: La lógica de proceso del proceso elemental contiene al menos un cálculo o fórmula matemática.
 - R4: La lógica del proceso del proceso elemental crea datos derivados.
 - R5: La lógica de proceso del proceso elemental mantiene al menos un ILF.
 - R6: La lógica de proceso del proceso elemental altera el funcionamiento del sistema.

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.

PAR 103-108 [T38]

Consultas Externas (EQ):

- El EQ es un proceso elemental que envía datos o información de control fuera de los límites de la aplicación. El objetivo principal de un EQ es presentar información al usuario a través de la recuperación de datos o información de control de un ILF o ELF.
- A diferencia del EO, el procesamiento lógico no debe contener ninguna fórmula o cálculo matemático, ni tampoco debe crear datos derivados de los obtenidos.
- Por otra parte ningún ILF es mantenido mientras se procesa la acción de un EQ ni tampoco el comportamiento del sistema se ve alterado.
- Es una combinación de entrada/salida que se obtiene de una búsqueda de datos, no actualiza ficheros lógicos y no contiene datos derivados (aquellos que requieren un proceso distinto a búsqueda, edición o clasificación).

En resumen, un EQ se encarga sólo de recuperar datos almacenados y proporcionárselos al usuario.

Reglas:

- R1: La función envía información de control o datos fuera de los límites
- R2: Una de las tres siguientes reglas deben aplicarse al proceso identificado:
 - El proceso lógico es único para el tratamiento lógico realizado por otras EQ
 - El conjunto de datos elementales es diferente de los conjuntos de otras EQ
 - Los ILFs y ELFs utilizados son diferentes de los archivos utilizados por otras EQ
- R3: La lógica del proceso elem. recupera datos o inf. control de un ILF/ELF.
- R4: La lógica del proceso elem. no contiene cálculo o fórmula matemática.
- R5: La lógica de proceso del proceso elemental no crea datos derivados.
- R6: La lógica del proceso del proceso elemental no mantiene un ILF.
- R7: La lógica de proceso del proceso elemental no altera el funcionamiento del sistema.

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.

[T39]

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.	

PAR 109-110 [T40]

Para hacer menos subjetiva la complejidad, algunas técnicas incluyen tablas orientativas (ver T[44]) 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),
 - o Ejemplo: Archivo con los campos: nombre de empleado, dirección y edad → 3 DETs
- Nº de tipos de registros referenciados:
 - o nº de tipos de archivos lógicos referenciados (sólo para Entradas, Salidas y Consultas)
 - o nº de archivos lógicos internos referenciados (RET, Record Element Type) (sólo para archivos)
 - o 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

PAR 109-111 [T41]

- DET. Reglas:
- “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”
- “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:
 - 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

PAR 109-111, 113-119 [T42] y [T43]

RET.

- “Contar un RET para cada subgrupo en un ILF o ELF opcional u obligatorio”.
- “Si no hay subgrupos contar como 1 RET”

Ejemplo de cálculo de RET

USC-25 [T44]

- Etapa 3. Clasificar las funciones de usuario en tres niveles de complejidad: baja, media, alta.
 - 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 elementales incluidos, y
 - Nº de ficheros 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
<i>Complejidad de las Entradas</i>			

- La tabla de complejidad para las Salidas y para 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
<i>Complejidad de las Salidas y Consultas</i>			

- En el caso de informes, algunos autores definen la complejidad de manera diferente:
 - Reducida: Informes de una o dos columnas. Pocas transformaciones entre datos.
 - Normal: Informes con múltiples columnas que pueden incluir subtotales. Múltiples transformaciones de datos.
 - Elevada: 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 la Salidas).
- 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
<i>Complejidad de los Archivos y las Interfaces</i>			

USC-24/25 [T45]

Cuarta Etapa. Cálculo de los PF 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

- USC-24/25, [T46] (ver apuntes de Gómez y otros)
- Puntos Función Totales:
 - $PF = PFSA * (0,65 + 0,01 * SUMA(GI))$
 - Siendo 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 (mucho).
 - 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.

- USC-24/25, [T47] (ver apéndice del tema, METRICA 3 y apuntes de Gómez y otros)
- Factores de influencia en la dificultad del sistema:
 1. **Comunicaciones de datos:** concierne a la transmisión de datos o información de control, enviados o recibidos mediante algún sistema de comunicaciones.
 2. **Procesamiento distribuido:** concierne 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
 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.

USC-26, [T48]

- existen tablas de equivalencias entre PF y líneas de código para diversos lenguajes de programación:

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

NOTA: en la web de la asignatura se incluye un archivo Excel que incluye hojas para lo siguiente: cálculo de los puntos función sin ajustar y ajustados; tablas de complejidades de cada tipo de función de usuario; lista de factores de influencia; equivalencias PF y líneas de código para cada lenguaje.

Método COCOMO para estimación del software

USC-1, [T49]

- 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 famoso y 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, ...

Características de COCOMO II.

USC-1, [T50]

- Los principales objetivos de la nueva versión COCOMO II son:
 - Desarrollar un modelo de estimación de costes y tiempos en consonancia con las prácticas actuales de ciclo de vida del software.
 - Construir una base de datos y una herramienta de costes del software que incluya capacidades para la mejora continua del modelo.
 - 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.

USC-2/3, [T51] y [T52]

- Ha sido diseñado pensando en un mercado del software con varios sectores:

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.
- Los desarrolladores deben conocer bien la tecnología (producida por el sector de 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 (SGBD, 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.

- 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.).

USC-2/4, [T53]

- Modelos para cada sector de mercado:
- *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).

USC-4/6, [T54]

- 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.

USC-4/6, [T55]

- 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:
- Proveer información más exacta sobre los generadores de costes, y
- 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).

USC-31/32, [T56] y [T57]

- COCOMO II utiliza tres métricas de tamaño diferentes:
 - Puntos Objeto (PO): (Object Points – PO en inglés) contadores de pantallas, informes y módulos 3GL, cada uno afectado de un peso según un factor de complejidad.
 - Puntos Función No Ajustados (PFNA): (Unadjustment Function Point – UFP en inglés) PF sin tener en cuenta los 14 factores de influencia en la dificultad del sistema [T33] porque ya se consideran mediante los conductores de coste de COCOMO; y

- Líneas de Código Fuente (LDCF): (Source Lines Of Code – SLOC en inglés) se contabilizan según la propuesta del Software Engineering Institute (SEI).

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	<input checked="" type="checkbox"/>		
Statement type	Definition <input checked="" type="checkbox"/>	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				
3 Declarations			2	<input checked="" type="checkbox"/>
4 Compiler directives			3	<input checked="" type="checkbox"/>
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 <input checked="" type="checkbox"/>	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 <input checked="" type="checkbox"/>	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				

Estimación del esfuerzo con COCOMO II

USC-6/7, [T58]

- 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 2⁹⁴).
- 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 1⁰ (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).

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

[T59]

- **Ajuste del tamaño:**
- El tamaño del software a desarrollar no es siempre igual al tamaño nominal. COCOMO II incorpora ajustes por 4 causas:
 - Breakage (desecho)
 - Reutilización,
 - Reingeniería o conversión, y
 - Mantenimiento.

USC-7, [T59]

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

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

USC-7/11, [T60]

- **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):

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

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

- Con:
- ASLOC = cantidad de LDCF adaptadas de software existente,

- AA (assesment and assimilation) = grado de valoración y asimilación necesarios para decidir cuando un modulo 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, cuyo valor es:

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

- siendo
- DM = % de modificación del diseño,
- CM = % de modificación del código,
- IM = % del esfuerzo de integración original requerido para integrar el software reutilizado,

USC-11/12, [T61]

- **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, y
- ATPROD: productividad de las herramientas en LDCF/PM (actualmente se estima en 2400).

USC-12, [T61]

- **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, para calcular el esfuerzo de mantenimiento con la fórmula ya conocida:

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

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

USC-13, [T62]

- **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).

USC-26/30, [T62]

- 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.

USC-33/40, [T62]

- 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.

USC-33/40, [T63]

- 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.

USC-33/34, [T64]

Factores del Producto

USC-34/37, [T65]

Factores del Personal y Plataforma

USC-37-38, [T66]

Factores del Proyecto

USC-15/20, [T67] y [T68]

- **Factores de escala:**
- afectan al exponente B en la ecuación principal de estimación del esfuerzo [T58].
- Son 5:
- 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 con COCOMO II

USC-13, CON2-10, [T69]

- 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

[T70]

La herramienta USC-COCOMO se estudiará en laboratorio

- **Ejercicios:** El alumno debe aprender bien las técnicas de puntos función para estimación de tamaño y COCOMO II para estimación de esfuerzo. Para ello se recomiendan las siguientes **actividades de aprendizaje:**
 - 1) Leer el método de estimación de puntos función de la guía de aprendizaje, que sigue las directrices del IFPUG. Como complemento leer el documento de METRICA 3 “Técnicas y Prácticas” en su parte dedicada a la técnica de Albrecht de Puntos Función.
 - 2) Practicar la técnica de puntos función, buscando aclarar las dudas, con alguno de los módulos del ejercicio de ejemplo disponible en la web.
 - 3) Realizar el trabajo de prácticas, para estimar los puntos función del proyecto, utilizando como ayuda el archivo Excel disponible.
 - 4) Leer los apuntes de Gómez y otros en la parte de COCOMO II.
 - 5) Practicar el modelo EDM de COCOMO II con los módulos elegidos anteriormente en el ejercicio de ejemplo.
 - 6) Realizar el trabajo de prácticas con USC COCOMO II.