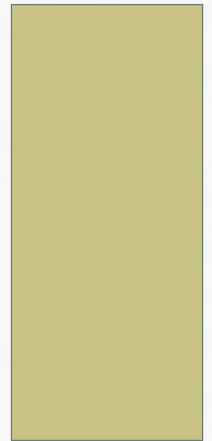


INGENIERÍA DEL SOFTWARE

UNIDAD 2: MEDIDAS, MÉTRICAS E INDICADORES DEL PROCESO
CICLO LECTIVO 2013



OBJETIVOS DE LA CLASE

- **Métricas del proceso, del proyecto y del producto.**
- **Mediciones directas e indirectas.**
- **Métricas de calidad.**
- **Métricas de punto de función.**
- **Otras métricas.**
- **Estimaciones: Modelo Cocomo.**

TEMAS PARA LA PRÓXIMA CLASE

- **COCOMO II – Software Cocomo.**
- **Estimación Delphi.**

MEDICIONES DEL SOFTWARE

- Las mediciones del mundo físico se pueden caracterizar de dos formas: medidas directas (la longitud de un tornillo) e indirectas (la calidad de un tornillo). Las métricas del software se pueden caracterizar de igual forma.
- Las medicas directas del proceso de ingeniería del software incluyen el costo y el esfuerzo aplicados.
- Pensemos ejemplos de métricas directas que vimos:
 - LCD.
 - Defectos informados, etc.
- ¿Y métricas indirectas? ...
 - Funcionalidad, calidad, eficiencia, facilidad de mantenimiento, etc.

MEDICIONES DEL SOFTWARE

- El costo y el esfuerzo para producir software, LCD, y otras medidas directas son relativamente fáciles de reunir. Las formas de evaluarlas se establecen mas adelante.
- Sin embargo, la calidad, eficiencia, funcionalidad son mas difíciles de evaluar y solo pueden ser medidas de forma indirecta.

MÉTRICAS DEL SOFTWARE

- El dominio de las métricas del software se divide en métricas del proceso, del proyecto y del producto.
- Las métricas del producto, que son **privadas** para un individuo, se combinan para desarrollar métricas del proyecto que son **públicas** para un equipo de software.
- Las métricas del proyecto se consolidan para crear métricas del proceso que sean **públicas** para toda la organización del software.


Proyecto	LDC	Esfuerzo	Coste £(000)	Pag. Doc.	Errores	Defectos	Personas
Alfa	12,100	24	168	365	134	29	3
Beta	27,200	62	440	1224	321	86	5
Gamma	20,200	43	314	1050	256	64	6

- ¿Recuerdan la diferencia entre error y defecto?
- A partir de estos datos obtenidos por una organización del software se pueden desarrollar para cada proyecto métricas simples orientadas al tamaño.
 - Errores por KLDC.
 - Defectos por KLDC.
 - Paginas de documentación por KLDC

MÉTRICAS ORIENTADAS A LA FUNCIÓN

- Las métricas del software orientadas a la función utilizan una medida de la funcionalidad entregada por la aplicación como un valor de normalización. Ya que la **funcionalidad** no se puede medir directamente, se debe derivar indirectamente mediante otras medidas.
- Se utiliza una medida llamada **punto de función**.
- Se determinan cinco características de dominios de información y se proporcionan las cuentas en la posición apropiada de la tabla.

MÉTRICAS ORIENTADAS A LA FUNCIÓN

Parámetros de medición	Factor de ponderación					
	Cuenta		Simple	Medio	Complejo	
Número de entradas de usuario	<input type="text"/>	×	3	4	6	= <input type="text"/>
Número de salidas de usuario	<input type="text"/>	×	4	5	7	= <input type="text"/>
Número de peticiones de usuario	<input type="text"/>	×	3	4	6	= <input type="text"/>
Número de archivos	<input type="text"/>	×	7	10	15	= <input type="text"/>
Número de interfaces externas	<input type="text"/>	×	5	7	10	= <input type="text"/>
Cuenta total						<input type="text"/>

MÉTRICAS ORIENTADAS A LA FUNCIÓN

Los valores de los dominios de información se definen de la forma siguiente:

- **Número de entradas de usuario.** Se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación.
- **Número de salidas de usuario.** Se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto la salida se refiere a informes, pantallas, mensajes de error, etc.
- **Número de peticiones de usuario.** Una petición se define como una entrada interactiva que produce la generación de alguna respuesta del software inmediata en forma de salida interactiva.
- **Número de archivos.** Se cuenta cada archivo maestro lógico (esto es, un grupo lógico de datos que puede ser una parte de una gran base de datos o un archivo independiente).
- **Número de interfaces externas.** Se cuentan todas las interfaces legibles por la máquina (por ejemplo: archivos de datos de cinta o disco) que se utilizan para transmitir información a otro sistema.

MÉTRICAS ORIENTADAS A LA FUNCIÓN

- Una vez que se han recopilado los datos anteriores, a la cuenta se asocia un valor de complejidad.
- Las organizaciones que utilizan métodos de puntos de función desarrollan criterios para determinar si una entrada en particular es simple, media o compleja.
- No obstante la determinación de la complejidad es algo subjetiva.

CARACTERÍSTICAS DE LOS PF

Responder a cada una de las siguientes catorce preguntas y asignarles un valor entre 0 y 5, donde 0 es no influencia, 1 es incidental, 2 es moderado, 3 es medio, 4 es significativo y 5 es esencial.

1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?
2. ¿Requiere comunicación de datos?
3. ¿Existen funciones de procesamiento distribuido?
4. ¿Es crítico el rendimiento?
5. ¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?
6. ¿Requiere entrada de datos interactiva?
7. ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?

CARACTERÍSTICAS DE LOS PF

8. ¿Se actualizan los archivos maestros de forma interactiva?
9. ¿Son complejas las entradas, las salidas, los archivos o las peticiones?
10. ¿Es complejo el procesamiento interno?
11. ¿Se ha diseñado el código para ser reutilizable?
12. ¿Están incluidas en el diseño la conversión y la instalación?
13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14. ¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?

Sumar los puntos asignados a cada respuesta y obtener un total F que indica un valor de ajuste de complejidad.

CÁLCULO DE PF

- El punto de función PF se calcula con la siguiente ecuación: **$PF = T * (0.65 + 0.01 * F)$** .
- Donde T es la suma de todos los puntos de Función no ajustados.
- F es la sumatoria de las características de los puntos de función. (o Grado Total de Influencia)
- A partir de esto, obtenemos los Puntos de Función ajustados.
- En esta formula, llamamos factor de ajuste a:

$$\text{Factor de Ajuste (VAF)} = \mathbf{0.65 + 0.01 * F}$$

EJERCICIO

- Calcule el valor del punto de función de un proyecto con las siguientes características del dominio de información:
- Número de entradas de usuario: 32
- Número de salidas de usuario: 60
- Número de peticiones de usuario: 24
- Número de archivos: 8
- Número de interfaces externos: 2

Asuma que todos los valores de ajuste de complejidad están en la media.

TÉCNICA DE PF

La técnica de puntos de función fue introducida por Albrecht (1979) y su propósito es medir el software cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema. Los objetivos de los puntos de función son:

- Medir lo que el usuario pide y lo que el usuario recibe.
- Medir independientemente la tecnología utilizada en la implantación del sistema.
- Proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y la productividad.
- Proporcionar un medio para la estimación del software.
- Proporcionar un factor de normalización para la comparación de distintos software.

MÉTRICAS AMPLIADAS DE PF

- Los puntos de función fueron diseñados originalmente para sistemas de información y por eso le da más importancia a los datos y menos a los aspectos de control y funcionales. Para subsanar esto, se han propuesto los puntos de característica y los puntos de función 3D.
- **Puntos de característica:** Se calcula igual que el punto de función y solo agrega la cuenta de algoritmos. En este contexto se define un algoritmo como un problema de cálculo limitado que se incluye dentro de un programa de computadora específico. Invertir una matriz, decodificar un string o manejar una interrupción son ejemplos de algoritmos.

PUNTOS DE FUNCIÓN 3D

Los puntos de función 3D se calculan llenando la siguiente tabla:

Elemento de Medición	Pesos de Complejidad			Subtotal
	Bajo	Medio	Alto	
Estructuras internas de datos	*7	*10	*15	
Datos externos	*5	*7	*10	
N° de entradas de usuarios	*3	*4	*6	
N° de salidas de usuarios	*4	*5	*7	
N° de peticiones de usuarios	*3	*4	*6	
Transformaciones	*7	*10	*15	
Transiciones				
Total Puntos de Función 3D				

PUNTOS DE FUNCIÓN 3D

- Estructuras internas de datos. Son arreglos, listas ligadas, pilas, colas, etc.
- Datos externos. Equivale a la suma de los archivos y las interfaces externas tal y como están definidos para el punto de función.
- Entradas de usuario. Definidas igual que para el punto de función.
- Salidas de usuario. Definidas igual que para el punto de función.
- Peticiones de usuario. Definidas igual que para el punto de función.
- Transformaciones. Son las operaciones internas requeridas para transformar datos de entrada en datos de salida. Multiplicar dos matrices cuenta como una transformación. Leer datos de un archivo y guardarlos en memoria no.
- Transiciones. Ocurre cuando el software pasa de un estado a otro como resultado de algún suceso. En un sistema de altas, bajas y cambios, al tomar la opción de altas, pasa del estado "menú principal" al estado "procesa altas" y puede ser que en ese momento pida datos para dar la alta.

A PENSAR!!!!

Calcule el valor del punto de función de un sistema empotrado con las características siguientes:

- Estructuras de datos interna: 6
- Estructuras de datos externa: 3
- Número de entradas de usuario: 12
- Número de salidas de usuario: 60
- Número de peticiones de usuario: 9
- Número de interfaces externos: 3
- Transformaciones: 36
- Transiciones: 24

Asuma que la complejidad de las cuentas anteriores se divide de igual manera entre bajo, medio y alto.

PUNTOS DE FUNCIÓN 3D

- Luego que se realizaron los anteriores cálculos, se realizan los mismos pasos que para puntos de función.
- El punto de función (y sus extensiones), como la medida LDC, es controvertido. Los partidarios afirman que PF es independiente del lenguaje de programación, lo cual es ideal para aplicaciones que utilizan lenguajes convencionales y no procedimentales; esto se basa en los datos que probablemente se conocen al principio de la evolución de un proyecto, y así el PF es más atractivo como enfoque de estimación.

RECONCILIACIÓN DE LAS DIFERENTES MÉTRICAS

- La relación entre las líneas de código y los puntos de función depende del lenguaje de programación que se utilice para implementar el software, y de la calidad del diseño.
- La tabla siguiente proporciona estimaciones informales del número medio de líneas de código que se requiere para construir un punto de función en varios lenguajes de programación:

LENGUAJE	LDC/PF
Ensamblador	320
C	150
Cobol	105
Pascal	91
Prolog/LISP	64
C++	64
FoxPro	31
Visual Basic	30
Delphi	29

EJERCICIO

- El software utilizado para controlar una fotocopidora avanzada requiere 32.000 líneas de C y 4.200 líneas de C++.
- Estime el número de puntos de función del software de la fotocopidora.

MÉTRICAS PARA LA CALIDAD

- Aunque hay muchas medidas de la calidad de software, **la corrección, facilidad de mantenimiento, integridad, y facilidad de uso** proporcionan indicadores útiles para el equipo del proyecto.
- **Corrección.** Un programa debe operar correctamente o proporcionará poco valor a sus usuarios. La corrección es el grado en el que el software lleva a cabo su función requerida. La medida más común de corrección es defectos por KLDC, en donde un defecto se define como una falta verificada de conformidad con los requisitos.

MÉTRICAS PARA LA CALIDAD

- **Facilidad de mantenimiento.** El mantenimiento del software cuenta con más esfuerzo que cualquier otra actividad de ingeniería del software. La facilidad de mantenimiento es la facilidad con la que se puede corregir un programa si se encuentra un error, se puede adaptar si su entorno cambia, o mejorar si el cliente desea un cambio de requisitos. No hay forma de medir directamente la facilidad de mantenimiento; por consiguiente, se deben utilizar medidas indirectas. Una simple métrica orientada al tiempo es el **tiempo medio de cambio (TMC)**, es decir el tiempo que se tarda en analizar la petición de cambio, hasta aplicar.

MÉTRICAS PARA LA CALIDAD

- **Integridad.** En esta época de hackers y firewalls , la integridad del software ha llegado a tener mucha importancia. Este atributo mide la capacidad de un sistema para resistir ataques (tanto accidentales como intencionados) contra su seguridad. El ataque se puede realizar en cualquiera de los tres componentes del software: programas, datos y documentos. **Amenaza** es la probabilidad (que se puede estimar o deducir de la evidencia empírica) de que un ataque de un tipo determinado ocurra en un tiempo determinado. La **seguridad** es la probabilidad (que se puede estimar o deducir de la evidencia empírica) de que se pueda repeler el ataque de un tipo determinado. La integridad del sistema se puede definir como:

$$\text{Integridad} = \sum [(1 - \text{amenaza}) \times (1 - \text{seguridad})]$$

MÉTRICAS PARA LA CALIDAD

- **Facilidad de uso.** El calificativo amigable con el usuario se ha convertido en omnipresente en las discusiones sobre productos de software. Si un programa no es amigable con el usuario frecuentemente está abocado al fracaso, incluso aunque las funciones que realice sean valiosas. La facilidad de uso es un intento de cuantificar lo amigable que puede ser con el usuario y se puede medir en función de cuatro características:
 - habilidad intelectual y/o física requerida para aprender el sistema;
 - el tiempo requerido para llegar a ser moderadamente eficiente en el uso del sistema;
 - aumento neto en productividad (sobre el enfoque que el sistema reemplaza) medida cuando alguien utiliza el sistema moderadamente y eficientemente;
 - valoración subjetiva (a veces obtenida mediante un cuestionario) de la disposición de los usuarios hacia el sistema.

METRICA DE CALIDAD: EED

- Una métrica de la calidad que proporciona beneficios tanto a nivel del proyecto como del proceso, **es la eficacia de la eliminación de defectos (EED)**.
- EED es una medida de la habilidad de filtrar las actividades de la garantía de calidad y de control al aplicarse a todas las actividades del marco de trabajo del proceso.

$$\text{EED} = E / (E + D)$$

- **El valor ideal de EED es 1.** Esto es, no se han encontrado defectos en el software. De forma realista, D será mayor que cero, pero el valor de EED todavía se puede aproximar a 1. Cuando E aumenta (para un valor de D dado), el valor total de EED empieza a aproximarse a 1. **De hecho, a medida que E aumenta, es probable que el valor final de D disminuya**

MÉTRICAS DEL MODELO DE ANÁLISIS: CALIDAD DE LA ESPECIFICACIÓN

- Propuestas por Alan Davis en 1993.
- Corresponden a la calidad de la especificación de requerimientos.
- Incluyen las siguientes características:
 - Especificidad, compleción, corrección, comprensión, verificación, consistencia, logro, concisión, trazabilidad, modificación, exactitud, reutilización.
- Cualitativas, pero algunas pueden medirse.

MÉTRICAS DEL MODELO DE ANÁLISIS: CALIDAD DE LA ESPECIFICACIÓN [DAVIS, 1993]

- Sea

$$N_r = N_f + N_{nf}$$

donde

- N_r = Número de requerimientos
- N_f = Número de requerimientos funcionales
- N_{nf} = Número de requerimientos NO funcionales

→ **Especificidad** (ausencia de ambigüedad)

$$E = N_{ui} / N_r, \text{ donde}$$

- N_{ui} = número de requerimientos donde todos los revisores tuvieron interpretaciones idénticas.

MÉTRICAS DEL MODELO DE ANÁLISIS: CALIDAD DE LA ESPECIFICACIÓN [DAVIS, 1993]

→ Grado de Validación de Requerimientos

$$V = N_c / (N_c + N_{nv}), \text{ donde}$$

- N_c = número de requerimientos validados como correctos.
- N_{nv} = número de requerimientos no validados aún.

MÉTRICAS DE MANTENIMIENTO

- En el estándar IEEE 982.1-1998, se propone un Índice de Madurez del Software (IMS).
 - Proporciona una indicación de la estabilidad del producto de software, basada en los cambios de cada versión del producto.
 - $IMS = [M_t - (F_c + F_a + F_d)] / M_t$, donde
 - M_t = número de módulos en la versión actual
 - F_c = número de módulos en la versión actual que han cambiado
 - F_a = número de módulos en la versión actual que se han añadido
 - F_d = número de módulos en la versión actual que han eliminado

MÉTRICAS DE DOCUMENTACIÓN

- Índice de Fog:
 - Para evaluar la dificultad de lectura de un texto.
 - Medida de la longitud promedio de las palabras y declaraciones en los documentos.
 - Cuanto mayor sea el índice de Fog, más difícil será comprender el documento.

$$IF = 0,4 \times \left[\left(\frac{\text{Total de Palabras}}{\text{N}^\circ \text{ de Frases}} \right) + 100 \left(\frac{\text{N}^\circ \text{ de palabras de 3 o más sílabas}}{\text{Total de Palabras}} \right) \right]$$

Si $IF > 12 \rightarrow$ Texto no comprensible

COCOMO - INTRODUCCIÓN

- Es un modelo de estimación de costes.
- Creado por Barry W. Boehm.
- Incluye 3 submodelos con un nivel de detalle cada vez mayor

COCOMO - CARACTERÍSTICAS

- Está basado en modelos de estimaciones matemáticas.
- Está orientado al producto final, no a fases intermedias.
- Se basa en la cantidad de líneas de código del proyecto.

INCONVENIENTES DEL MODELO

- Comentarios en líneas de código.
- Estimaciones sobre un nº de líneas de código variable.
- No se le da importancia a la productividad, referente a los hábitos de trabajo
- Dificultad para contemplar costes de revisiones, reuniones...

MODELOS DE ESTIMACIÓN

- Modelo básico
- Modelo intermedio
- Modelo avanzado

COCOMO - MODOS

- **Orgánico:** proyectos relativamente sencillos, menores de 50.000 líneas de código. Se tiene experiencia en proyectos similares y se encuentra en un entorno estable.
- **Semiorgánico:** proyectos intermedios en complejidad y tamaño. La experiencia en este tipo de proyectos es variable, y las restricciones intermedias.
- **Empotrado:** proyectos bastante complejos, en los que apenas se tiene experiencia y en un entorno de gran innovación técnica. Se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

ESTIMACIÓN POR MODELO COCOMO

- Para estimar costos, esfuerzo y tiempo requerido para desarrollar el proyecto, se utilizará el modelo de COCOMO (COnstructive COst MOdel).
- Para calcular el Esfuerzo, necesitaremos hallar la variable KDLC (Kilo-líneas de código), donde necesitaremos conocer las líneas por cada punto de función según el lenguaje utilizado.

KDLC

LENGUAJE

LDC/PF

Ensamblador

320

C

150

Cobol

105

Pascal

91

Prolog/LISP

64

C++

64

FoxPro

31

Visual Basic

30

Delphi

29

LAS FORMULAS A APLICAR SON

- **$E = \text{Esfuerzo} = a \text{ KLDC}^e * \text{FAE}$** (persona x mes)
- **$T = \text{Tiempo de duración del desarrollo} = c \text{ Esfuerzo}^d$** (meses)
- **$P = \text{Personal} = E/T$** (personas)
- Si el desarrollo se realiza en FoxPro, y la cantidad de PFA es de 2350, entonces el KLDC seria: $(31 * 2350) / 1000$.

$$\text{KLDC} = 72,85$$

Este proyecto será considerado como software semiorganico, ya que posee más de 50.000 líneas de código, y además el mismo presenta una complejidad y tamaño intermedio.

COEFICIENTES A UTILIZAR POR TIPO DE PROYECTO

PROYECTO SOFTWARE	a	e	c	d
Orgánico	3.2	1.05	2.5	0.38
Semi - acoplado	3.0	1.12	2.5	0.35
Empotrado	2.8	1.2	2.5	0.32

- También, se requiere conocer el valor de la variable FAE (Factor de ajuste de esfuerzo) que se obtiene mediante la multiplicación de los valores evaluados en los 15 conductores de coste que se observan en la tabla siguiente

FAE

CONDUCTORES DE COSTE	VALORIZACIÓN					
	Muy bajo	Bajo	Normal	Alto	Muy alto	Extr. Alto
Atributos del producto						
Fiabilidad requerida	0.75	0.88	1	1.15	1.4	-
Volumen de la base de datos	-	0.94	1	1.08	1.16	-
Complejidad	0.7	0.85	1	1.15	1.3	1.65
Atributos del ordenador						
Limitaciones de tiempo de ejecución	-	-	1	1.11	1.3	1.65
Limitaciones de volumen de memoria	-	-	1	1.06	1.21	-
Volatilidad de la maquina virtual	-	0.87	1	1.15	1.3	-
Tiempo de respuesta	-	0.87	1	1.07	1.15	1.65
Atributos del personal						
Capacidad de análisis	1.46	1.19	1	0.86	0.71	1.65
Experiencia en la aplicación	1.29	1.113	1	0.91	0.82	1.56
Capacidad de programación	1.42	1.17	1	0.86	0.7	-
Experiencia en la maquina virtual	1.21	1.1	1	0.9	-	-
Experiencia en los lenguajes	1.14	1.07	1	0.95	-	-
Atributos del proyecto						
Uso de prácticas modernas	1.24	1.1	1	0.91	0.82	-
Uso de herramientas de software	1.24	1.1	1	0.91	0.83	-
Exigencias de planificación	1.23	1.08	1	-	1.1	-
$FAE = 1,00 * 1,00 * 0,85 * 1,00 * 1,00 * 0,87 * 0,87 * 0,86 * 0,82 * 0,70 * 1,00 * 0,95 * 1,00 * 1,00 * 1,00$						
FAE=0,30171 ≈ 0,30						

FAE - JUSTIFICACIÓN

- Se deben justificar cada uno de los valores seleccionados, para cada atributo de coste.
- Ej:
 - **Fiabilidad requerida del software:** Si se produce un fallo ocasionarían pérdidas a los clientes que utilicen el software (Valoración Normal).
 - **Tamaño de la base de datos:** La base de datos del sistema será relativamente grande (Valoración Normal).
 - **Complejidad del producto:** La aplicación va a realizar cálculos sencillos (Valoración Baja).

CALCULO DE ESFUERZO

- Con todos los datos que hemos obtenido, ya podemos calcular el esfuerzo:
- **$E = \text{Esfuerzo} = a \text{ KLDC}^e * \text{FAE}$** (persona x mes)
- $E = 3.00 * 72,85^{1,12} * 0,30$
- $E = 119,85 \text{ personas/mes}$

CALCULO DE LA DURACIÓN DEL DESARROLLO

- **T = Tiempo de duración del desarrollo = c**
Esfuerzo^d (meses)
- $T = 2,5 * 119,85^{0,35}$
- $T = 13,345$
- T aproximadamente 14 meses.

CALCULO DE PERSONAL POR MES

- **$P = \text{Personal} = E/T$ (personas)**
- **$P = 119,85 / 13,345$**
- **$P = 8,98$**
- P aproximadamente 9 personas x mes, para cumplir el proyecto en 14 meses. Esto debe ajustarse según los tiempos planificados.

CALCULO DE PRODUCTIVIDAD

- $PR = LDC / \text{Esfuerzo} = 72850 / 119,85 = \mathbf{607,85}$
LDC/personas mes
- **Esto da una estimación de la cantidad de líneas de código por persona por mes a generar.**

COSTO ESTIMADO DE LA CODIFICACIÓN

- Costo estimado de la codificación = Días de codificación * Horas de trabajo * Costo hora de trabajo
- Costo estimado de codificación = $40 * 8 * 120$
- Costo estimado de codificación = \$ 38400
- Este costo solo hace referencia a la etapa de codificación, no se tienen en cuenta las demás etapas.

MODELO INTERMEDIO

- Añade al modelo básico 15 factores de ajuste o guías de coste.
- Logramos mayor precisión en la estimación gracias a los nuevos factores.
- La fórmula es la misma que la del modelo básico pero con el añadido del factor (multiplicando).

MODELO INTERMEDIO

Atributos del modelo:

- Software:
 - **RELY**: Indica las consecuencias para el usuario si falla el producto.
 - **DATA**: Relación Tamaño de la BD / Líneas de código.
 - **CPLX**: Complejidad del producto.

MODELO INTERMEDIO

Atributos del modelo:

- Hardware:
 - **TIME**: Limitaciones en el porcentaje del uso de la CPU.
 - **STOR**: Limitaciones en el porcentaje del uso de la memoria.
 - **VIRT**: Volatilidad de la máquina virtual.
 - **TURN**: Tiempo de respuesta.

MODELO INTERMEDIO

Atributos del modelo:

- Personal:
 - **ACAP**: calificación de los analistas.
 - **AEXP**: experiencia del personal.
 - **PCAP**: calificación de los programadores.
 - **VEXP**: experiencia del personal en la máquina virtual.
 - **LEXP**: experiencia en el lenguaje.

MODELO INTERMEDIO

Atributos del modelo:

- Proyecto:
 - **MODP**: uso de prácticas modernas de programación.
 - **TOOL**: uso de herramientas de desarrollo de software.
 - **SCED**: limitaciones en el cumplimiento de la planificación.

EJEMPLO ESTIMACIÓN:

- Debemos desarrollar un software de no muy elevada dificultad, con las siguientes restricciones:
 - 3 meses para el desarrollo del proyecto software.
 - Debe estar implementado en el lenguaje Visual Basic.

EJEMPLO ESTIMACIÓN:

- Calculo del esfuerzo:

Necesitamos hallar la variable KDLC.

LENGUAJE	LDC/PF
Ensamblador	320
C	150
COBOL	105
Pascal	91
Prolog/LISP	64
C++	64
Visual Basic	32
SQL	12

EJEMPLO ESTIMACIÓN:

- **KLDC** = $(PF * \text{Líneas de código por cada PF}) / 1000 = (261,36 * 32) / 1000 = \mathbf{8,363}$
- Usaremos el tipo Organico ya que nuestro proyecto no supera las 50 KLDC, y es el mas apropiado en este caso.

EJEMPLO ESTIMACIÓN:

- Coeficientes a usar:

PROYECTO SOFTWARE	a	b	c	d
Orgánico	3,2	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,20	2,5	0,32

EJEMPLO ESTIMACIÓN:

- Calculo de la variable FAE:

CONDUCTORES DE COSTE	VALORACIÓN					
	<i>Muy bajo</i>	<i>Bajo</i>	<i>Nominal</i>	<i>Alto</i>	<i>Muy alto</i>	<i>Extr. alto</i>
Fiabilidad requerida del software	0,75	0,88	1,00	1,15	1,40	-
Tamaño de la base de datos	-	0,94	1,00	1,08	1,16	-
Complejidad del producto	0,70	0,85	1,00	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	1,00	1,11	1,30	1,66
Restricciones del almacenamiento principal	-	-	1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	1,00	1,15	1,30	-
Tiempo de respuesta del ordenador	-	0,87	1,00	1,07	1,15	-
Capacidad del analista	1,46	1,19	1,00	0,86	0,71	-
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	-
Capacidad de los programadores	1,42	1,17	1,00	0,86	0,70	-
Experiencia en S.O. utilizado	1,21	1,10	1,00	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1,00	0,95	-	-
Prácticas de programación modernas	1,24	1,10	1,00	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1,00	0,91	0,83	-
Limitaciones de planificación del proyecto	1,23	1,08	1,00	1,04	1,10	-

EJEMPLO ESTIMACION

- Cálculo de la variable FAE:
- $FAE = 1,15 * 1,00 * 0,85 * 1,11 * 1,00 * 1,00 * 1,07 * 0,86 * 0,82 * 0,70 * 1,00 * 0,95 * 1,00 * 0,91 * 1,08 = 0,53508480$
- Cálculo del esfuerzo del desarrollo:
- $E = a \text{ KLDC}^{(b)} * FAE = 3,2 * (8.363)^{1,05} * 0,53508480 = 15,91 \text{ personas /mes}$

EJEMPLO ESTIMACIÓN:

- Cálculo tiempo de desarrollo:
- $T = c \text{ Esfuerzo } d = 2,5 * (15,91)^{0,38} = \mathbf{7,15 \text{ meses}}$
- Productividad:
- $PR = LDC / \text{Esfuerzo} = 8363 / 15,91 = \mathbf{525,64}$
LDC/personas mes

EJEMPLO ESTIMACION:

- Personal promedio:
- $P = E/T = 15,91/7,15 = \mathbf{2,22 \text{ personas}}$
- Segun los resultados necesitaremos un equipo de 3 personas trabajando alrededor de 7 meses, pero como una restricción era 3 meses incrementamos a 6 el numero de personas. 1 Jefe de proyecto, 2 Analistas, 2 programadores y 1 Responsable de calidad.

CONCLUSIONES...

- Medición → Esencial si se desea conseguir calidad.
- Métricas de producto: proporcionan la visión interna necesaria para crear modelos efectivos de análisis y diseño, código sólido y pruebas minuciosas.
- Medidas absolutas: no existen en software → se obtienen medidas indirectas que nos dan indicaciones de las representaciones del software.

REFERENCIAS

- **Bibliografía Obligatoria**

- [Pressman, 2005] Pressman, Roger S. "Ingeniería del Software: un Enfoque Práctico". **Capítulos 4**. 6ta. Edición. Mc Graw-Hill. 2005.
- [Sommerville, 2005] Sommerville, Ian. "Ingeniería de Software". **Capítulo 27**. Séptima Edición. Pearson Education, 2005.

- **Bibliografía Complementaria**

- [Basili, Caldiera y Rombach, 1994]. Basili, V., Caldiera, G., Rombach, H.D. The Goal Question Metric Approach. 1994.
- [SEI, 1997]. Florac, W.A.; Park, R.E.; Carleton, A.D. Practical Software Measurement: Measuring for Process Management and Improvement. April, 1997.
- [ISO, 2007] Std. ISO/IEC 15939-2007, Software Engineering - Software Measurement Process.
- [Franzoy y Vrancken, 2010]. Franzoy, M. y Vrancken, L. Sistema de Medición y Análisis. Proyecto Final de Carrera, Ingeniería en Sistemas de Información. FRSF – UTN. Octubre, 2010.
- [Wieggers, 2007]. Wieggers, K.E. Practical Project Initiation: A Handbook with Tools. Microsoft Press, 2007.

¿Dudas, consultas?

