Ingeniería de software II

Métricas

Otras actividades a realizar¿?

Planificacion

Gestión de Proyectos

Métricas

Elementos clave de la gestión de proyectos

- Métricas
- Estimaciones
- Calendario temporal
- Organización del personal
- Análisis de riesgos
- Seguimiento y control

Métricas y Estimaciones

4

Métricas

»Las métricas son la clave tecnológica para el desarrollo y mantenimiento exitoso del software. (Briand et al., 1996)

»En general, la medición persigue los siguientes objetivos fundamentales (Fenton y Pfleeger, 1997):

Entender qué ocurre durante el desarrollo y el mantenimiento

Controlar qué es lo que ocurre en nuestros proyectos

Mejorar nuestros procesos y nuestros productos

Evaluar la calidad.

Métricas – Definiciones

»Medida:

indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto.

»Medición:

es el acto de determinar una medida.

»Pressman Cap. 4

Métricas – Definiciones

»Métrica:

medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. El ingeniero de software recopila medidas y desarrolla métricas para obtener indicadores.

LOC por punto de función			
Lenguaje	LOC/FP	Lenguaje	LOC/FP
Ensamblador	320	Basic ANSI/Quick/Turbo	64
Macroensamblador	213	Java	53
C	150	Visual C++	34
Fortran	106	Foxpro 2,5	34
Cobol	106	Visual Basic	32
Pascal	91	Delphi	29
Cobol ANSI 85	91	C++	29
Basic	91	Visual Cobol	20
RPG	80	Clipper	19
PL/I	80	Power Builder	16
Ada	71	Hoja de Calculo	6

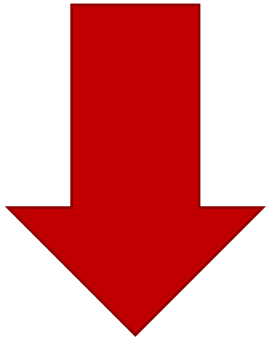
»Indicador:

combinación de métricas. Proporciona una visión profunda que permite al gestor de proyectos o a los ingenieros de software ajustar el producto, el proyecto o el proceso para que las cosas salgan mejor

»M Pressman Cap. 4

Métricas

»Las métricas pueden ser utilizadas para que los profesionales e investigadores puedan tomar las mejores decisiones



Métricas como medio para asegurar la calidad en los Productos/Procesos/ Proyectos Software

Métricas

- Existen dos formas en que pueden usarse las mediciones de un sistema de software:
  - Para **asignar un valor** a los atributos de calidad del software.
  - Para **identificar los componentes** del sistema cuya calidad está por debajo de un estándar.

Clasificación de las métricas

- Métricas **de control**

  Apoyan la gestión del proceso.

  Ej: esfuerzo promedio, tiempo requerido para reparar defectos

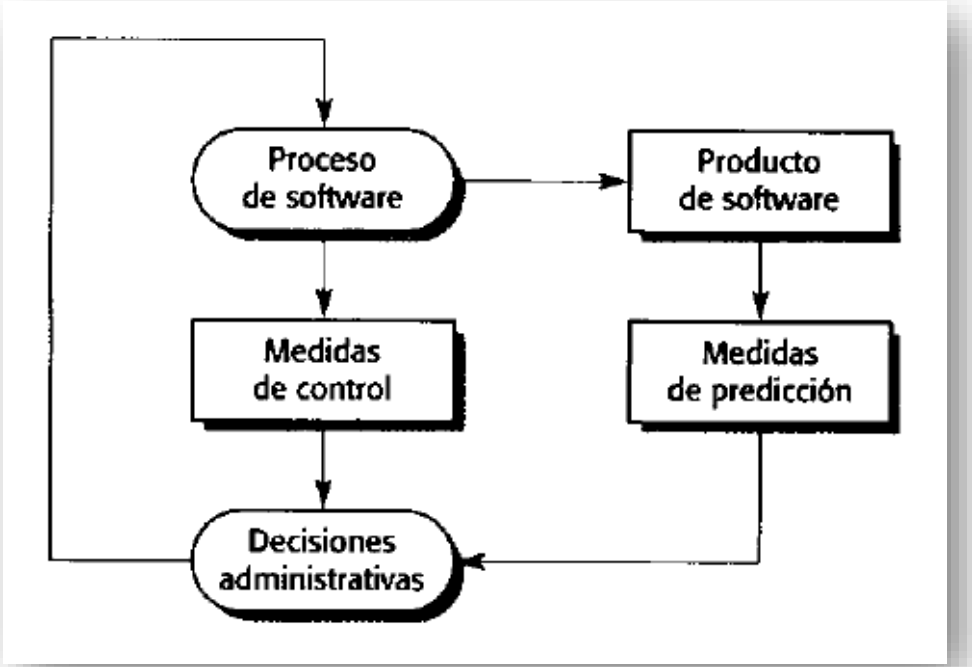
- Métricas de **predicción** (del producto)

  Ayudan a predecir las características del software. Se conocen también como métricas del producto.

  Por ej.: Tamaño, complejidad

Clasificación de las métricas

»Tanto las métricas de control como las de predicción influyen en la toma de decisiones.



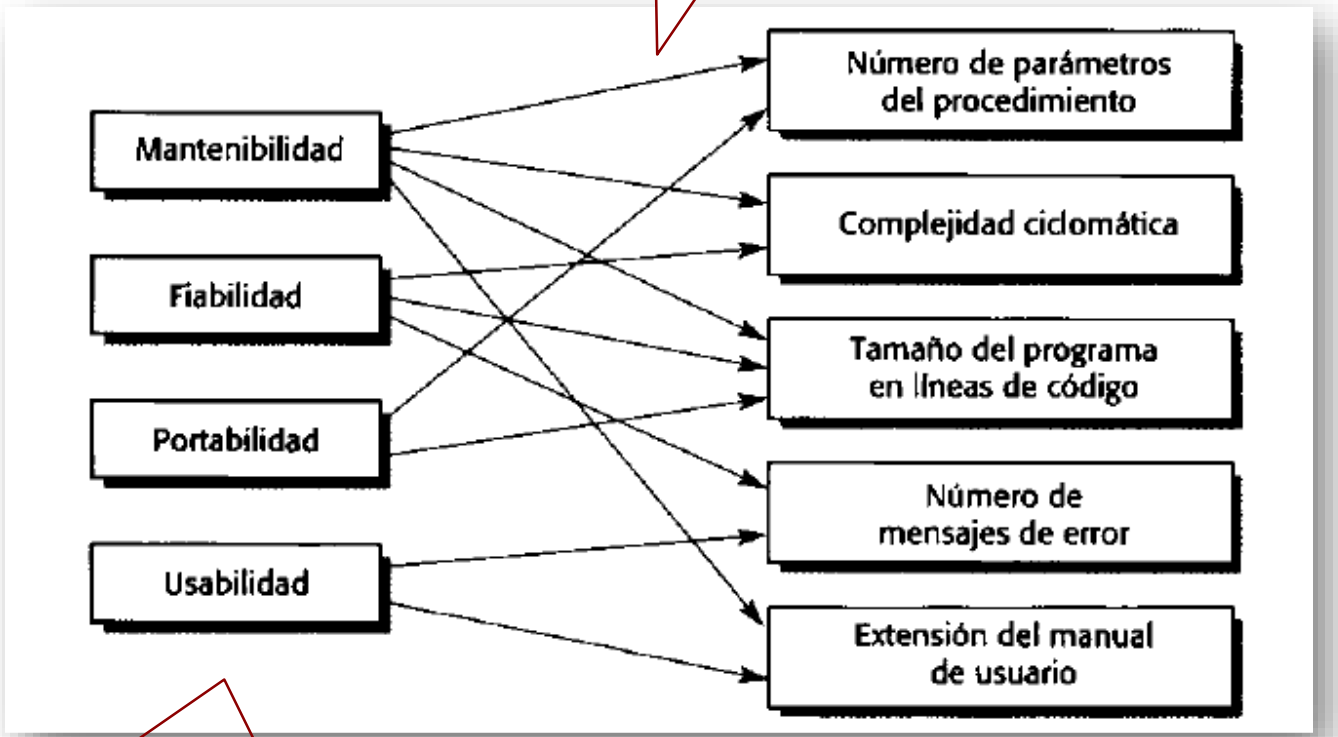
Métricas

- Es difícil hacer mediciones directas de muchos de los atributos de calidad.
- Los atributos externos se ven afectados por factores subjetivos como la experiencia, educación del usuario. Para evaluarlos hay que medir algunos atributos internos del software y relacionarlos.

Métricas

La relación entre atributos internos y externos debe comprenderse, validarse y expresarse en términos de una formula o modelo.

Atributos de calidad externos Atributos internos



Atributos que se refieren a cómo los desarrolladores y usuarios experimentan.

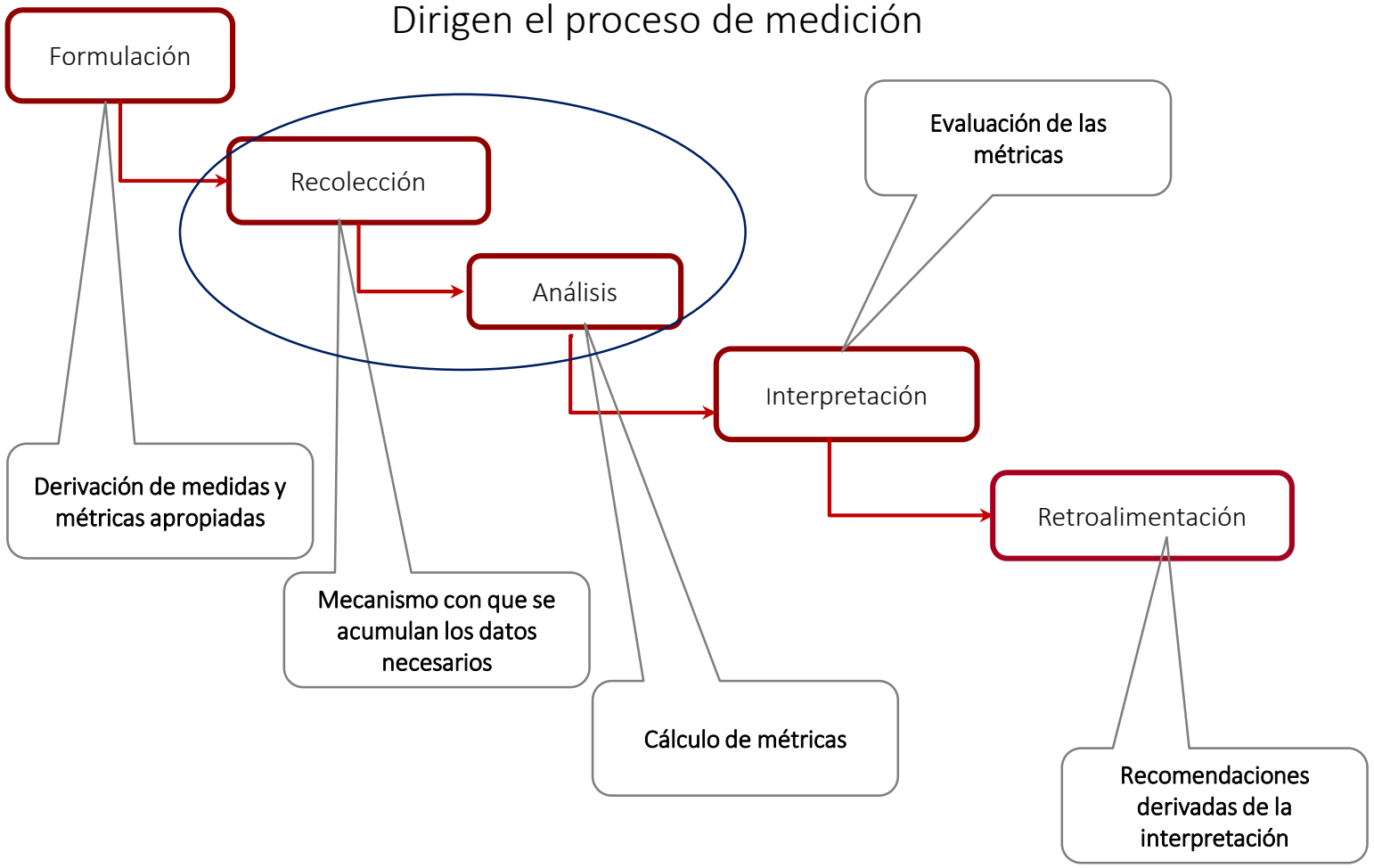
Métricas del producto

- Las métricas del producto son métricas de predicción para medir los atributos internos de un sistema de software
- Se dividen en dos clases:
  - **Métricas dinámicas**, que se recopilan de un programa en ejecución. Ej. Nro. de reportes de bugs.
  - **Métricas estáticas**, que se recopilan mediante mediciones hechas de representaciones del sistema. Ej: tamaño del código.

Métricas del producto

- Las **métricas dinámicas** ayudan a valorar la eficiencia y fiabilidad de un programa.
- Las **métricas estáticas**, ayudan a valorar la complejidad, comprensibilidad y mantenibilidad de un sistema de software o sus componentes.

Proceso de medición



Métricas estáticas del producto

Fan-in/Fan-out	Fan-in es una medida del número de funciones o métodos que llaman a otra función o método (por ejemplo, X). Fan-out es el número de funciones que son llamadas por una función X. Un valor alto de fan significa que X está fuertemente acoplada al resto del diseño y que los cambios en X tendrán muchos efectos importantes. Un valor alto de fan-out sugiere que la complejidad de X podría ser alta debido a la complejidad de la lógica de control necesaria para coordinar los componentes llamados.
Longitud del código	Ésta es una medida del tamaño del programa. Generalmente, cuanto más grande sea el tamaño del código de un componente, más complejo y susceptible de errores será el componente. La longitud del código ha mostrado ser la métrica más fiable para predecir errores en los componentes.
Complejidad ciclomática	Ésta es una medida de la complejidad del control de un programa. Esta complejidad del control está relacionada con la comprensión del programa.
Longitud de los identificadores	Es una medida de la longitud promedio de los diferentes identificadores en un programa. Cuanto más grande sea la longitud de los identificadores, más probable será que tengan significado; por lo tanto, el programa será más comprensible.
Profundidad del anidamiento de las condicionales	Ésta es una medida de la profundidad de anidamiento de las instrucciones condicionales «if» en un programa. Muchas condiciones anidadas son difíciles de comprender y son potencialmente susceptibles de errores.
Índice de Fog	Ésta es una medida de la longitud promedio de las palabras y las frases en los documentos. Cuanto más grande sea el índice de Fog, el documento será más difícil de comprender.

Métricas del producto

»Las métricas sólo serán útiles si están caracterizadas de manera efectiva y se validan para probar su valor.

Métricas del producto

Principios que se pueden usar para caracterizar y validar las métricas

- Una métrica **debe tener propiedades matemáticas deseables** (rango significativo)
- Cuando una métrica **representa una característica de software** que aumenta cuando se presentan rasgos positivos o que disminuye al encontrar rasgos indeseables, el valor de la métrica debe aumentar o disminuir en el mismo sentido.
- Cada métrica **debe validarse empíricamente** en una amplia variedad de contextos antes de publicarse o aplicarse en la toma de decisiones.

Métricas del producto

La métrica más común para el tamaño de un producto es el número de líneas de código.

»LDC - LÍNEAS DE CÓDIGO

Medida directa del software y del proceso

Medida discutida porque depende del lenguaje y es post-mortem

- Es para saber en qué tiempo voy a terminar el software y cuántas personas voy a necesitar.
- Si una organización de software mantiene registros sencillos, se puede crear una tabla de datos orientados al tamaño

Utilidad de las métricas postmortem

- Conformar una línea base para futuras métricas
- Ayudar al mantenimiento conociendo la complejidad lógica, tamaño, flujo de información, identificando módulos críticos
- Ayudar en los procesos de reingeniería

Métricas orientadas al tamaño

»Métricas derivadas del proceso de desarrollo:

KLDC (miles de líneas de código)

Productividad: relación entre KLDC / Persona mes
Calidad: relación entre Errores / KLDC
Costo: relación entre $ / KLDC

Métricas orientadas al tamaño

- LDC - LÍNEAS DE CÓDIGO
- Exigen explicar el manejo de:

  líneas en blanco, líneas de comentarios , declaraciones de datos, líneas con varias instrucciones separadas

  Sino…

→ **Información que se pierde:** espacio que ocupa en disco, páginas que requiere el listado.

Propuesta Fenton/Pfleeger

- Medir :
  - CLOC = **Cantidad de líneas de comentarios**
- Luego:
  - long total (LOC) = NCLOC + CLOC
- Surgen medidas indirectas:
  - CLOC/LOC mide la densidad de comentarios

Ejemplo

Productividad = KLDC/persona-mes
Calidad = errores/KLDC
Documentación = págs.. Doc./ KLDC
Costo = $/KLDC

»Calcular, usando **LDC** , la productividad, calidad y costo para los cuatro proyectos de los cuales se proporcionan los datos.

Proyecto	LDC	U$S	Errores	Personas-mes	Errores/KLDC	U$S/KLDC	KLDC/Personas-mes
P1	25.500	15000	567	15	22,23 %	588,23	1,7
P2	19.100	7200	210	10	10,99 %	376,96	1,91
P3	10.700	6000	100	20	9,34 %	560,74	0,53
P4	100.000	18000	2200	30	22 %	180	3,33

»¿Cuál es el proyecto de **mayor calidad** (errores/KLDC)?

»¿Cuál es el proyecto de **mayor costo por línea** ($/KLDC)?

»¿Cuál es el proyecto de **menor productividad por persona** (KLDC/personas-mes)?

Métricas de control

»Un ejemplo de métrica de control es la llamada ***Métrica de Punto Función (FP),*** que examina el modelo de análisis con la intención de predecir el tamaño del sistema.

Mide la cantidad de funcionalidad de un sistema descripto en una especificación

Métrica de Punto función

PF- Punto función (Albrecht 1978)

Factor de Ponderación, es subjetivo y esta dado por la organización/equipo

PF = TOTAL * [0.65 + 0.01 * SUM(Fi)] i=1 a 14 0 <= Fi<=5

	simple | medio | complejo	
# Entradas	* [3 | 4 | 6]	= .....
# Salidas	* [4 | 5 | 7]	= .....
# Consultas	* [3 | 4 | 6]	= .....
# Almacenamientos internos	* [7 | 10 | 15]	= .....
# Interfaces externas	* [5 | 7 | 10]	= .....

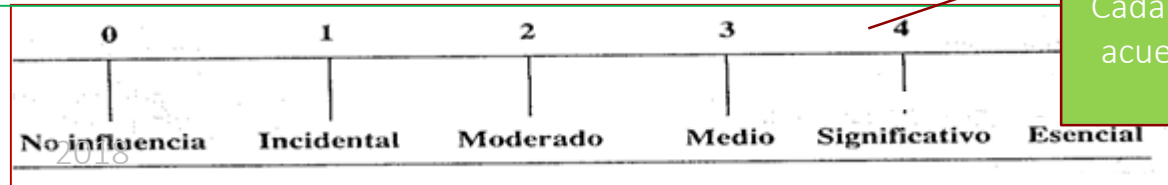
TOTAL

Son valores de ajuste de la complejidad según las preguntas de la siguiente pantalla

Medida subjetiva independiente del lenguaje, de estimación más fácil. Métrica temprana

Métrica de Punto función

1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?
2.¿Se requiere comunicación de datos?
3. Existen funciones de procesamiento distribuido?
4. ¿Es crítico el rendimiento?
5. ¿Se ejecuta el sistema en un entorno operativo existente y fuertemente utilizado?
6. ¿Requiere el sistema 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?
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?



Cada una de las preguntas se contesta de acuerdo a la siguiente escala de valores

Métrica de Punto función

- Métricas derivadas:
  - **Productividad:** relación entre PF y Persona_mes
  - **Calidad**: relación entre Errores y PF
  - **Costo**: relación entre $ y PF

Productividad = PF / Persona_mes
Calidad = Errores / PF
Costo = $ / PF

- Pressman Cap. 15

Desarrollo de una métrica

- Victor Basili desarrolló un método llamado **GQM** (Goal, Question, Metric) (o en castellano: OPM Objetivo, Pregunta, Métrica).

- (ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf)

- Dicho método esta orientado a lograr una métrica que “mida” cierto objetivo. El mismo nos permite mejorar la calidad de nuestro proyecto.

GQM (OPM)

»Estructura :

Nivel **Conceptual** (Goal / Objetivo).

Se define un objetivo (en nuestro caso, para el proyecto).

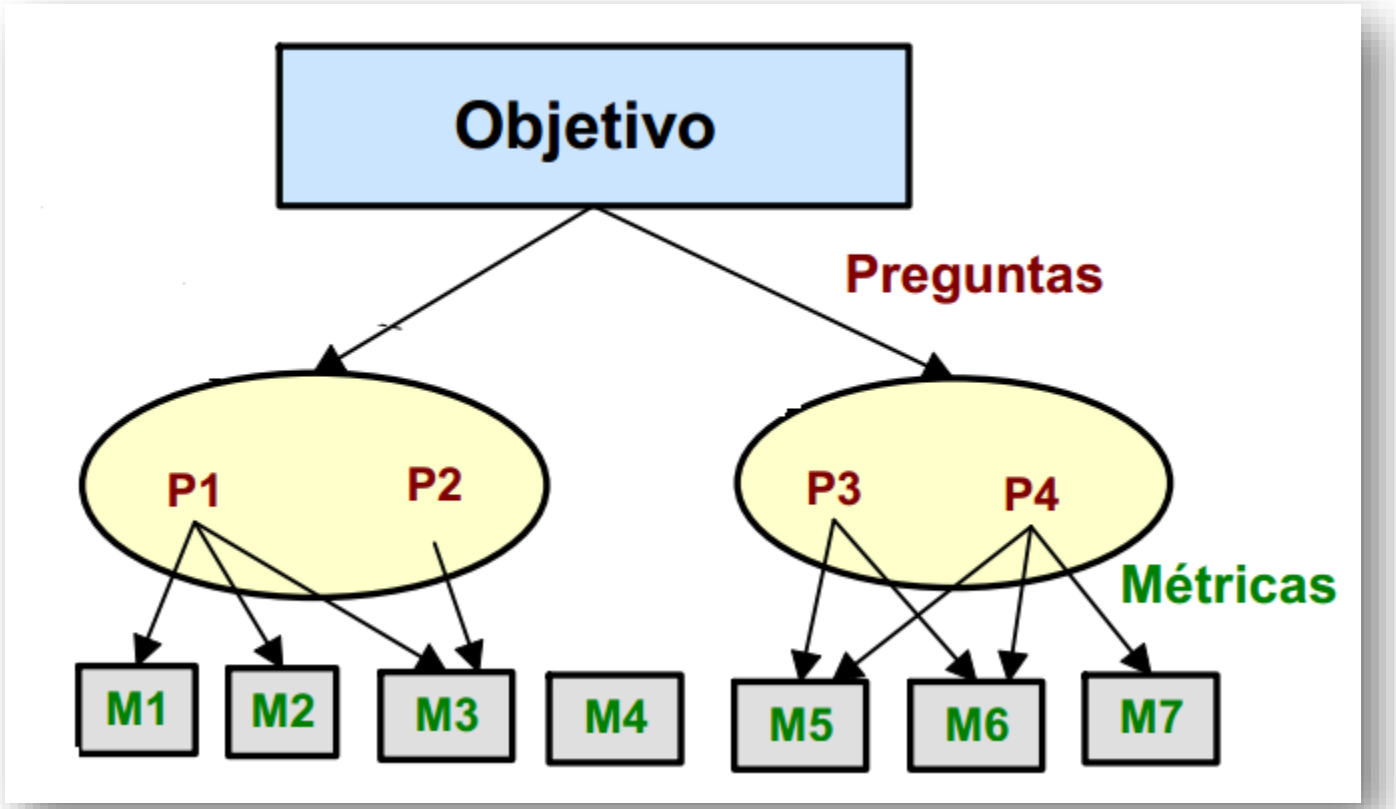
Nivel **Operativo** (Question / Pregunta).

Se refina un conjunto de preguntas a partir del objetivo, con el propósito de verificar su cumplimiento.

Nivel **Cuantitativo** (Metric / Métrica).

Se asocia un conjunto de métricas para cada pregunta, de modo de responder a cada una de un modo cuantitativo.

GQM (OPM)



GQM ejemplo

- Evaluamos, en la etapa de Análisis de Requerimientos, la tarea Asignación de responsabilidades (es sólo un ejemplo, se puede tomar la actividad o tarea que se crea prioritaria).

GQM Ejemplo

Propósito	**Evaluar**	
Característica	**Asignación de responsabilidades**	
Punto de Vista	**Gerencia de Proyecto**	
Pregunta 1	¿Existe un proceso para la asignación de roles?	
	M1	Valor Binario
Pregunta 2	¿Hay un responsable de asignar roles?	
	M2	Valor Binario
Pregunta 3	¿El responsable siempre realiza su tarea?	
	M3	Valor Binario
Pregunta 4	¿Existe información anterior sobre las tareas realizadas por cada integrante?	
	M4	Valor Binario
Pregunta 5	¿Esa información esta disponible?	
	M5	Valor Binario

Indicadores

Nombre	Descripción	Fórmula
I1	Gestión de Asignación de roles	M2 & M3 & M4 & M5
I2	Proceso de Asignación de roles	M1 & M4 & M5

A partir de los indicadores definidos, se propone realizar el control de la meta a través de un tablero de control de indicadores específicos. Podemos decir que nuestra meta se cumple si los indicadores muestran los siguientes valores:

I1	Gestión de Asignación de roles	Verdadero
I2	Proceso de Asignación de roles	Verdadero

GQM (OPM)

- Es útil para decidir qué medir.
- Debe estar orientado a metas.
- Es flexible.

Estimaciones

37

Estimaciones

- Estimación

  Técnica que permiten dar un valor aproximado.

- Para obtener estimaciones confiables generalmente se usan varias técnicas y se comparan y concilian resultados.
- La estimación no es una ciencia exacta.
- Modificaciones en la especificación hacen peligrar las estimaciones.
- Requieren experiencia, acceso a información histórica y decisión para convertir información cualitativa en cuantitativa.

Estimaciones

El riesgo de la estimación decrece con la disponibilidad de historia

- Se realizan estimaciones de recursos, costos y tiempos

- Los factores que influyen son la complejidad, el tamaño, la estructuración del proyecto.

Estimaciones de costos

Existen tres principales parámetros que se deben usar al calcular los costos de un proyecto.

- **Costos de esfuerzo** (pagar desarrolladores e ingenieros)
- **Costos de hardware y software**, incluido el mantenimiento
- **Costos de viaje**

Para la mayoría de los proyectos, el mayor costo es el primer rubro.

Estimaciones de costos

- Debe estimarse el esfuerzo total (meses-hombre), sin embargo se cuenta con datos limitados para esta valoración.
- Es posible que se deba licenciar el middleware y la plataforma, o que se requieran mayor cantidad de viajes cuando se desarrolla en distintos lugares.
- Se debe iniciar con un bosquejo de Plan de Proyecto y se debe contar con una especificación de los requerimientos.

Fijación de precio

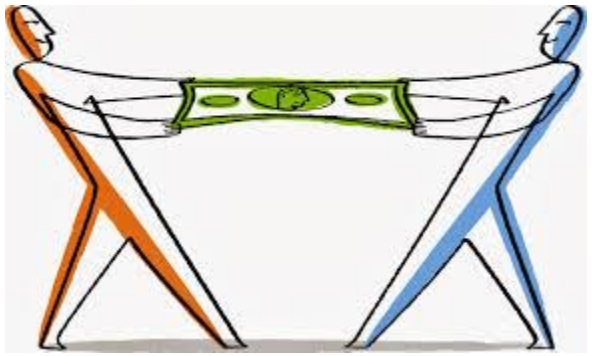
»En principio el precio es simplemente el costo de desarrollo, sin embargo en la práctica , la relación entre el costo y el precio al cliente no es tan simple.

»Cuando se calcula un precio hay que considerar temas de índole organizacional, económica, política y empresarial.

Oportunidad de mercado	Una organización de desarrollo podría ofertar un bajo precio debido a que desea conseguir cuota de mercado. Aceptar un beneficio bajo en un proyecto podría darle la oportunidad de obtener más beneficios posteriormente. La experiencia obtenida le permite desarrollar nuevos productos.
Incertidumbre en la estimación de costes	Si una organización está insegura de su coste estimado, puede incrementar su precio por encima del beneficio normal para cubrir alguna contingencia.
Términos contractuales	Un cliente puede estar dispuesto a permitir que el desarrollador retenga la propiedad del código fuente y que reutilice el código en otros proyectos. Por lo tanto, el precio podría ser menor que si el código fuente del software se le entregara al cliente.
Volatilidad de los requerimientos	Si es probable que los requerimientos cambien, una organización puede reducir los precios para ganar un contrato. Después de que el contrato se le asigne, se cargan precios altos a los cambios en los requerimientos.
Salud financiera	Los desarrolladores en dificultades financieras podrían bajar sus precios para obtener un contrato. Es mejor tener beneficios más bajos de los normales o incluso quebrar antes de quedar fuera de los negocios.

Fijación de precio

»Debe pensarse en los intereses de la empresa, los riesgos y el tipo de contrato. Esto puede hacer que el precio suba o baje.



Estimaciones de recursos

- Recursos humanos
- Recursos de software reutilizables
- Recursos de hardware y herramientas de software

- Cada recurso requiere:
  - Descripción
  - Informe de disponibilidad
  - Fecha en que se lo requiere
  - Tiempo que se lo necesita

Técnicas de estimación

- Juicio experto:

  Se consultan varios expertos. Cada uno de ellos estima. Se comparan y discuten

- Técnica Delphi:

  Consiste en la selección de un grupo de expertos a los que se les pregunta su opinión. Las estimaciones de los expertos se realizan en sucesivas rondas, anónimas, con el objeto de tratar de conseguir consenso, pero con la máxima autonomía por parte de los participantes.

- División de trabajo:

  Jerárquica hacia arriba

Modelos empíricos de estimación

»Utilizan fórmulas derivadas empíricamente para predecir costos o esfuerzo requerido en el desarrollo del proyecto.

Ej: **MODELO COCOMO** de Boehm (1981)

(COnstructive COst MOdel, modelo constructivo de costos) se obtuvo recopilando datos de varios proyectos grandes.

»Las formulas que utiliza el COCOMO vinculan el tamaño del sistema y del producto, factores del proyecto y del equipo con el esfuerzo necesario para desarrollar el sistema.

Fuente: Somerville Cap. 26

Estimaciones COCOMO 81

El modelo inicial consideraba tres tipos de proyectos**:**

Orgánicos: Proyectos pequeños y de poca gente
Semiacoplados: Proyectos intermedios
Empotrados: Proyectos con restricciones rígidas

Modelo COCOMO

El tipo BASICO estima a través de LDC

Tipos de proyectos: Orgánicos - Semiacoplados - Empotrados

a	2.4	3.0	3.6
b	1.05	1.12	1.20

$E = a\ (KLDC)^b$ **Esfuerzo en persona-mes**

Fórmula básica de nivel 1 de estimación, había tres niveles y profundizaban el detalle de la estimación.

También suponía un modelo de proceso en cascada con uso de lenguajes imperativos del estilo “C” o fortran

Estimaciones COCOMO 81

Ejemplo: Suponer que se cuenta con un Proyecto con: LDC : 19.100 y Semiacoplado

E = 3* (19,1)1,12 =81,63 esfuerzo en personas/mes

D = 2,5 * E 0,35 =11,67 meses

Modelo COCOMO

Tipos de proyectos: Orgánicos - Semiacoplados - Empotrados

c	2.5	2.5	2.5
d	0.38	0.35	0.32

$D = cE^d$ **Duración en meses**

Formula de nivel 1 para el calculo de duración en meses

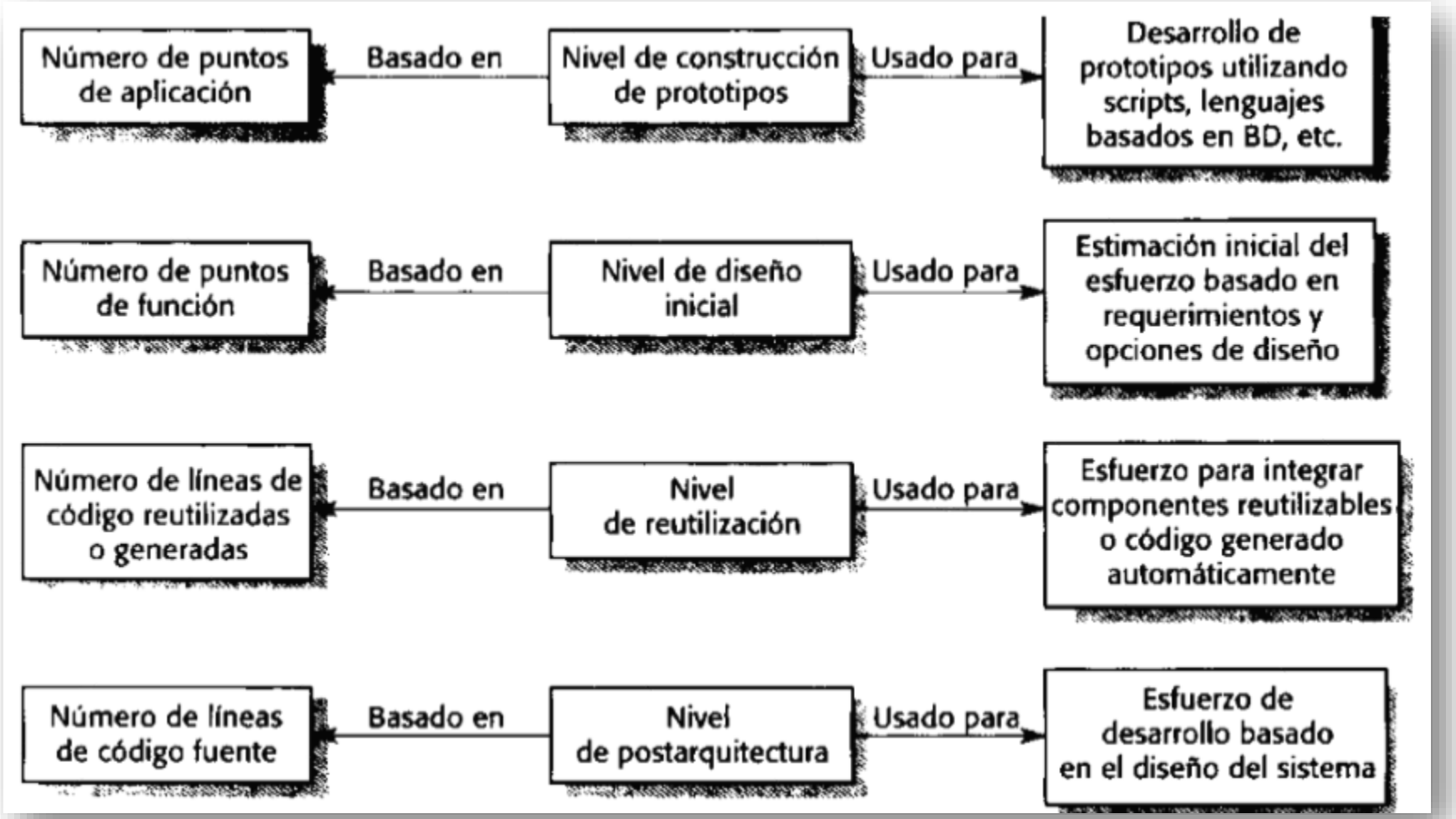
El esfuerzo se expresa en personas/**mes** (PM) y representa las personas en un mes fulltime, requeridos para desarrollar el proyecto

COCOMO II (2000)

- Reconoce que las líneas de código son difíciles de estimar tempranamente. Considera diferentes enfoques para el desarrollo , como la construcción de prototipos, el desarrollo basado en componentes, desarrollo en espiral y engloba varios niveles que producen estimaciones detalladas de forma incremental.
- COCOMO II está compuesto por 4 niveles:
  - De construcción de prototipos
  - De diseño inicial
  - De reutilización
  - De post-arquitectura

Fuente: Somerville Cap. 26

COCOMO II
Niveles de COCOMO II



COCOMO II
1. Nivel Construcción de prototipos

Punto de aplicación = Punto objeto

»La fórmula para el cálculo del esfuerzo para el prot[illegible]ado del sistema es:

»PM = (NAP x (1-%reutilización/100))/P[illegible]D

»PM = esfuerzo estimado en perso

»NAP = total de puntos de aplic

»PROD = productividad medida en

1. El número de pantallas independientes que se despliegan. Las pantallas sencillas cuentan como 1 punto objeto, las pantallas moderadamente complejas cuentan como 2 y las pantallas muy complejas cuentan como 3 puntos objeto.
2. El número de informes que se producen. Los informes simples cuentan como 2 puntos objeto, los informes moderadamente complejos cuentan como 5, y los informes que son difíciles de producir cuentan como 8 puntos objeto.
3. El número de módulos en lenguajes imperativos como Java o C++ que deben ser desarrollados para complementar el código de programación de la base de datos se contabilizará como 10 puntos objeto.

Fuente: Somerville Cap. 26

COCOMO II
1. Nivel de construcción de prototipos

»Productividad de puntos de objeto, se basa en la siguiente tabla para obtener el PROD de la formula anterior

Experiencia y capacidad de los desarrolladores	Muy baja	Baja	Media	Alta	Muy alta
Madurez y capacidad de las herramientas CASE	Muy baja	Baja	Media	Alta	Muy alta
PROD (NOP/mes)	4	7	13	25	50

Fuente: Somerville Cap. 26

COCOMO II
2. Nivel de diseño inicial

»La fórmula para las estimaciones en este nivel es:

»**Esfuerzo = A x Tamaño^{B} x M**

»Boehm propone que **A** = 2.94. (Otros autores proponen: 2.45)

»**Tamaño** = KLDC (miles de líneas de código fuente).

»**B** = 0.91 X Σ SFj (ver tabla)

»M = PERS x RCPX x RUSE x PDIF x PREX x FCIL x SCED (ver mas adelante)

Fuente: Somerville Cap. 26

COCOMO II - 2. Nivel de diseño inicial

Tabla de Valores de escala *SF para calcular B*

	Muy bajo	bajo	nominal	alto	Muy alto	Extra alto
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

PREC: experiencia de proyectos precedentes. Desde totalmente sin precedentes hasta totalmente familiar
FLEX: flexibilidad de desarrollo. Desde requerimientos muy rígidos hasta solamente metas generales
RESL: Nivel de riesgos y tiempo dedicado a arquitectura, desde casi nada a total
TEAM: cohesión del equipo. Desde dificultades graves en interacción hasta interacción suave**.**
PMAT: Madurez de acuerdo a CMM: Desde 1 hasta 5 (uno se repite)

Fuente: Somerville Cap. 26

COCOMO II – 2. Nivel de diseño inicial- *Aclaración siglas de M*

M: Multiplicador.

Son siete características/atributos del proyecto y del proceso que influyen en la estimación. Éstas hacen que aumente o disminuya el esfuerzo requerido.

- **RCPX** = Fiabilidad y complejidad del producto.
- **RUSE** = Reutilización requerida.
- **PDIF** = Dificultad de la plataforma.
- **PERS** = Capacidad del personal.
- **PREX** = Experiencia del personal.
- **SCED** = Calendario.
- **FCIL** = Facilidades de apoyo.

Se pueden estimar directamente en una escala de 1 (valor muy bajo) a 6 (valor muy alto).

Fuente: Somerville Cap. 26

COCOMO II
3. Nivel de reutilización

Para el código generado automáticamente, el modelo estima el número de persona/mes necesarias para integrar este código.

$$PMauto = (ASLOC \times AT/100)/ATPROD.$$

AT = porcentaje de código adaptado que se genera automáticamente.

ATPROD = productividad de los ingenieros que integran el código

ASLOC = Nro de líneas de código en los componentes que deben ser adaptadas

Fuente: Somerville Cap. 26

COCOMO II
4. Nivel de post-arquitectura

Las estimaciones están basadas en la misma fórmula básica

$$PM = A \times Tamaño^{B} \times M$$

pero se utiliza un conjunto más extenso de atributos (17 en lugar de 7) de producto, proceso y organización para refinar el cálculo del esfuerzo inicial.

COCOMO II
4. Nivel de post-arquitectura

»La estimación del número de líneas de código se calcula utilizando tres componentes:

- ❑ Una estimación del número total de **líneas nuevas** de código a desarrollar.
- ❑ Una estimación del número de líneas de código fuente **equivalentes** (ESLOC) calculadas usando el nivel de reutilización.
- ❑ Una estimación del número de líneas de código que **tienen que modificarse** debido a cambios en los requerimientos.

»Estas estimaciones se añaden para obtener el tamaño del código (KLDC).

Fuente: Somerville Cap. 26

COCOMO II
4. Nivel de post-arquitectura

- Estas estimaciones se añaden para obtener el tamaño del código (KLDC).
- El exponente **B** se calcula considerando 5 factores de escala.
- Como
  - Productividad de desarrollo
  - Cohesión del equipo
  - Entre otros

Fuente: Somerville Cap. 26

Estimaciones on-line

»Visitar las páginas:

»Para COCOMO81:

»http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/cocomo81.html

»Para COCOMO 2:

»http://csse.usc.edu/tools/COCOMOII.php