



Gestión de Proyectos

Métricas

Elementos clave de la gestión de proyectos

- » Métricas
- » Estimaciones

} Tema de la clase de hoy

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

Métricas

Clave tecnológica

Objetivos fundamentales:

- ❖ Entender
- ❖ Controlar
- ❖ Mejorar
- ❖ Evaluar



Métricas – Definiciones

Medida



Medición



Métrica

LOC por punto de función			
Ensayos	LOC-EP	Ensayos	LOC-EP
Examinador	320	Base Análisis de Tareas	44
Macros/Programador	213	Java	53
C	185	Visual C++	28
Fortran	108	Prolog 3.8	34
Cobol	108	Visual Basic	53
Pascal	84	Delphi	25
Visual Basic 6.0	84	C++	23
Basic	84	Visual C++ 6.0	22
PHP	30	C#	63
PLI	85	Power Builder	18
Ada	71	Modulo Calculator	8

Indicador

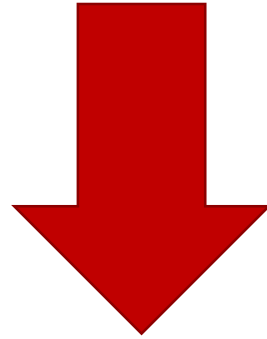


Métrica -definición

- » A diferencia de una simple medida, una métrica a menudo implica un cálculo o una fórmula que relaciona varias medidas para proporcionar una comprensión más profunda

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
Procesos/Proyectos Software/Productos

Métricas del proyecto

Propósitos tácticos

Uso

- ❖ Ajustes en el calendario y evitar demoras
- ❖ Valorar el estado de un proyecto en marcha
- ❖ Rastrear riesgos
- ❖ Descubrir áreas de problemas
- ❖ Ajustar flujo de trabajo/tareas
- ❖ Evaluar habilidad del equipo.



Métricas del proceso

» Propósitos estratégicos



Recopilación

a través de todos los proyectos y por un espacio de tiempo.



Intención

proporcionar un conjunto de indicadores para mejorar el proceso.

Métricas del proceso

Uso

- ❖ Sentido común y sensibilidad organizacional.
- ❖ Retroalimentación.
- ❖ No usar métricas para valorar a los individuos.
- ❖ Establecer metas y métricas claras.
- ❖ No excluir métricas.



Métricas del producto

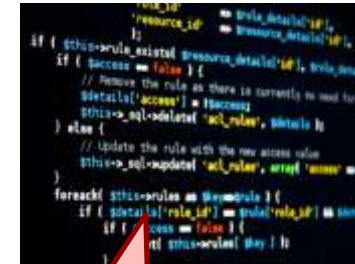
Cómo los medimos



Dinámicas

- ❖ Hecho sobre un programa en ejecución.
- ❖ Ayudan a valorar la eficiencia y fiabilidad de un programa.

Se relacionan con las características de calidad del software



Estáticas

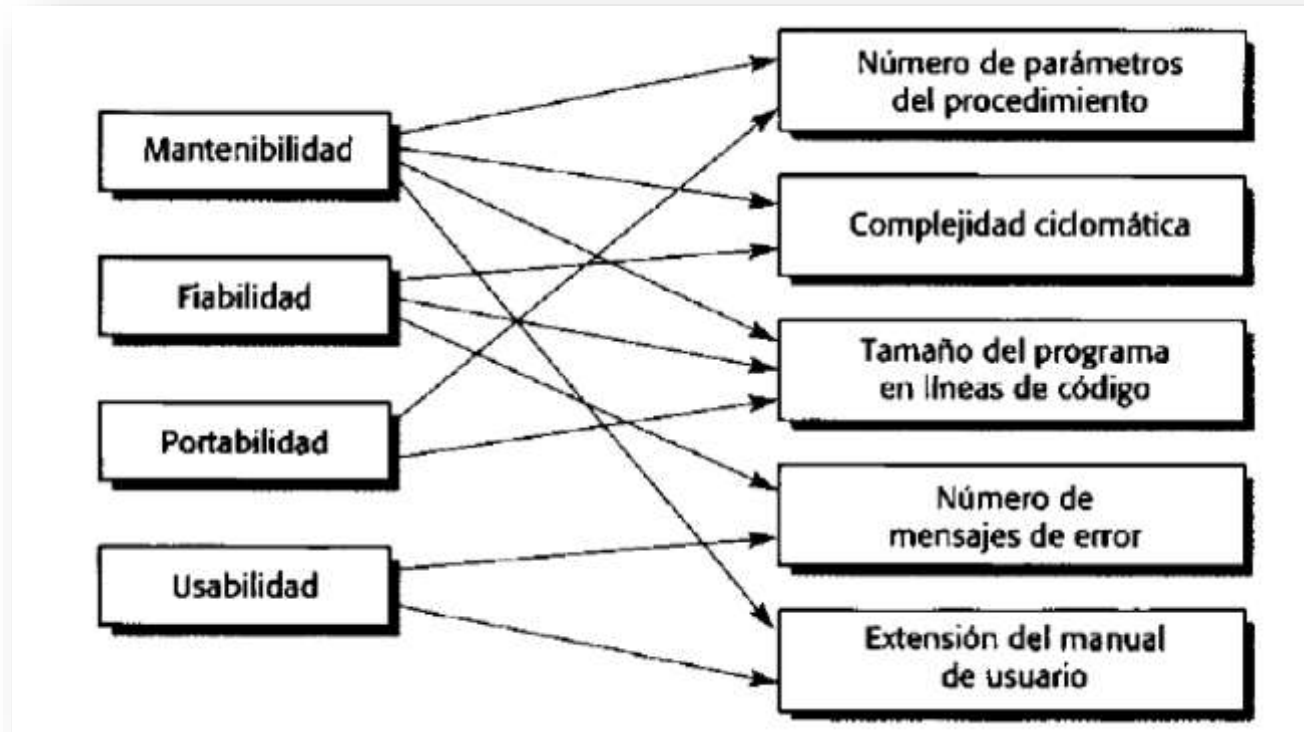
- ❖ Hecho sobre representaciones del sistema.

Se relacionan de manera indirecta con los atributos de calidad del software

d.

Métricas del producto

Atributos de calidad externos vs Atributos internos



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. El cálculo de la complejidad ciclomática se trata en el Capítulo 22.
Longitud de los identificadores	Ésta es una medida del promedio de los diferentes identificadores en un programa. Cuando 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. Cuando más grande sea el índice de Fog, el documento será más difícil de comprender.

Métricas estáticas del producto

Métricas OO

Métrica orientada a objetos	Descripción
Métodos ponderados por clase (<i>weighted methods per class, WMC</i>)	Este es el número de métodos en cada clase, ponderado por la complejidad de cada método. Por lo tanto, un método simple puede tener una complejidad de 1, y un método grande y complejo tendrá un valor mucho mayor. Cuanto más grande sea el valor para esta métrica, más compleja será la clase de objeto. Es más probable que los objetos complejos sean más difíciles de entender. Tal vez no sean lógicamente cohesivos, por lo que no pueden reutilizarse de manera efectiva como superclases en un árbol de herencia.
Profundidad de árbol de herencia (<i>depth of inheritance tree, DIT</i>)	Esto representa el número de niveles discretos en el árbol de herencia en que las subclases heredan atributos y operaciones (métodos) de las superclases. Cuanto más profundo sea el árbol de herencia, más complejo será el diseño. Es posible que tengan que comprenderse muchas clases de objetos para entender las clases de objetos en las hojas del árbol.
Número de hijos (<i>number of children, NOC</i>)	Esta es una medida del número de subclases inmediatas en una clase. Mide la amplitud de una jerarquía de clase, mientras que DIT mide su profundidad. Un valor alto de NOC puede indicar mayor reutilización. Podría significar que debe realizarse más esfuerzo para validar las clases base, debido al número de subclases que dependen de ellas.
Acoplamiento entre clases de objetos (<i>coupling between object classes, CBO</i>)	Las clases están acopladas cuando los métodos en una clase usan los métodos o variables de instancia definidos en una clase diferente. CBO es una medida de cuánto acoplamiento existe. Un valor alto para CBO significa que las clases son estrechamente dependientes y, por lo tanto, es más probable que el hecho de cambiar una clase afecte a otras clases en el programa.
Respuesta por clase (<i>response for a class, RFC</i>)	RFC es una medida del número de métodos que potencialmente podrían ejecutarse en respuesta a un mensaje recibido por un objeto de dicha clase. Nuevamente, RFC se relaciona con la complejidad. Cuanto más alto sea el valor para RFC, más compleja será una clase y, por ende, es más probable que incluya errores.
Falta de cohesión en métodos (<i>lack of cohesion in methods, LCOM</i>)	LCOM se calcula al considerar pares de métodos en una clase. LCOM es la diferencia entre el número de pares de método sin compartir atributos y el número de pares de método con atributos compartidos. El valor de esta métrica se debate ampliamente y existe en muchas variaciones. No es claro si realmente agrega alguna información útil además de la proporcionada por otras métricas.



Métricas del producto - LDC – LÍNEAS DE CÓDIGO



La métrica más común para el tamaño de un producto

postmorten



Medida discutida !!

Resultados



¿Qué tiempo?



¿Cuántas personas?



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

❖ Se puede obtener :

Productividad: relación entre KLDC / Persona mes

Calidad: relación entre Errores / KLDC

Costo: relación entre \$ / KLDC



KLDC (miles de líneas de código)
LDC - LÍNEAS DE CÓDIGO



Como manejo las líneas en blanco, comentarios , etc

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)?

Métrica de

Punto función

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

PF- Punto función (Albrecht 1978)

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

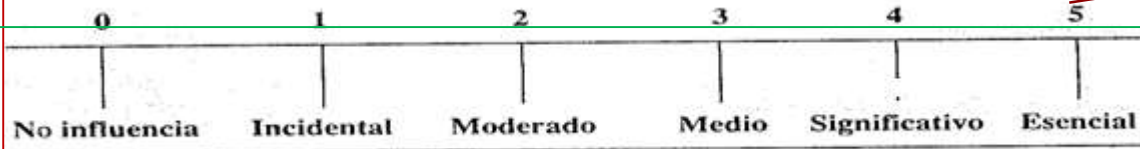
$$PF = TOTAL * [0.65 + 0.01 * SUM(F_i)] \quad i=1 \text{ a } 14 \quad 0 \leq F_i \leq 5$$

	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

Métrica de Punto función F(i)

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 = $\$ / PF$

Medida subjetiva ***independiente del lenguaje***, de estimación más fácil.

Métrica temprana

Desarrollo de una métrica- **GQM**

- ❖ 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 GQM

Objetivo

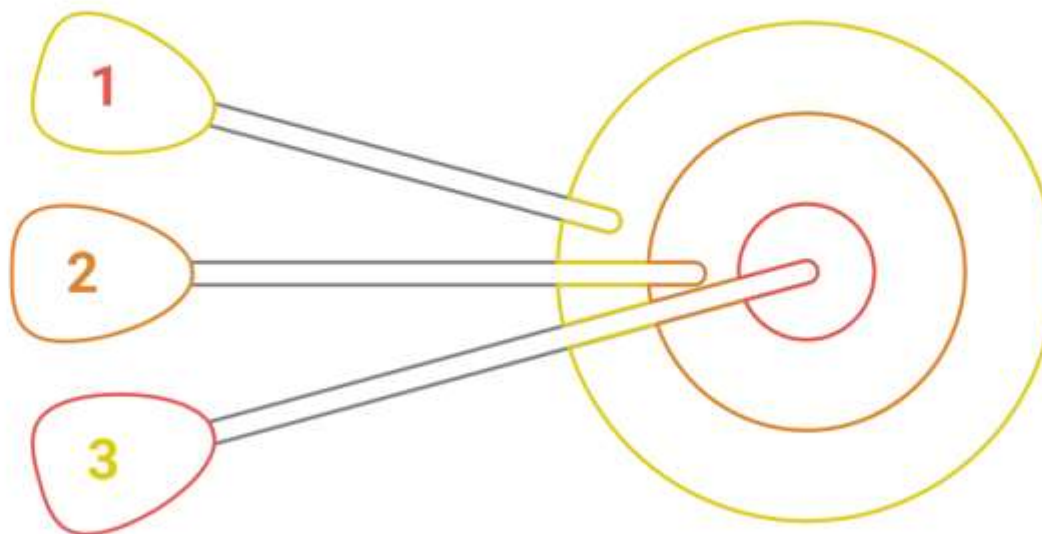
El objetivo central del proyecto

Preguntas

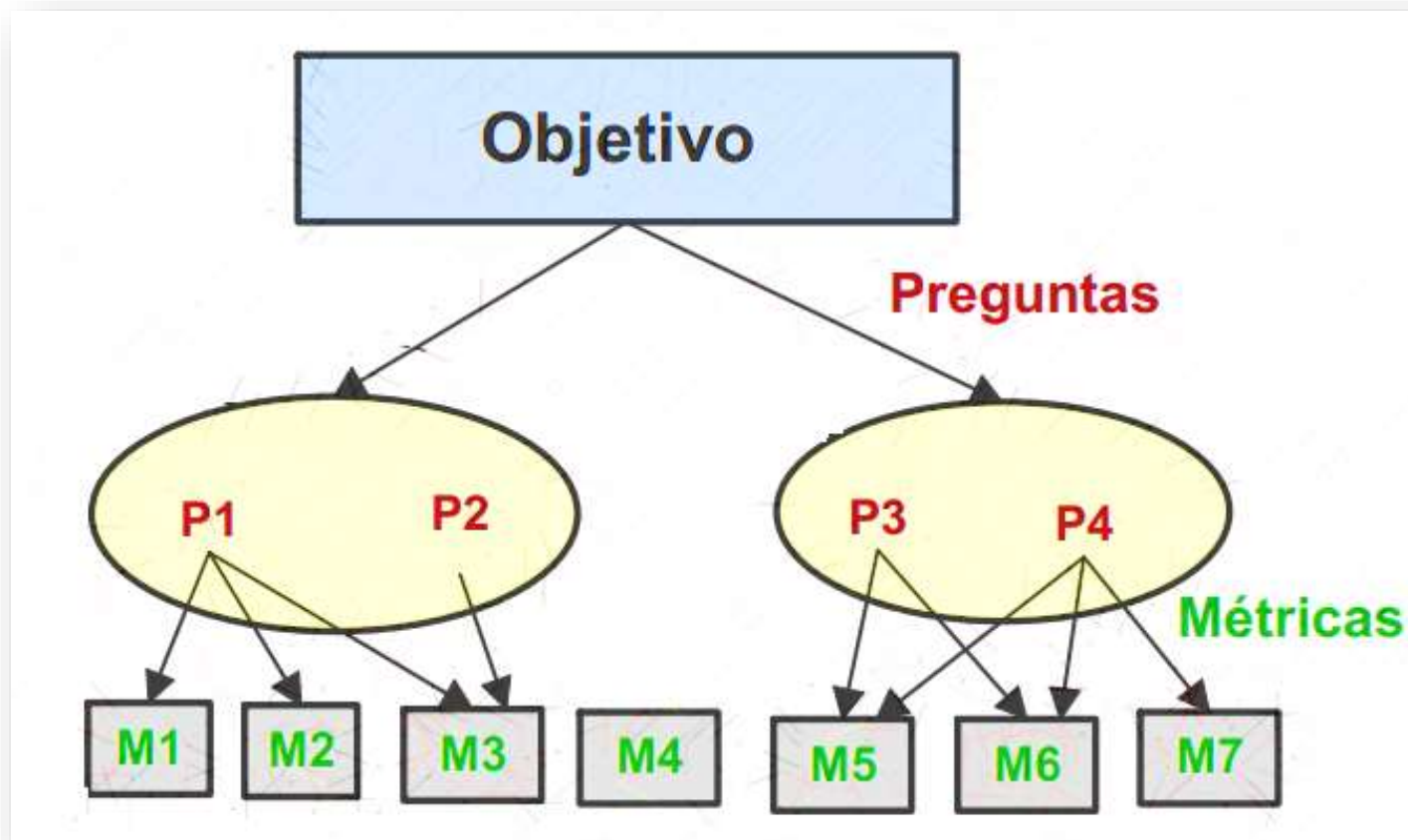
Preguntas para verificar el cumplimiento del objetivo

Métricas

Medidas cuantitativas para responder a las preguntas



GQM (OPM)



GQM Ejemplo

- ❖ Evaluamos, en la etapa de Análisis de Requerimientos, la tarea Asignación de responsabilidades.

<i>Propósito</i>	<i>Evaluar</i>	
<i>Característica</i>	<i>Asignación de responsabilidades</i>	
<i>Punto de Vista</i>	<i>Gerencia de Proyecto</i>	
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 está disponible?	
	M5	Valor Binario

Indicadores

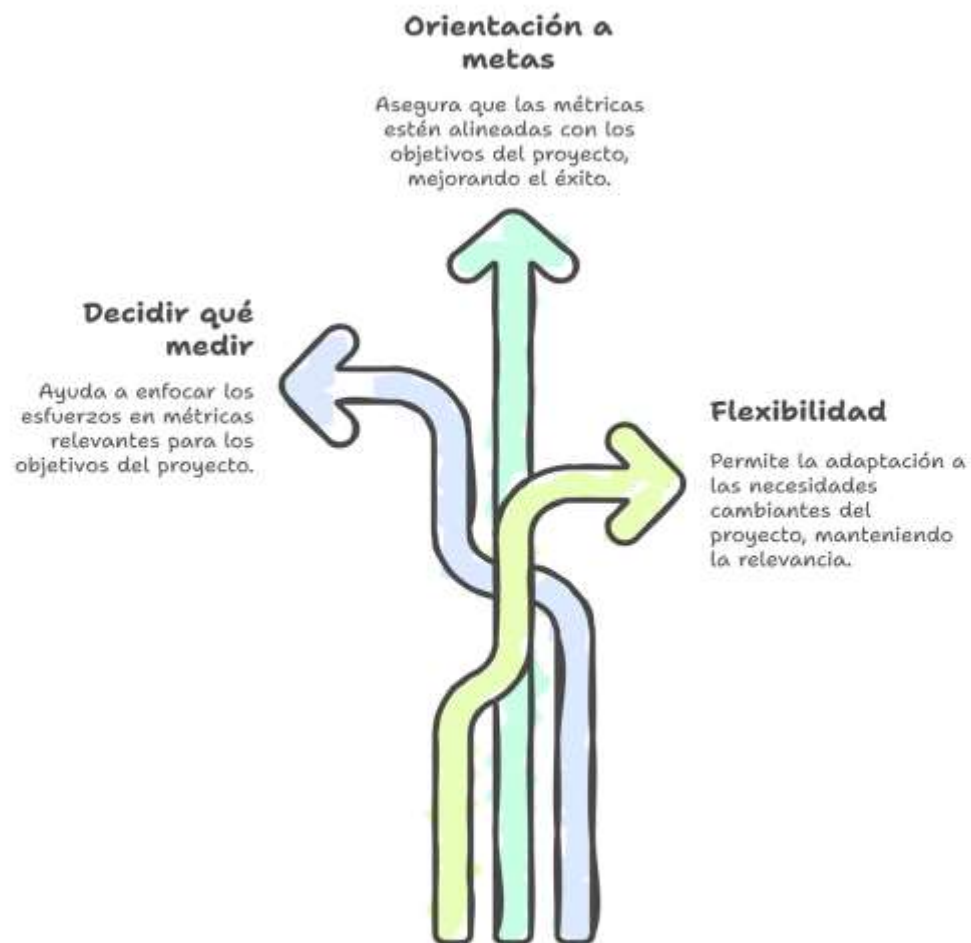
<u>Nombre</u>	<u>Descripción</u>	<u>Fórmula</u>
I1	Gestión de Asignación de roles	M2 & M3 & M4 & M5
I2	Proceso de Asignación de roles	M1 & M4 & M5

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

¿Cómo implementar el marco GQM?



Made with Napkin



Gestión de Proyectos

Estimaciones

Estimaciones



¿Qué son?



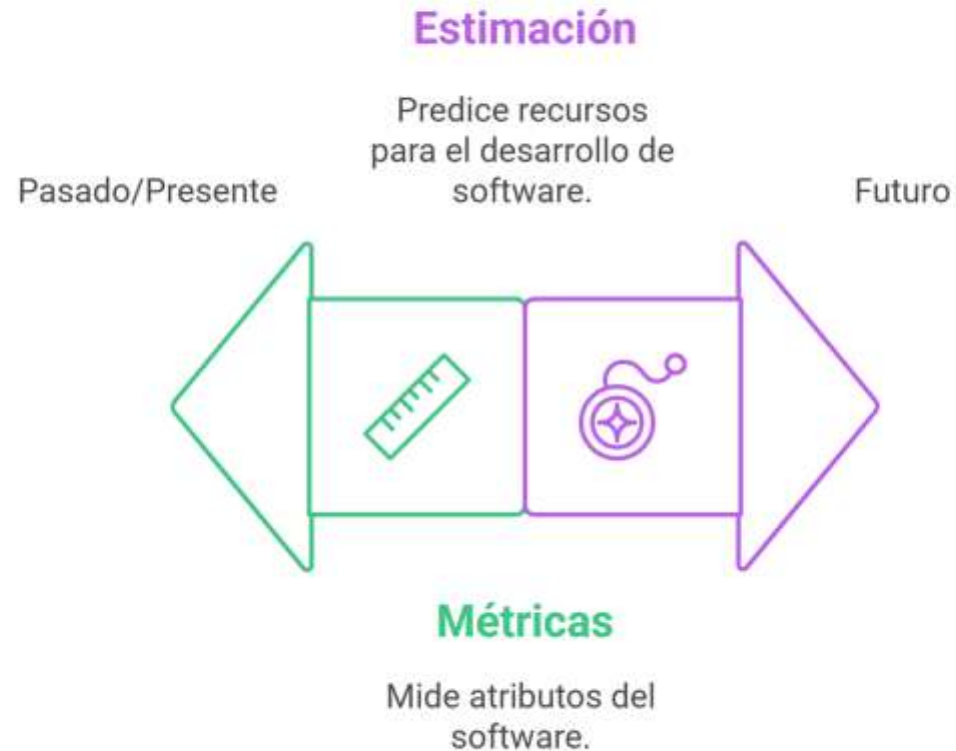
¿Cómo usarlas?

La estimación del software es el proceso de predecir la cantidad más realista de esfuerzo (expresado comúnmente en horas persona o costo monetario) requerido para desarrollar o mantener software.

29

Estimaciones

Las métricas y la estimación de software se enfocan en la perspectiva temporal.



30

Made with  Napkin

Estimaciones



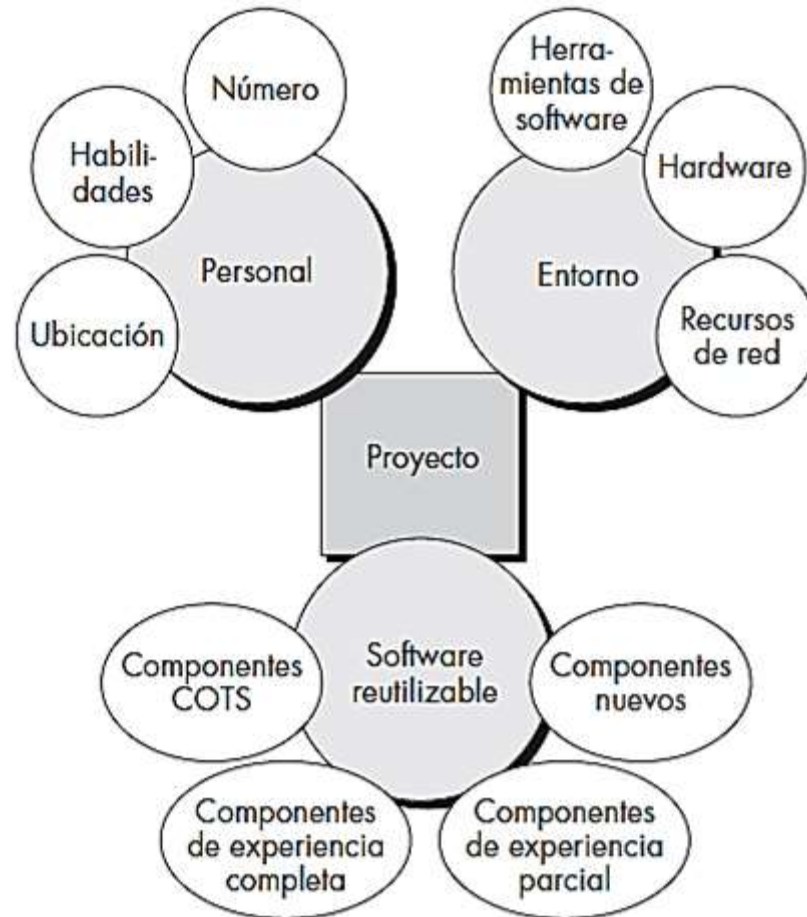
❖ ¿Qué podemos estimar?



- ❖ ¿Qué tener en cuenta?
- ❖ ¿Qué factores influyen?

31

Estimaciones de recursos



32

Estimaciones de costos

3 parámetros para calcular costo de un proyecto



33

Fijación de precio - Relación Precio Costo

❖ ¿Qué tener en cuenta ?

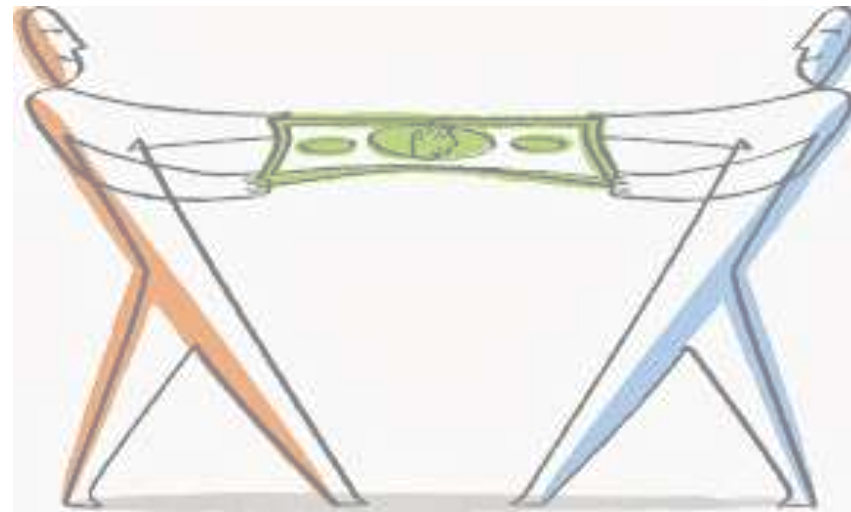
Oportunidad de mercado	Una organización de desarrollo podría ofertar un bajo precio debido a que desea conseguir cuota de mercado. Aceptar un beneficio bajo en un proyecto podría darle la oportunidad de obtener más beneficios posteriormente. La experiencia obtenida le permite desarrollar nuevos productos.
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.

34

Fijación de precio

- ❖ Debe pensarse en :
- ❖ los intereses de la empresa,
- ❖ los riesgos,
- ❖ el tipo de contrato.

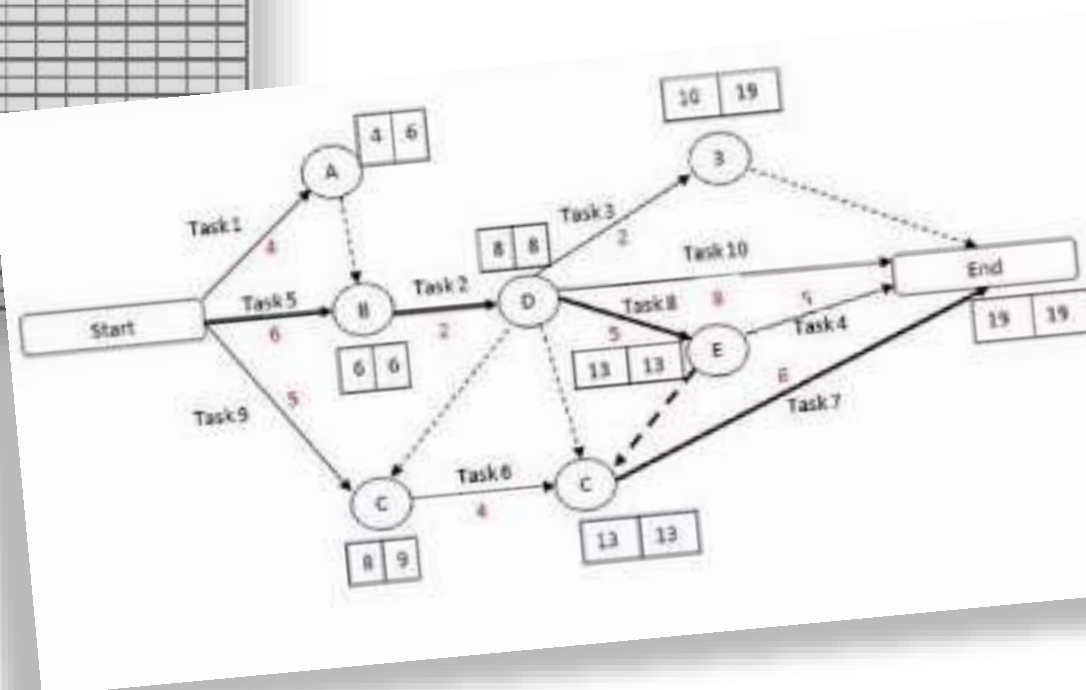
Esto puede hacer que el precio suba o baje.



35

Estimaciones de tiempo

Carta Gantt (Control de las Actividades Proyecto Servicio)																									
Nombre del Equipo: Los Triunfadores		Curso: 2º Medio "X"																Año: 2008							
Actividades		Agosto				Septiembre				Octubre				Noviembre				Diciembre							
Nº	Meses - Semanas	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4				
1	Redacción del Proyecto																								
2	Diseño del Servicio																								
3	Cotización de Materiales																								
4	Compra de materiales																								
5	Presentación del Proyecto																								
6	Promoción del Servicio																								
7	Prestación del Servicio																								
8	Evaluación de Proceso																								
9	Control de Calidad																								



36

Técnicas de estimación

Juicio experto

Consulta a varios expertos.
Estiman. Comparan.
Discuten

Técnica Delphi

Sucesivas rondas. Anónimas.
Consenso

División de Trabajo

Jerárquica hacia arriba

Planning Poker

Basada en consenso,
incluye todo el equipo de
desarrollo, iterativa

Técnicas de estimación- Juicio experto

Se basa en la opinión y el conocimiento acumulado de un individuo o grupos que poseen experiencia y pericia relevante en el desarrollo de software, tecnologías específicas, dominios de negocio o gestión de proyectos.

38

Consiste en solicitar a una o varias personas consideradas "**expertas**" que proporcionen una estimación del esfuerzo, tiempo, costo o complejidad necesaria para completar una tarea, característica, módulo o proyecto de software determinado. Los expertos en general no son anónimos.

Técnicas de estimación-

Proceso de estimación por Juicio experto



Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education Limited.

Tecnica Delphi

Método estructurado de comunicación y pronóstico que se basa en la agregación de opiniones de un panel de expertos ((más de un siempre) para llegar a un consenso sobre una estimación

Se caracteriza por:

Anonimato: las identidades de los expertos que participan siempre se mantienen anónimas entre sí. Esto reduce la influencia de personalidades dominantes y permite que cada experto exprese sus opiniones libremente.

Iteración: el proceso se lleva a cabo en múltiples rondas. En cada ronda, los expertos proporcionan sus estimaciones iniciales y las justificaciones de la estimación.

Retroalimentación controlada: después de cada ronda, un facilitador recopila, resume y distribuye las respuestas a todos los expertos.

Búsqueda de consenso: basándose en la retroalimentación, los expertos tienen la oportunidad de revisar y ajustar sus estimaciones en las rondas subsiguientes. El proceso continúa hasta que se alcanza un consenso razonable de estimación.

Fuente: Pfleeger, S. L., & Atlee, J. M. (2010). *Software Engineering: Theory and Practice* (4th ed.). Pearson Education.

División del trabajo (Top-down)

Consiste en descomponer un proyecto de software grande y complejo en componentes o actividades más pequeños, manejables y, por lo tanto, más fáciles de estimar de manera individual.

En esencia, el proceso implica:

- 1. Descomposición:** el proyecto de software se divide jerárquicamente en elementos de trabajo progresivamente más pequeños. La descomposición puede basarse en las funciones del software, las actividades del proceso de desarrollo (como análisis, diseño, codificación, pruebas) o una combinación de ambos.
- 2. Estimación de las Partes:** se estima el esfuerzo, el costo y/o la duración requerida para cada una de estas partes individuales. La estimación de estas partes más pequeñas tiende a ser más precisa que intentar estimar el proyecto completo de una sola vez, ya que la complejidad se reduce y es más fácil visualizar el trabajo involucrado.
- 3. Composición (Agregación):** las estimaciones de las partes individuales se suman o combinan para obtener una estimación global del proyecto completo.

Sommerville, I. (2016). *Software Engineering* (10th ed.).

Fuente: Pearson Education Limited.

Técnicas de estimación- Planning Poker

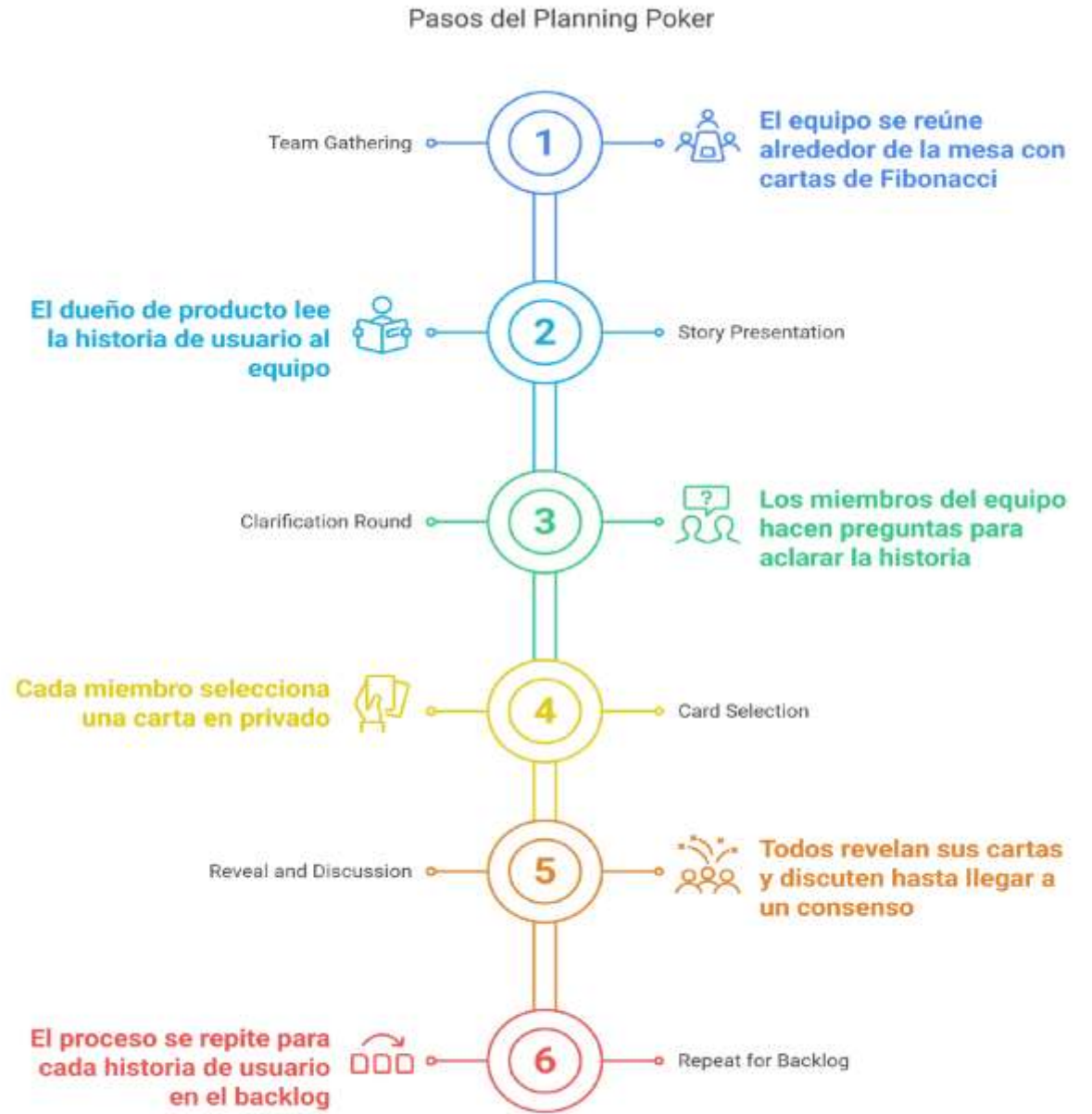
Es una técnica de estimación colaborativa y basada en el consenso utilizada en el desarrollo de software ágil, especialmente en los marcos de trabajo Scrum. Su objetivo es obtener estimaciones más precisas del esfuerzo requerido para las historias de usuario o elementos del backlog, fomentando la participación y el entendimiento compartido del equipo

- ❖ Participan todas las personas comprometidas en el desarrollo
- ❖ Cada una de las historias de usuario de un Sprint.
- ❖ Se llama así porque se utilizan cartas numeradas. Lo normal es numerar las cartas con una serie de Fibonacci .



Técnicas de estimación- Planning Poker

Se usan los números de Fibonacci para evitar la "trampa del anclaje" (donde la primera estimación mencionada influye en las demás)



Técnicas de estimación- Ejemplo

Historia de usuario: **Como administrador, quiero generar informes mensuales sobre la actividad de los usuarios y el rendimiento de las ventas.**

Estimación: 8 puntos de historia (secuencia de Fibonacci).

Justificación: El equipo considera que esta historia de usuario es una tarea más compleja y compleja. Requiere diseñar e implementar funcionalidades de informes, recopilar y agregar datos de múltiples fuentes y generar información valiosa. La complejidad y el esfuerzo que esto implica resultan en una estimación más alta.

44

Técnicas de estimación- App

En línea:

<https://planningpokeronline.com/>

45

¿Cuándo usar cada técnica de estimación?

Técnica de estimación Uso

Juicio experto

- En las primeras etapas del proyecto, cuando la información es limitada o el alcance es aún vago.
- Para proyectos pequeños o de corta duración donde la inversión en técnicas más elaboradas no se justifica.
- Cuando se necesita una estimación rápida y la precisión extrema no es crítica.
- Para tareas o proyectos altamente especializados donde pocas personas poseen el conocimiento necesario.
- Como punto de partida para refinar con otras técnicas más adelante.

Técnica Delphi

- Cuando se busca obtener un consenso de un grupo diverso de expertos, incluso si están geográficamente dispersos.
- Para proyectos complejos o de alto riesgo donde el sesgo de una sola persona o las dinámicas grupales pueden influir negativamente. Cuando no hay datos históricos suficientes y se depende en gran medida de la experiencia subjetiva.
- Para mitigar el efecto de la personalidad dominante en las discusiones grupales, ya que las respuestas son anónimas.

¿Cuándo usar cada técnica de estimación?

Técnica de estimación Uso

División del trabajo	<ul style="list-style-type: none">• Para proyectos grandes y complejos que necesitan ser descompuestos en componentes manejables.• Cuando se requiere un alto nivel de detalle y visibilidad sobre el alcance del proyecto.• Para asignar responsabilidades claras a equipos o individuos para partes específicas del proyecto. Como base para la planificación, el seguimiento y el control del proyecto.• Generalmente se utiliza en conjunto con otras técnicas de estimación (como el juicio experto o la estimación análoga/paramétrica) una vez que las tareas de bajo nivel han sido definidas
Planning Poker	<ul style="list-style-type: none">• Principalmente en metodologías ágiles (Scrum, Kanban) para estimar historias de usuario o elementos del backlog.• Cuando se desea fomentar la colaboración y el consenso dentro del equipo de desarrollo.• Para identificar y discutir suposiciones, dependencias y riesgos de las tareas de forma proactiva.• Cuando se busca que el equipo de desarrollo sea dueño de sus propias estimaciones, lo que fomenta un mayor compromiso y precisión.• Ideal para estimar elementos que varían en complejidad y requieren una discusión abierta para ser bien entendidos.

Modelos empíricos de estimación

COCOMO (Constructive Cost Model) es un modelo de estimación de costos para proyectos de software, desarrollado por Barry Boehm.

Este modelo utiliza fórmulas matemáticas para predecir el esfuerzo, el tiempo y el costo de un proyecto de software en función de su tamaño, complejidad y otros factores.

- Se basa en la experiencia y en datos de proyectos reales.
- Considera factores como el tamaño del proyecto, la complejidad del código, la experiencia del equipo y el entorno de desarrollo.
- Ofrece diferentes niveles de detalle para la estimación, desde modelos básicos hasta modelos detallados.

48

Modelos empíricos de estimación

Utilizan fórmulas derivadas empíricamente para predecir costos o esfuerzo requerido en el proyecto

COCOMO II

Modelo empírico que derivó de recopilar datos a partir de un gran número de proyectos de software.

49

fórmulas



vinculan

el tamaño del sistema y los factores del producto, proyecto y equipo **CON**



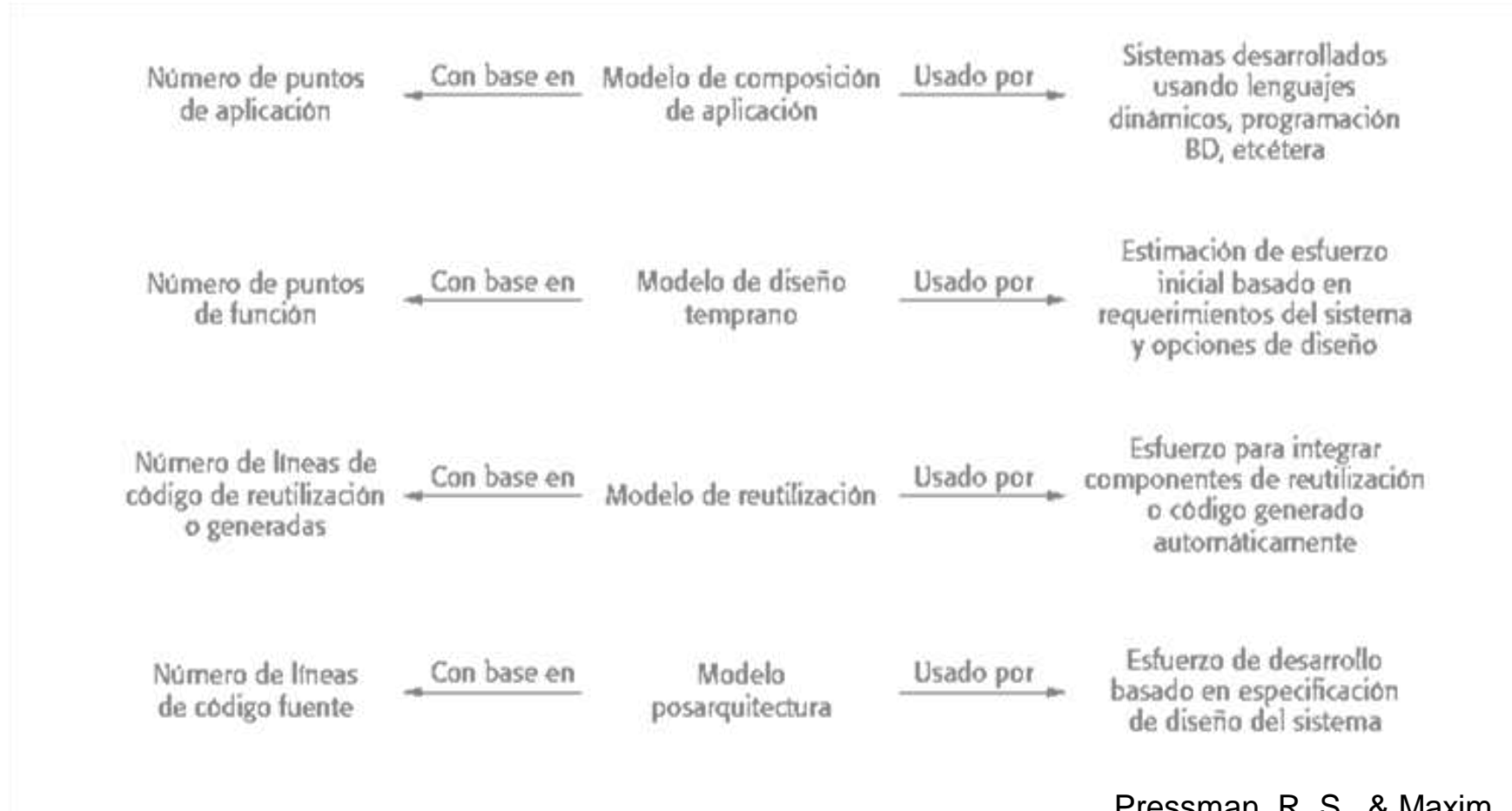
el esfuerzo para desarrollar el sistema.

Fuente: Somerville Cap. 26

Fuente:

Pressman, R. S., & Maxim, B. R. (2023). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.

COCOMO II - Modelos



50

COCOMO II

1. Modelo de composición de aplicación

- ➡ Modela el **esfuerzo** requerido para desarrollar sistemas creados a partir de proyectos de creación de prototipos.
- ➡ Se basa en una **estimación** de *puntos de aplicación* (o puntos objetos)

$$PM = (NAP \times (1 - \%reutilización/100))/PROD$$

de programación imperativa (como Java) y el número de líneas de lenguaje de script (*scripting language*).

Parámetros

PM = esfuerzo estimado en personas/mes




NAP = total de puntos de aplicación.

PROD = productividad medida en puntos objeto

1

COCOMO II

2. Modelo de diseño temprano

-  Puede usarse durante las primeras etapas de un proyecto arquitectónico detallado para el sistema.
-  Supone que se acordaron los requerimientos del usuario antes del proceso de diseño del sistema
-  La meta en esta etapa debe ser elaborar una estimación

$$PM = A \times \text{Tamaño}^B \times M$$

Parámetros

PM = esfuerzo estimado en personas/mes

A = 2,94. (según Boehm)


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


B = esfuerzo requerido conforme aumenta el tamaño del proyecto. Puede variar de 1,1 a 1,24

M = definido por 7 atributos de proyecto y proceso que aumentan o disminuyen la estimación. EJ: fiabilidad y complejidad del producto, etc

COCOMO II

3. Modelo de reutilización

 se emplea para estimar el esfuerzo requerido al integrar código

 Para el código generado automáticamente, el modelo estima el esfuerzo requerido para integrar este código

$$PM_{\text{auto}} = (ASLOC \times AT/100)/ATPROD.$$

Parámetros

PM =esfuerzo estimado en persona s/mes

AT = % 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

53

COCOMO II

4. Modelo de post-arquitectura



Se usa cuando está disponible un diseño arquitectónico inicial (se conoce la estructura) . Entonces es posible hacer estimaciones para cada parte del sistema.

Las estimaciones están basadas en la misma fórmula básica

$$PM = A \times \text{Tamaño}^B \times M$$

54



pero se utiliza un conjunto más extenso de atributos de M



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

- **líneas nuevas** de código a desarrollar.
- líneas de código fuente **equivalentes** (ESLOC) calculadas usando el nivel de reutilización.
- líneas de código que **tienen que modificarse** debido a cambios en los requerimientos.
- Estas estimaciones se añaden para obtener el tamaño del código (KLDC).

Modelos COCOMO

Limitaciones:

- Los modelos de COCOMO pueden ser complejos de usar, y requieren de datos históricos y experiencia para calibrar los factores de costo.
- Se miden los costes del producto, de acuerdo a su tamaño y otras características, pero no la productividad.
- La medición por líneas de código no es válida para orientación a objetos; entre otras cosas por la "reusabilidad" y la herencia, características de este paradigma (e.g., puede implicar importante aumento en productividad; pero no en líneas de código).

55

Estimaciones

❖ COCOMO 1:

<https://strs.grc.nasa.gov/repository/forms/cocomo-calculation/>

❖ COCOMO 2:

<http://softwarecost.org/tools/COCOMO/>

56