

Recordar que de la gestión de proyecto forman parte varias tareas

## Elementos clave de la gestión de proyectos

---

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

### Qué es una métrica?

Es una clave tecnológica para el desarrollo y el mantenimiento del sistema. Esas métricas, en realidad tienen como objetivo fundamental:

- entender qué ocurre durante el desarrollo y el mantenimiento de nuestro proyecto
- controlar qué es lo que ocurre
- mejorar nuestro proyecto y nuestro producto
- evaluar la calidad.

---

algunas definiciones....

Medida → indicación cuantitativa de algo, de una extensión, de una cantidad, de una capacidad, de un atributo de un proceso o un producto. *cuántas personas requiere un proyecto ; qué cantidad de líneas de código tiene mi producto. qué cantidad de tiempo me va a llevar realizar ese proyecto.*

Medición→ El acto de medir. El acto de determinar una medida.

Métrica → lo que utilizo para lograr la medida. Aplico una métrica para realizar la medición y obtengo una medida.

Indicador→ es una combinación de métricas que proporciona una visión profunda que permite al gestor de proyecto ajustar al producto o el proyecto. *Por ejemplo, estoy midiendo el tiempo. Yo defino como indicador que para mi proyecto esté funcionando bien esa medida tiene que ser de 10 semanas. Si me da 20, eso significa que tengo que hacer algún ajuste en esta tarea porque el indicador me decía que la tenía que realizar en 10 días.*

---

Las métricas pueden ser usadas para que los profesionales e investigadores tomen mejores decisiones.



Métricas como medio de asegurar la calidad en  
procesos/proyectos/ productos

### Las métricas de 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.

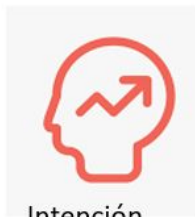
## Métricas del proceso

» 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

## Uso

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

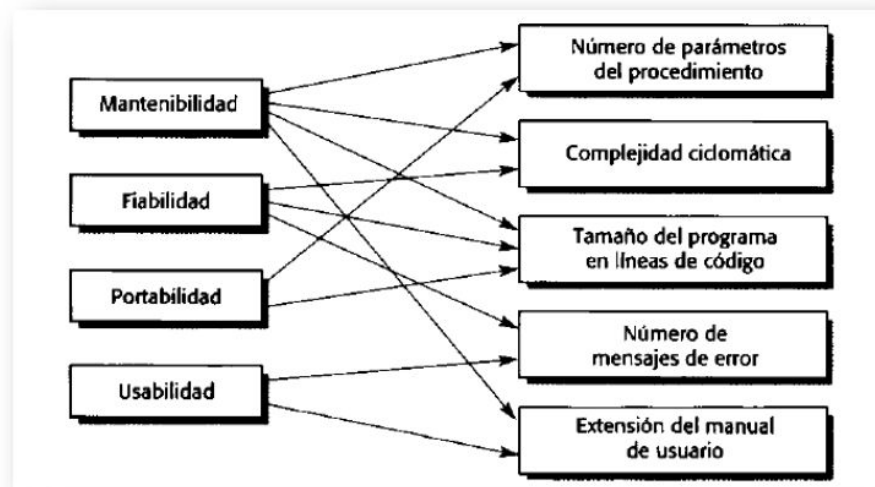
Característica del proceso	Cuestiones clave
Comprensión	¿En qué medida el proceso está definido explícitamente y qué tan fácil es entender la definición del proceso?
Estandarización	¿Hasta qué punto el proceso se basa en el proceso genérico estándar? Esto puede ser importante para algunos clientes que requieran la conformidad con un conjunto de estándares definidos de proceso. ¿Hasta dónde se usa el mismo proceso en todas las partes de una compañía?
Visibilidad	¿Las actividades del proceso culminan en resultados claros, de modo que el avance del proceso se observa externamente?
Mensurabilidad	¿El proceso incluye recolección de datos u otras actividades que permitan medir las características del proceso o del producto?
Soportabilidad	¿En qué medida pueden usarse herramientas de software para apoyar las actividades del proceso?
Aceptabilidad	¿El proceso definido es aceptable y útil para los ingenieros responsables de elaborar el producto de software?
Fiabilidad	¿El proceso está diseñado de tal forma que se evitan o se detectan errores de proceso antes de que deriven en errores de producto?
Robustez	¿El proceso puede continuar a pesar de problemas inesperados?
Mantenibilidad	¿El proceso puede evolucionar para reflejar los requerimientos cambiantes de la organización o mejoras identificadas en el proceso?
Rapidez	¿Qué tan rápido puede completarse el proceso de entrega de un sistema a partir de una especificación dada?

Estas son algunas de las características del proceso que se podrían medir.

**Cómo los medimos?** es muy difícil para nosotros decir que nuestro proceso es visible, o que tiene una comprensión establecida...

entonces tenemos que **conectar estos atributos del proceso con atributos internos de mi producto**. De esta manera tengo que encontrar los atributos internos que me permiten a mí medir por ejemplo la fiabilidad o la rapidez de un proceso.

## Atributos de calidad externos vs Atributos internos



## Métricas del producto



### dinámicas

- Hechas en un programa en ejecución.
- Ayudan a valorar la eficiencia y fiabilidad de un programa.



### estáticas

- Hechas de representaciones del sistema
- Ayudan a valorar la complejidad, comprensibilidad y mantenibilidad.

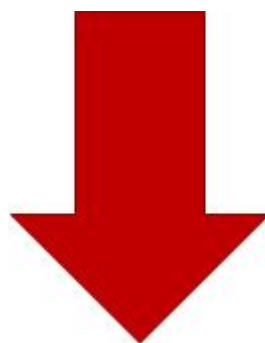
- métricas estáticas

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	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. <i>Cuanto más grande sea el índice de Fog, el documento será más difícil de comprender.</i>

- Métricas 00(estáticas tmb)- solo a modo de comentario, no entra-



Métrica orientada a objetos	Descripción
Métodos ponderados por clase ( <i>weighted methods per class</i> , WMC)	Éste 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</i> , DIT)	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</i> , NOC)	Ésta 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</i> , CBO)	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</i> , RFC)	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</i> , LCOM)	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.



★ La métrica de producto que ha sido la más común a lo largo del tiempo para el **tamaño de un producto** es → **LDC líneas de código**. Es una métrica muy discutida



porque si estoy iniciando un proyecto y decido aplicar esta métrica, no sé cuántas líneas de código va a tener, es decir, **recién se puede utilizar cuando el producto está terminado**. Uno tmb podría aplicarla si tengo registro de proyectos anteriores.

Qué medidas podemos sacar de esta métrica? medidas relacionadas con el tiempo, la cantidad de personas precisadas,

La utilidad de estas métricas postmortem es

- conformar una línea base para futuras métricas
- ayudar al mantenimiento conociendo la complejidad lógica, el tamaño, el 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



no tendríamos que tener en cuenta las líneas en blanco ni los comentarios, ..etc.

Entonces se puede incorporar una variable más para hacer más certera la longitud del programa.

## 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

**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

### PF- Punto función (Albrecht 1978)

Factor de Ponderación, es subjetivo y esta 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

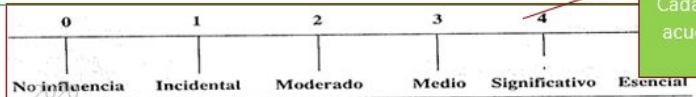
Son valores de ajuste de la complejidad según las preguntas de la siguiente pantalla

La métrica de **punto función** mide la cantidad de **funcionalidad de un sistema descrito en una especificación.**

Esas cantidades que tengo (#entradas, #salidas, etc) lo tengo que multiplicar por un **factor de ponderación** que es muy subjetivo que es dado por el equipo de desarrollo. Ese factor de ponderación puede ser simple, medio o complejo. Sumo todas estas cantidades y me da un TOTAL.

La fórmula de Punto Función está definida por este TOTAL anterior, multiplicada por 0.65+0.01 multiplicado por una sumatoria de  $F_i$  donde ese  $F_i$  son valores de ajuste de complejidad de acuerdo con el sistema que estemos queriendo realizar. Este  $F_i$  está dado por darle un valor a cada una de estas 14 preguntas:

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

Con la medida de punto función también puedo sacar métricas derivadas de:

- **Productividad:** relación entre PF y Persona\_mes
- **Calidad:** relación entre Errores y PF
- **Costo:** relación entre \$ y PF

$$\text{Productividad} = \text{PF} / \text{Persona\_mes}$$

$$\text{Calidad} = \text{Errores} / \text{PF}$$

$$\text{Costo} = \$ / \text{PF}$$

Lo que tiene de beneficioso esta métrica es que es **subjetiva independiente del lenguaje**, de **estimación más fácil**. Y además es **temprana**. (no hace falta que termine el proyecto para utilizarla)

## 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 está orientado a lograr una métrica que “mida” cierto objetivo. El mismo nos permite mejorar la calidad de nuestro proyecto.

Es una métrica que nosotros vamos a definir el objetivo y en base a esa vamos a generar nuestra propia métrica. Por ello es que es muy amplia y se puede aplicar en cualquier parte del proceso.

» 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 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 esta disponible?	
	M5	Valor Binario

Tengo que tratar de establecer cómo armar la pregunta para que me ayude a entender o a evaluar lo que estemos evaluando.

Ahora tengo que asignar o crear los indicadores que me van a asegurar de que la asignación de responsabilidades se realizó bien o mal, si cumplí con el objetivo o no.

<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:

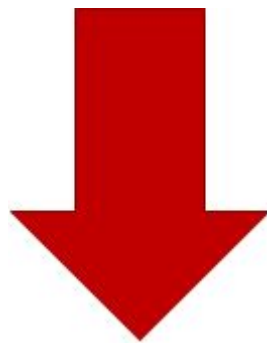
<i>I1</i>	Gestión de Asignación de roles	Verdadero
<i>I2</i>	Proceso de Asignación de roles	Verdadero



tenemos dos indicadores : i1, i2.

i1 lo defino en base a una concatenación de M2 & M3 & M4 & M5

Esto significa que la gestión de asignación de roles va a ser verdadera, va a estar bien , si las preguntas establecidas por M2 M3... son todas verdaderas. A esas preguntas yo debería haber respondido que sí.



## GQM (OPM)

---

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

Es flexible porque las preguntas las voy a generar yo y le voy a dar la profundidad que necesite.

---

<https://www.youtube.com/watch?v=TV8lytb4znU&list=PLBbCqd1I5NdJYyXSh7K8a3leffXSZioDr&index=8>

## ESTIMACIONES

### ¿Qué son las estimaciones?

Son técnicas que permiten dar un valor aproximado a algo. Si uno está tratando de medir alguna cosa pero evaluamos que eso no se puede medir exactamente sino que es aproximado entonces utilizamos las estimaciones.

Cuales son las diferencias con las métricas? justamente eso, que el valor es aproximado. Podemos decir que para hacer alguna tarea específica podemos decir que vamos a tardar 2 días, pero si no estamos seguros vamos a decir que vamos aproximadamente tardar 2 días. Esa aproximación lo calculamos con estimaciones.

Cómo usarlas? hay que tener cuidado porque son aproximadas.

Generalmente se calcula con diferentes tipos de fórmulas de estimación y luego compararlas y hacer un resultado más certero.

qué podemos estimar? Podemos estimar recursos, podemos estimar costos, podemos estimar tiempo, pero siempre de manera aproximada.

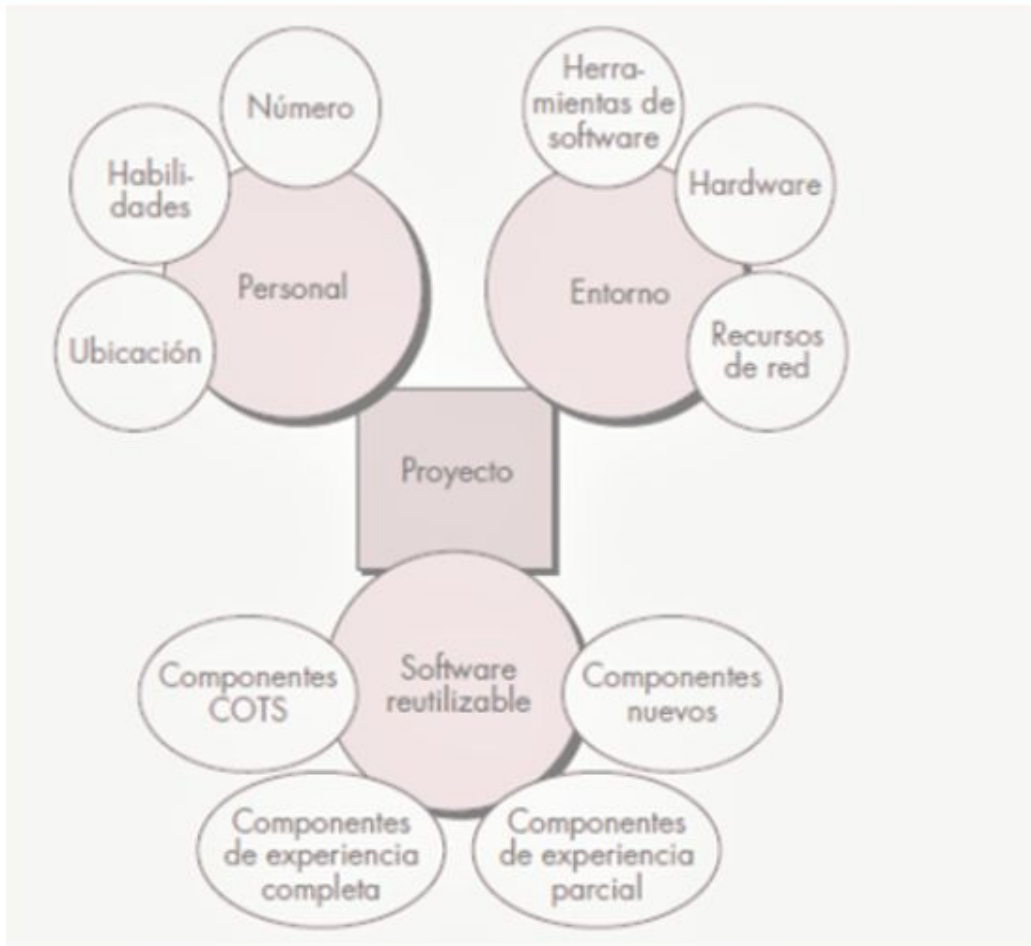
Qué tenemos que tener en cuenta? que no es exacta, que puede cambiar. Se requiere experiencia para estimar.

Qué factores se incluyen cuando uno hace una estimación? la complejidad del proyecto es crucial para que la estimación sea lo más certera posible. El tamaño del producto, la estructuración de un proyecto....

Dijimos que podemos estimar recursos, costos, y tiempo.

## **Estimación de recursos**

Qué recursos requiere nuestro proyecto?

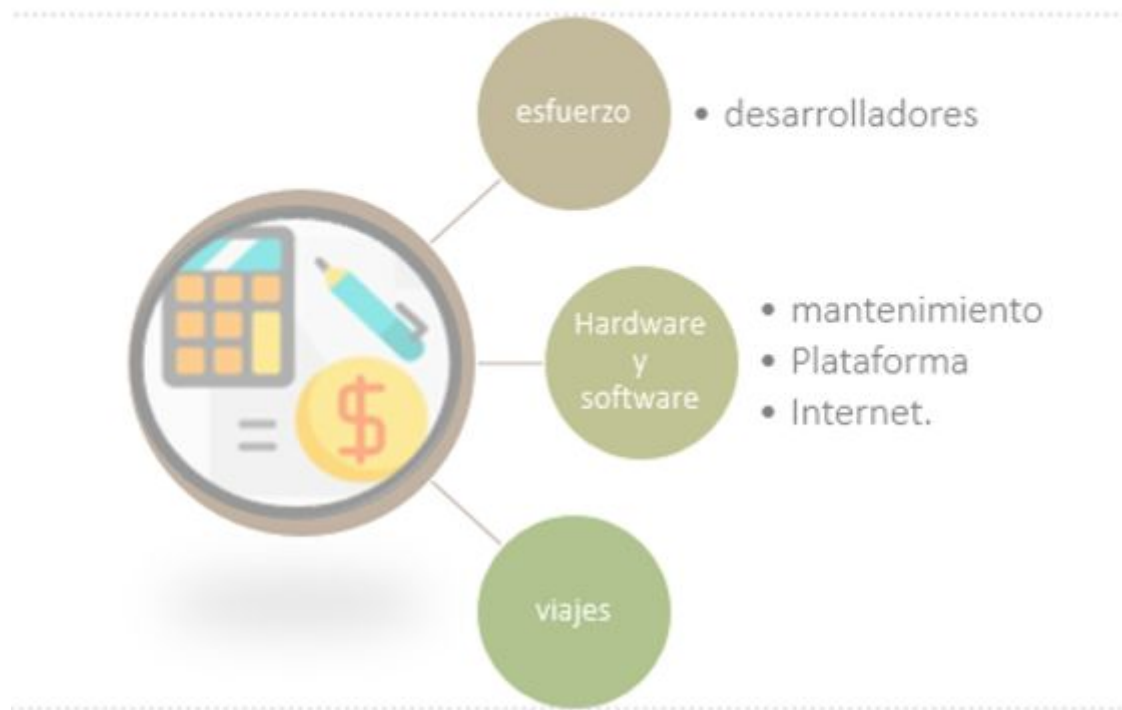


cuando hablamos de estimaciones de recursos tenemos que dejar muy claro qué cantidad de recursos necesitamos, qué habilidades deben tener esos recursos (los programadores x ej)

Todos estos recursos los voy a tener que establecer, describir, definir cuándo los voy a necesitar...etc

## **Estimaciones de costos**

Precisamos 3 parámetros para la estimación del costo:



Lo importante para estimar el costo del proyecto es tener en claro que siempre tenemos que tener establecido el plan de proyecto y las especificaciones de requerimientos bien establecidas para que el cálculo sea lo más certero posible. Cuando estimamos el costo no estamos estimando el precio. El precio es lo que le vamos a cobrar al cliente y puede ser tan cercano al costo como nosotros queramos.  
(un 10% más, un 20% más)

### **Fijación de precio- relación precio costo**

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.

2020

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

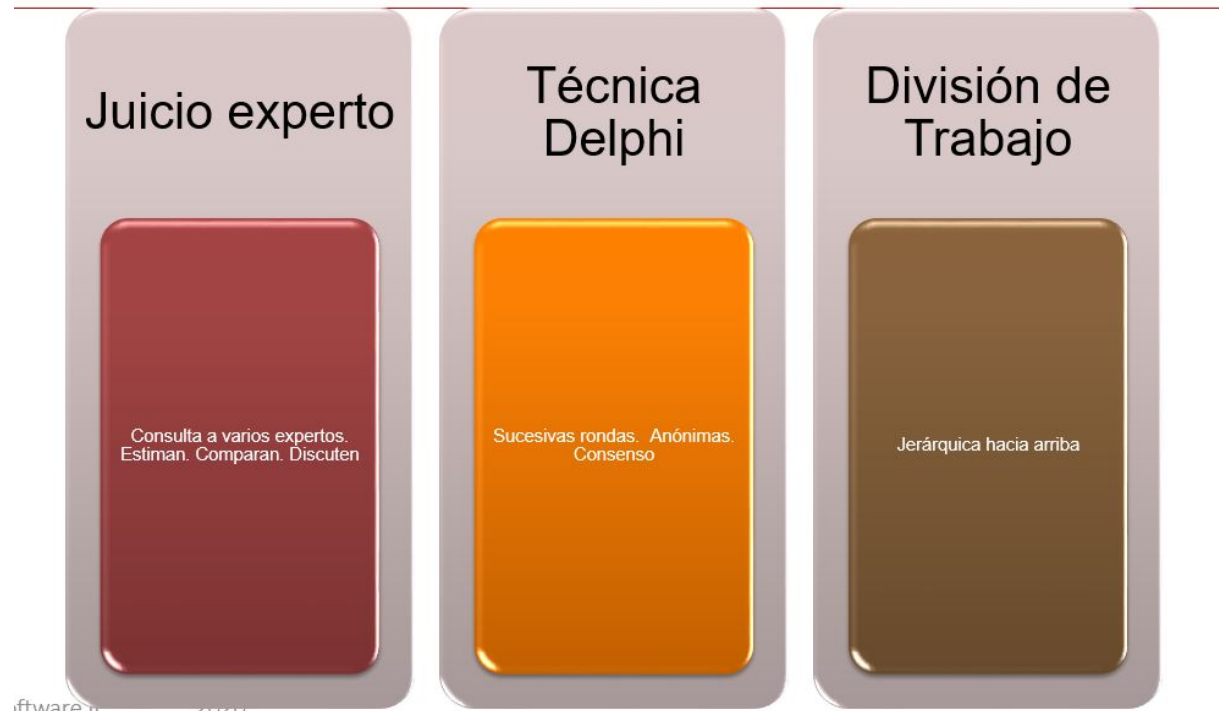
Esto puede hacer que el precio suba o baje.

Si un cliente nos pide que hagamos un producto pero que además le demos el código fuente. En ese caso deberíamos cobrarle más porque le estamos dando todo nuestro trabajo, si tuviéramos el código fuente lo podemos volver a vender a otro cliente.



## Estimaciones de tiempo

Las 3 técnicas más usadas son:



Son bastante similares entre sí.

- El juicio experto es una técnica de estimación que consulta sobre algo que nosotros queremos estimar a diferentes expertos. Ellos estiman, comparan, discuten y nos dan un número, una estimación.
- La técnica delfi se diferencia en que nosotros elegimos a los expertos y les pedimos que nos estimen algo que queremos estimar pero esos expertos son anónimos y no se conocen entre sí. Hacemos varias rondas de estimaciones hasta llegar a un consenso. Es más independiente.

- División del trabajo: en vez de preguntarle a expertos empezamos a preguntarle a los que están mas abajo en la jerarquía y a medida que van llegando a un consenso se va subiendo en la jerarquía.

Estas 3 anteriores son las que más se usan pero también existen los **Modelos empíricos de estimación**

**Los modelos empíricos de estimación** utilizan fórmulas derivadas empíricamente para predecir costos o esfuerzo requerido en el proyecto. Se utilizan más que nada en proyectos grandes por la complejidad que tiene la aplicación de estos modelos empíricos.

uno es el COCOMO II



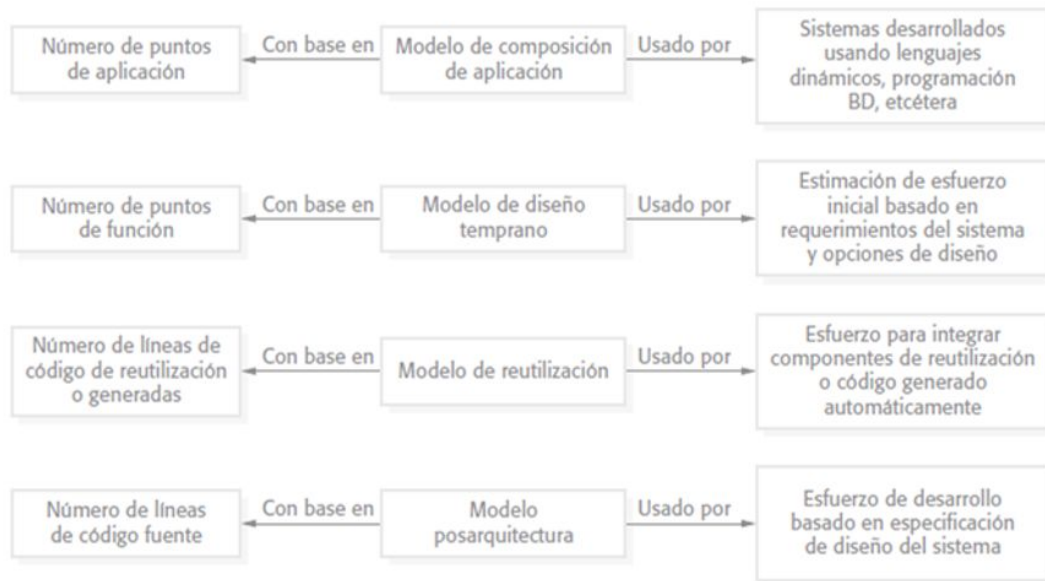
Fuente: Somanilla Cap. 26

este cocomo II presenta una fórmula que vincula....

El COCOMO II reconoce que las líneas de código son difíciles de estimar tempranamente y considera diferentes

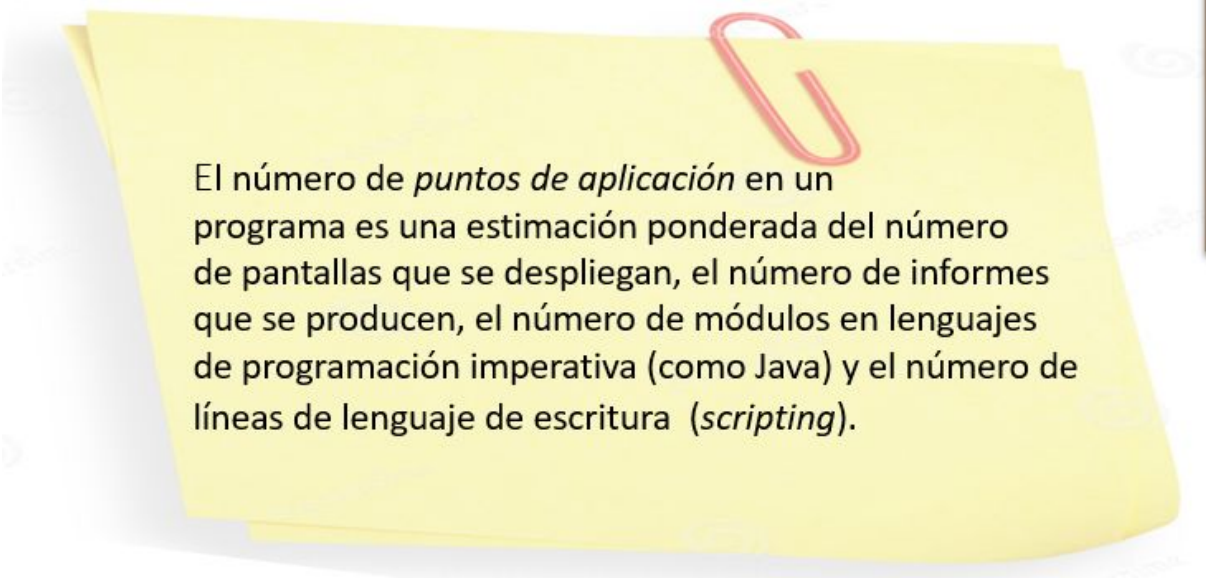
enfoques para el desarrollo como : el desarrollo de prototipos, el desarrollo en espiral, el desarrollo basado en componentes..etc

Este COCOMO II está compuesto por 4 modelos:



**1. El modelo de composición de aplicación** que modela el **esfuerzo** requerido para desarrollar sistemas que se crean a partir de componentes de reutilización y de proyectos de creación de prototipos .

Se basa en la estimación de puntos de aplicación (o puntos objeto)



El número de *puntos de aplicación* en un programa es una estimación ponderada del número de pantallas que se despliegan, el número de informes que se producen, el número de módulos en lenguajes de programación imperativa (como Java) y el número de líneas de lenguaje de escritura (*scripting*).

La fórmula es:

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

### Parámetros

**PM** = esfuerzo estimado en personas/mes

**NAP** = total de puntos de aplicación.

**PROD** = productividad medida en puntos objeto

El porcentaje de reutilización es una estimación de la cantidad de código que se va a reutilizar.

**2. Modelo de diseño temprano:** se puede usar para las primeras etapas de un proyecto, antes de que esté disponible un diseño arquitectónico detallado para el sistema.

→ supone que se acordaron requerimientos del usuario y que están en marcha las etapas iniciales del proceso de diseño del sistema. *Este es el modelo que vamos a usar en este momento.*

La idea es que la meta en esta etapa es estimar de manera temprana rápida y aproximada de los costos.

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

### 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

- 3. Modelo de reutilización:** Se emplea para estimar el esfuerzo requerido al integrar código de reutilización o generado.



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

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

### Parámetros

**PM** =esfuerzo estimado en personas/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

**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

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

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).

PM(persona mes)

tamaño B acá es en líneas de código (arriba dice como se calcular).

M es un valor que estaba basado en atributos igual que el anterior pero mucho más extenso , con mucha mas cantidad de atributos.