

Ingeniería de Software

Estimación

Juan José Vanzetti



¿Qué es la estimación?

- ▶ La realización de estimaciones adecuadas sobre el tamaño y esfuerzo requerido es una de las características fundamentales de un proyecto de desarrollo de software exitoso.
- ▶ Las malas estimaciones o más comúnmente las no estimaciones, son posiblemente una de las principales causas de los fracasos.

¿Qué es la estimación?

- ▶ **Estimación** “Apreciar, poner precio, evaluar algo”
- ▶ **Estimación de proyectos de software** “Actividad de la planificación del proyecto de software que intenta determinar cuánto dinero, esfuerzo, recursos y tiempo tomará construir un sistema o producto software ”.

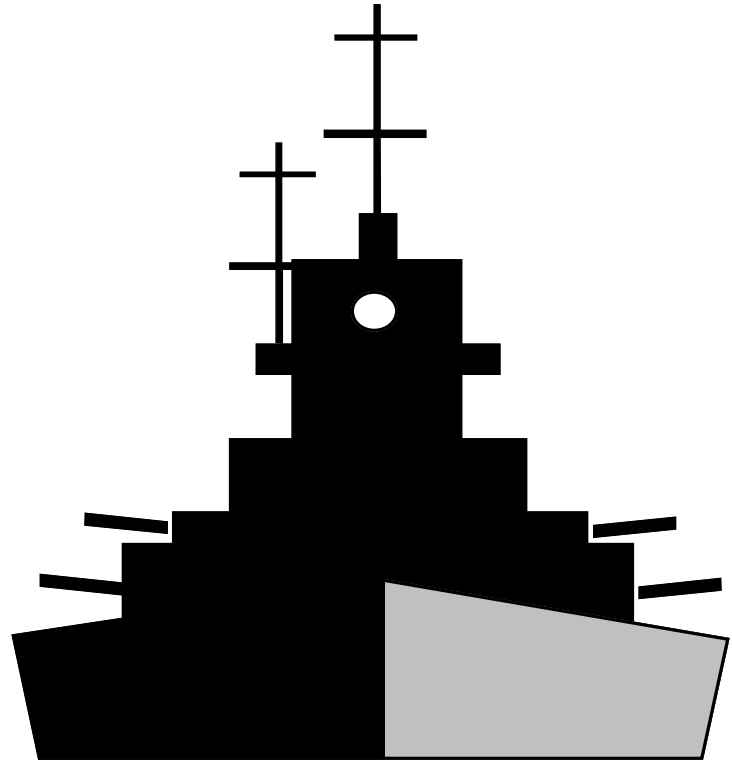
Problemática de la estimación

- ▶ Averiguar lo que costara desarrollar una aplicación.(meses-persona, \$, ...)
- ▶ Momento en que se desea conocer el costo
- ▶ Siempre se quiere muy pronto (Yourdon)

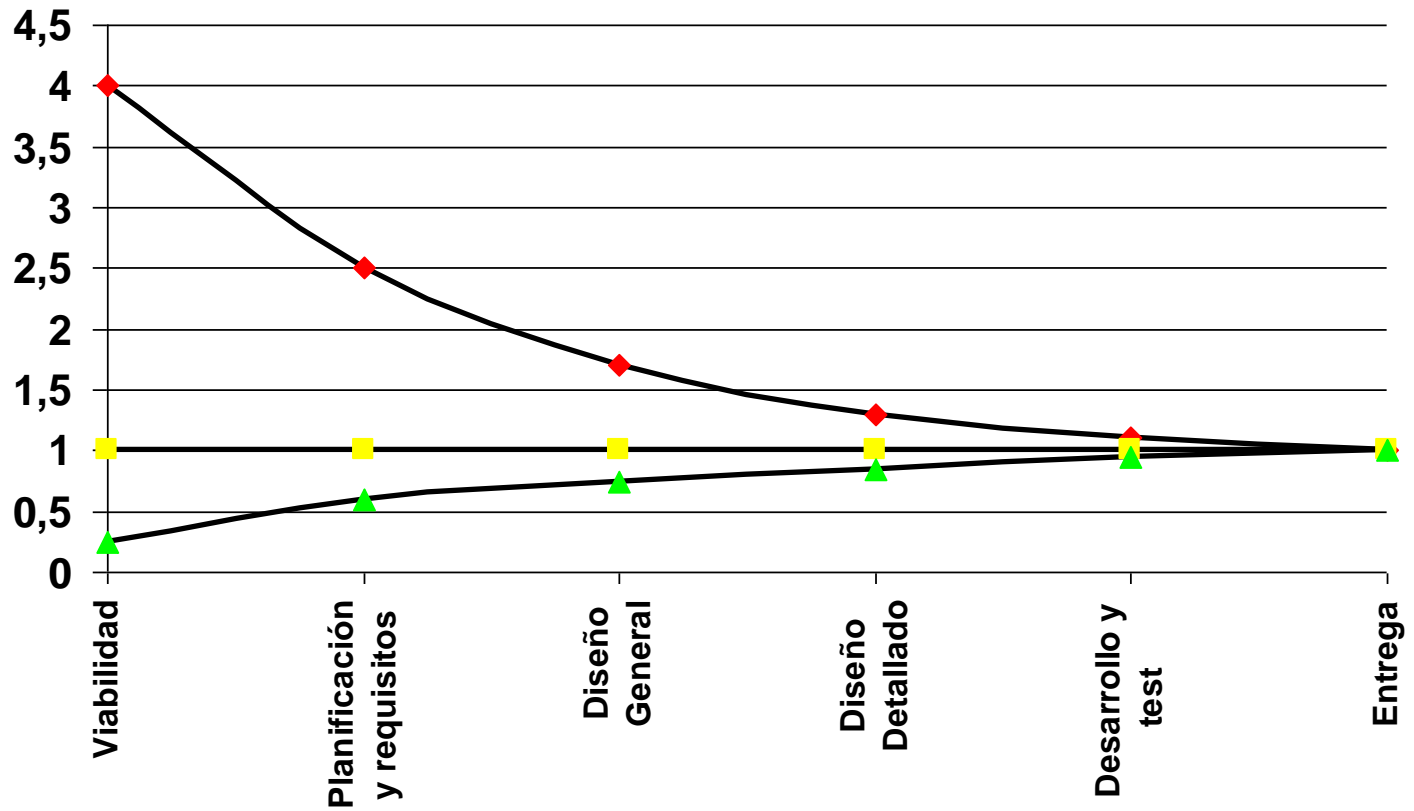
CONCEPTO DE ESTIMACIÓN DE PROYECTO SOFTWARE

- ▶ Estimar es echar un vistazo al futuro con algún grado de incertidumbre.
- ▶ La estimación, es mas un arte que una Ciencia.
- ▶ Es una actividad importante que no debe llevarse a cabo de forma descuidada.
- ▶ Una estimación es una predicción basada en un modelo probabilístico, no un modelo determinístico; es decir, la cantidad que se está estimando puede tomar no solamente un valor sino distintos valores
- ▶ *«Aplicación continua de técnicas basadas en las medidas de los procesos de desarrollo del software y sus productos, para producir una información de gestión significativa y a tiempo. Esta información se utilizará para mejorar esos procesos los productos que se obtienen de ellos» (SYMONS, C., 1998).*

Medir lo que quiere el usuario



Precisión de las estimaciones en función de la fase del proyecto



Objetivos de la estimación

- ▶ Predecir las variables involucradas en el proyecto con cierto grado de certeza.
- ▶ Trata de aportar una predicción de algún indicador importante para la gestión de proyectos de software tiempo, esfuerzo, cantidad de defectos esperados entre otros.
- ▶ Es razonable conocer, antes de comenzar a desarrollar el SW, cuánto se va a invertir, qué tareas se deben realizar y cuánto tiempo se necesitará.

Necesidad de un estimador

- ▶ Necesitamos un estimador
 - Temprano
 - Independiente del lenguaje
 - Fácil de calcular

Objetivo del estimador

- ▶ El estimador debe ser un profesional que no tenga ningún interés, directo o indirecto, en los resultados del proceso de estimación y que este únicamente guiado por su profesionalismo.
- ▶ El principal objetivo del estimador es obtener estimaciones de calidad, las cuales no tienen siempre por qué coincidir con las expectativas de la empresa en términos de costo y tiempo.

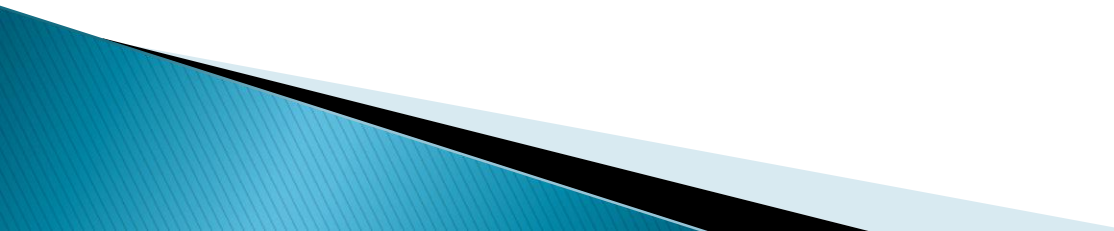
Requisitos que debe cumplir un buen estimador...

- ▶ Formación y experiencia profesional adecuada.
- ▶ Una posición en la organización que le permita adoptar un juicio independiente.
- ▶ Debe basarse en un método que pueda ser explicado, cuestionado, discutido y auditado.
- ▶ Debe poder describir su experiencia en cada estimación.
- ▶ Debe documentar su estimación, incluyendo los resultados obtenidos y cualquier información necesaria para hacer el proceso de estimación repetible y verificable.

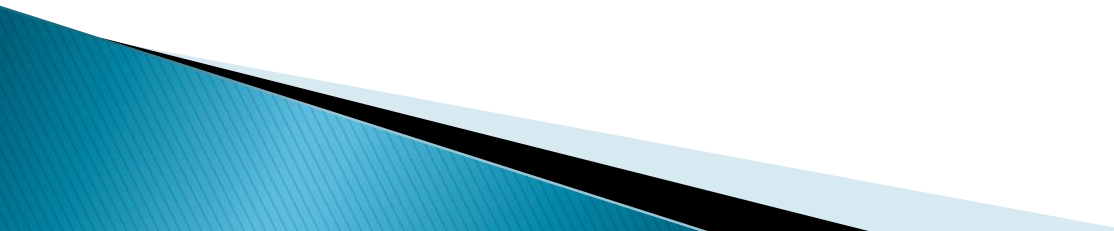
¿Cuándo se debe llevar a cabo?

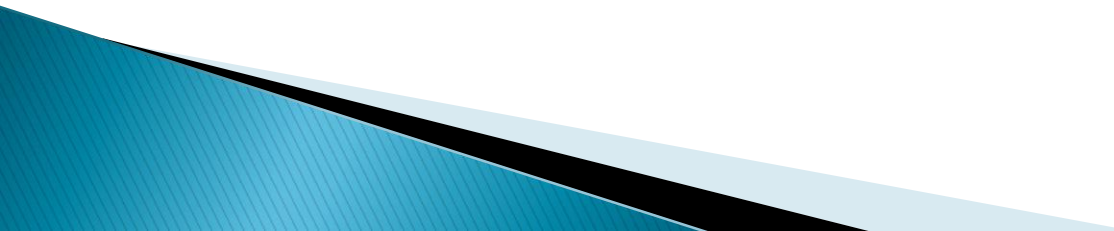
- ▶ La estimación es un proceso continuo. A medida que el proyecto avanza, más se conoce de él, y por lo tanto más parámetros están disponibles para introducir en un modelo de estimación.
- ▶ La estimación continua nos permite el uso de un único modelo coherente que pueda capturar y utilizar la información sobre el proyecto a medida que éste se conozca.
- ▶ El proceso de estimación comienza usando unas pocas variables claves para proveer las «macrocaracterísticas» de un proyecto, y evoluciona incorporando información de más bajo nivel para producir las «micro- características» del proyecto.

Factores que afectan al riesgo de la estimación

- ▶ **Complejidad del proyecto:** medida relativa.
 - ▶ **Tamaño del proyecto:** interdependencia de los elementos del software.
 - ▶ **Grado de estructuración del proyecto:** facilidad para poder agrupar las funciones y naturaleza jerárquica de la información.
 - ▶ **Disponibilidad de información histórica:** métricas sobre proyectos pasados.
- 

Problemas de Estimación:

- ▶ 1) **Problemas Políticos:** cuando las estimaciones se convierten en objetivos, cuando se ajusta el precio por conveniencia
 - ▶ 2) **Problemas Técnicos:** No existen datos históricos para estimar
- 

- ▶ Función.
 - ▶ Rendimiento: tiempos de respuesta y de proceso.
 - ▶ Restricciones: limitaciones del software debidas al hardware.
 - ▶ Interfaces.
- 

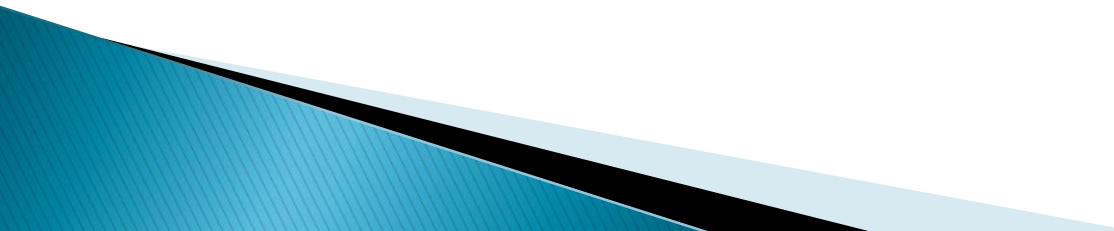
Humanos

- Habilidades requeridas: posición y especialidad.
- Disponibilidad.
- Duración de las tareas.
- Fecha de comienzo.

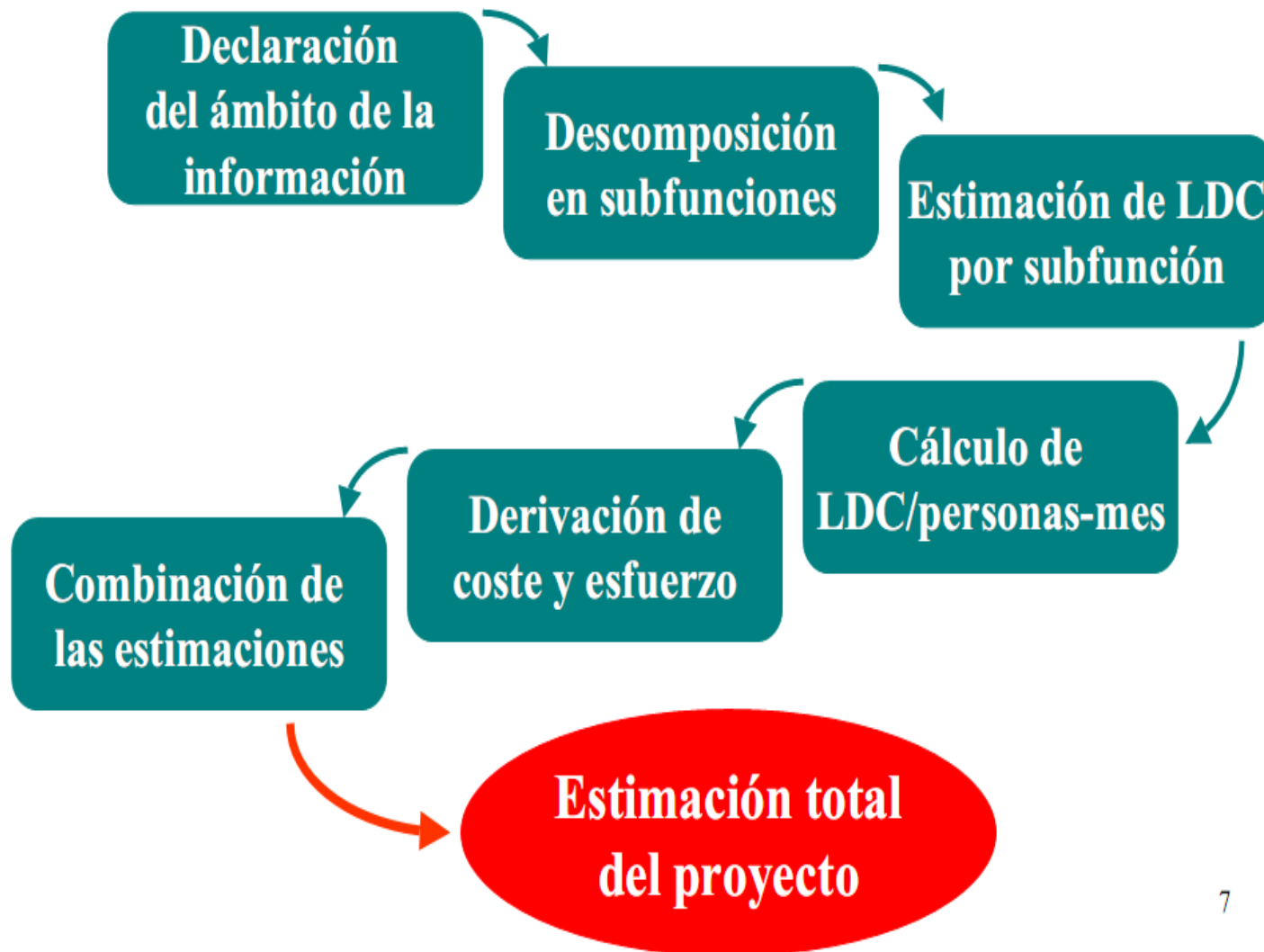
Software / Hardware

- Descripción.
- Disponibilidad.
- Duración del uso.
- Fecha de comienzo.

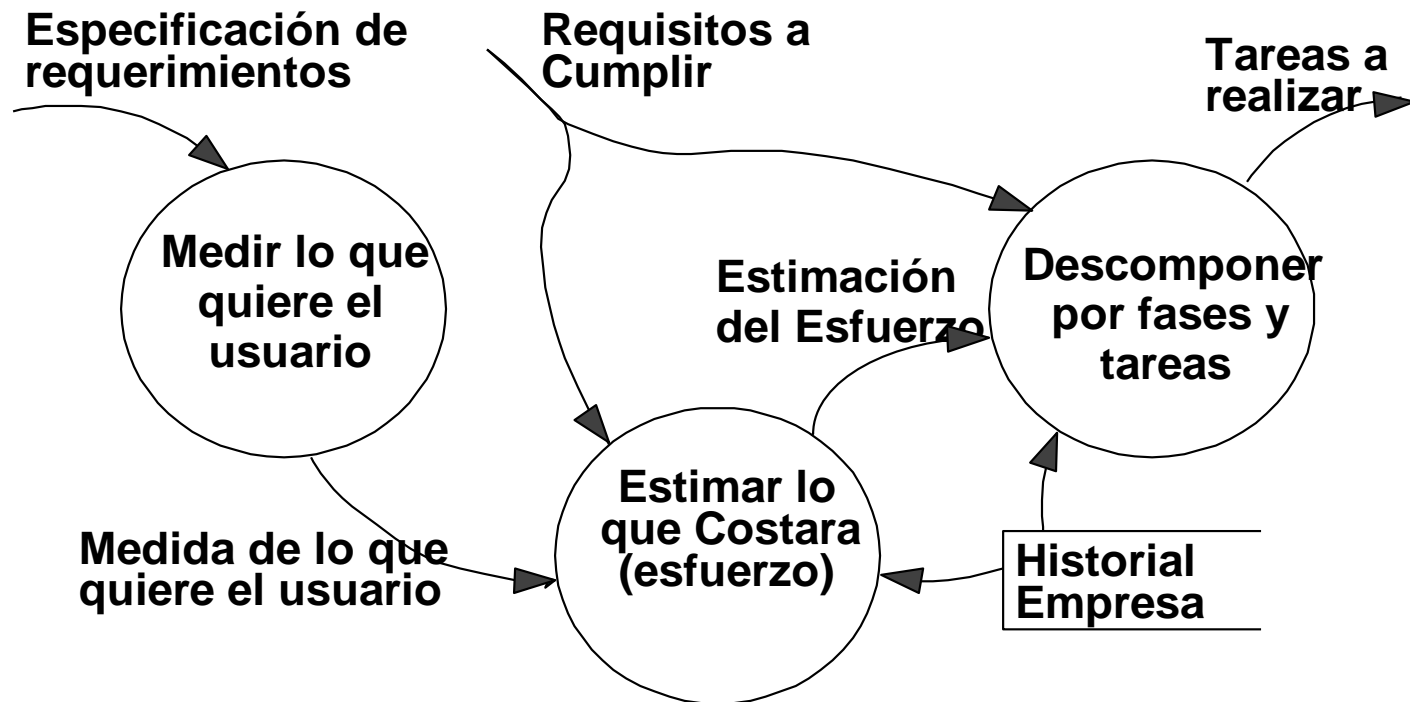
Estimación del proyecto

- ▶ Dejar la estimación para más adelante
 - ▶ Utilizar técnicas de descomposición
 - ▶ Utilizar un modelo empírico.
 - ▶ Adquirir herramientas automáticas de estimación.
- 

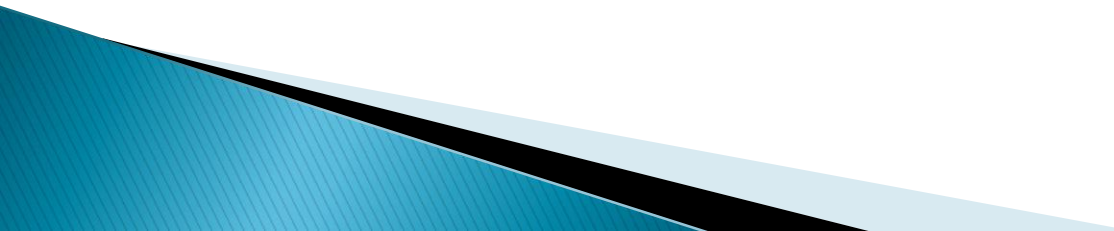
Estimación por descomposición



Proceso de Estimación propuesto



Métodos utilizados para la estimación de proyectos.

- ▶ Basados en la experiencia.
 - ▶ Basado exclusivamente en los recursos.
 - ▶ Método basado exclusivamente en el mercado.
 - ▶ Basado en los componentes del producto o en el proceso de desarrollo.
 - ▶ Métodos algorítmicos
- 

Métodos basados exclusivamente en la experiencia

- ▶ Juicio experto
 - Puro,
 - Delphi
- ▶ Analogía
- ▶ Distribución de la utilización de recursos en el ciclo de vida

Analogía

- ▶ Consiste en comparar las especificaciones de un proyecto, con las de otros proyectos.

Analogía, pueden variar los siguientes factores:

- ▶ **Tamaño:** ¿mayor o menor?
- ▶ **Complejidad:** ¿Más complejo de lo usual?
- ▶ **Usuarios:** Si hay más usuarios habrán más complicaciones.
- ▶ Otros factores:
 - Sistema Operativo, entornos (la primera vez más).
 - Hardware, ¿Es la primera vez que se va a utilizar?
 - Personal del proyecto, ¿nuevos en la organización?

Juicio experto: Puro

- ▶ Un experto estudia las especificaciones y hace su estimación.
- ▶ Se basa fundamentalmente en los conocimientos del experto.
- ▶ Si desaparece el experto, la empresa deja de estimar



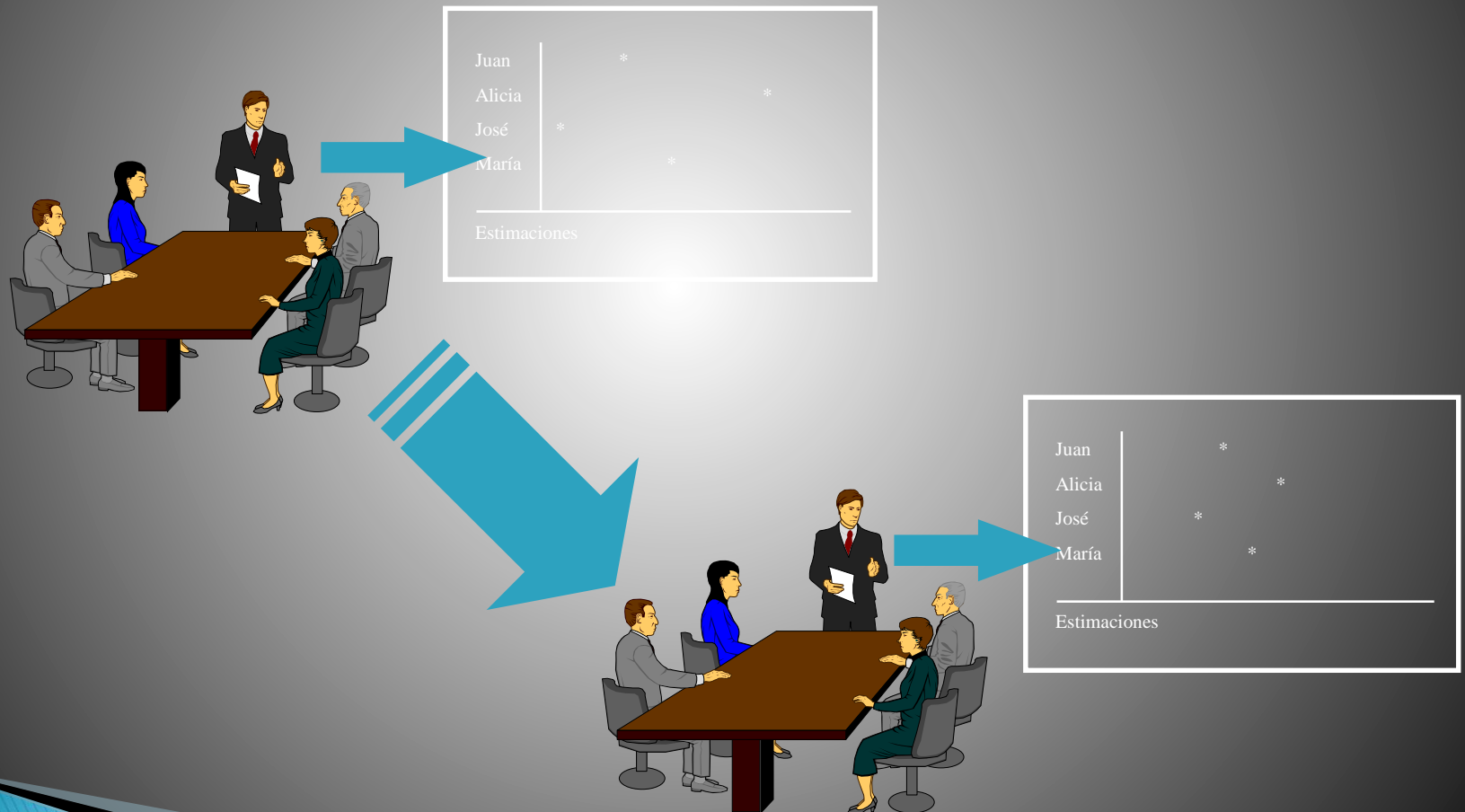
Juicio experto: Wideband Delphi

- ▶ Un grupo de personas son informadas y tratan de adivinar lo que costara el desarrollo tanto en esfuerzo, como su duración.
- ▶ Las estimaciones en grupo suelen ser mejores que las individuales.

Método de trabajo

- ▶ Se dan las especificaciones a un grupo de expertos.
- ▶ Se les reúne para que discutan tanto el producto como la estimación.
- ▶ Remiten sus estimaciones individuales al coordinador.
- ▶ Cada estimador recibe información sobre su estimación, y las ajenas pero de forma anónima.
- ▶ Se reúnen de nuevo para discutir las estimaciones.
- ▶ Cada uno revisa su propia estimación y la envía al coordinador.
- ▶ Se repite el proceso hasta que la estimación converge de forma razonable.

Método de trabajo del Wideband Delphi



Distribución de la utilización de recursos en el ciclo de vida

- ▶ Usualmente las organizaciones tienen una estructura de costes similar entre proyectos.
- ▶ Si en un proyecto ya hemos realizado algunas fases, es de esperar que los costes se distribuyan de manera proporcional.

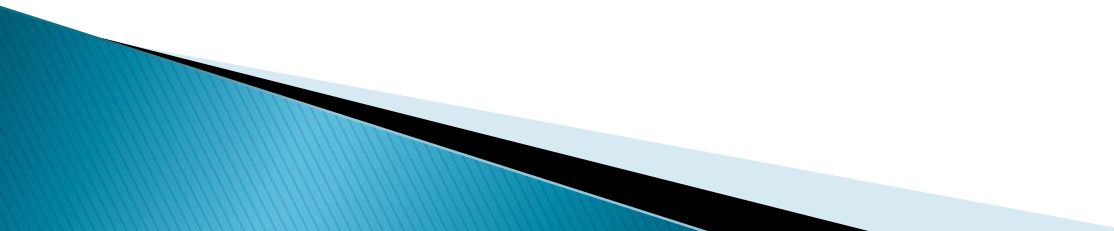
Estudio Viabilidad	Planificación y Requisitos	Diseño General	Diseño Detallado	Desarrollo	Prueba
10%	17%	15%	15%	33%	10%

← 2 m. → ? →

Método basado exclusivamente en los recursos

- ▶ En la estimación consiste en ver de cuanto personal y durante cuanto tiempo se dispone de el, haciendo esa estimación.
- ▶ En la realización:
“El trabajo se expande hasta consumir todos los recursos disponibles” (Ley de Parkinson)

Método basado exclusivamente en el mercado: precio para vender

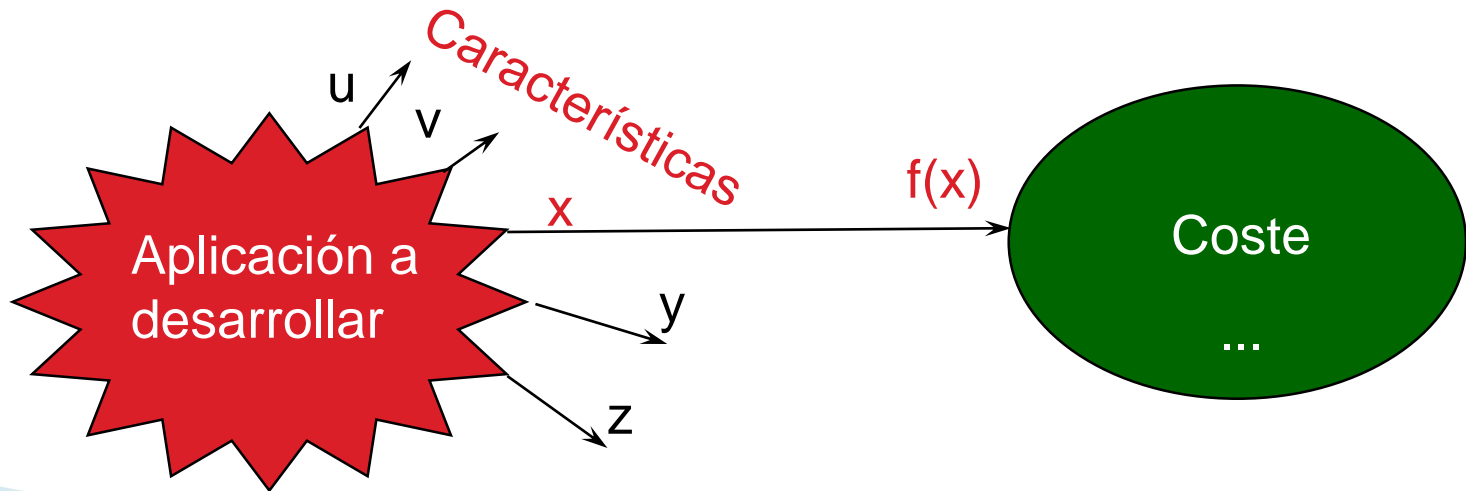
- ▶ Lo importante es conseguir el contrato.
 - ▶ El precio se fija en función de lo que creemos que esta dispuesto a pagar el cliente.
 - ▶ Si se usa en conjunción con otros métodos puede ser aceptable, para ajustar la oferta.
 - ▶ Peligro si es el único método utilizado.
- 

Basado en los componentes del producto o proceso de desarrollo

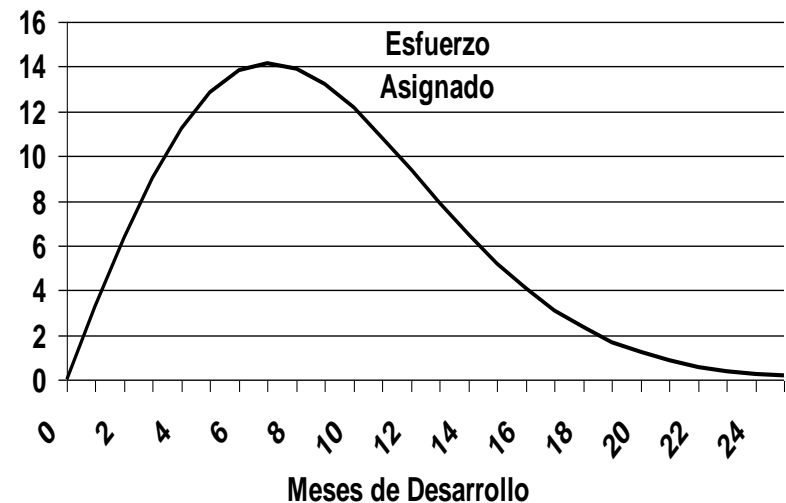
- ▶ Bottom-up
 - Se descompone el proyecto en las unidades lo menores posibles.
 - Se estima cada unidad y se calcula el coste total.
- ▶ Top-Down
 - Se ve todo el proyecto, se descompone en grandes bloques o fases.
 - Se estima el coste de cada componente.

Métodos algorítmicos

- Se basan en la utilización de fórmulas que aplicadas sobre modelos top-down o bottom-up producen una estimación de coste del proyecto

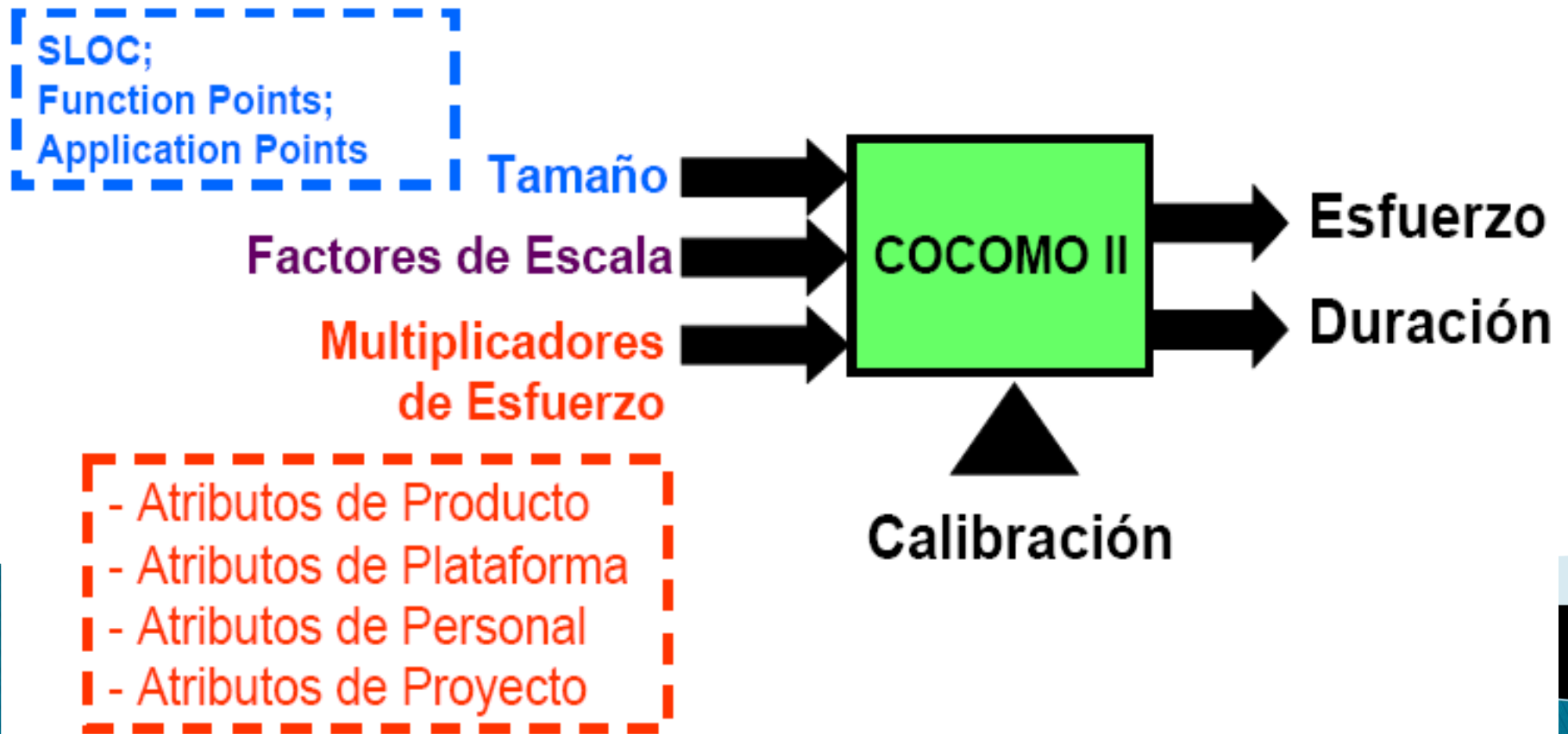


- ▶ Relaciona cantidad de personas-mes y la duración del proyecto.
- ▶ $Y = 2Kate - at^2$
Y = Personas-mes en cada punto
K = Esfuerzo total del proyecto,
(Área bajo la curva)
a = Cte. asociada a la aceleración
de entrada de personas en el
proyecto,
t = instante del tiempo.

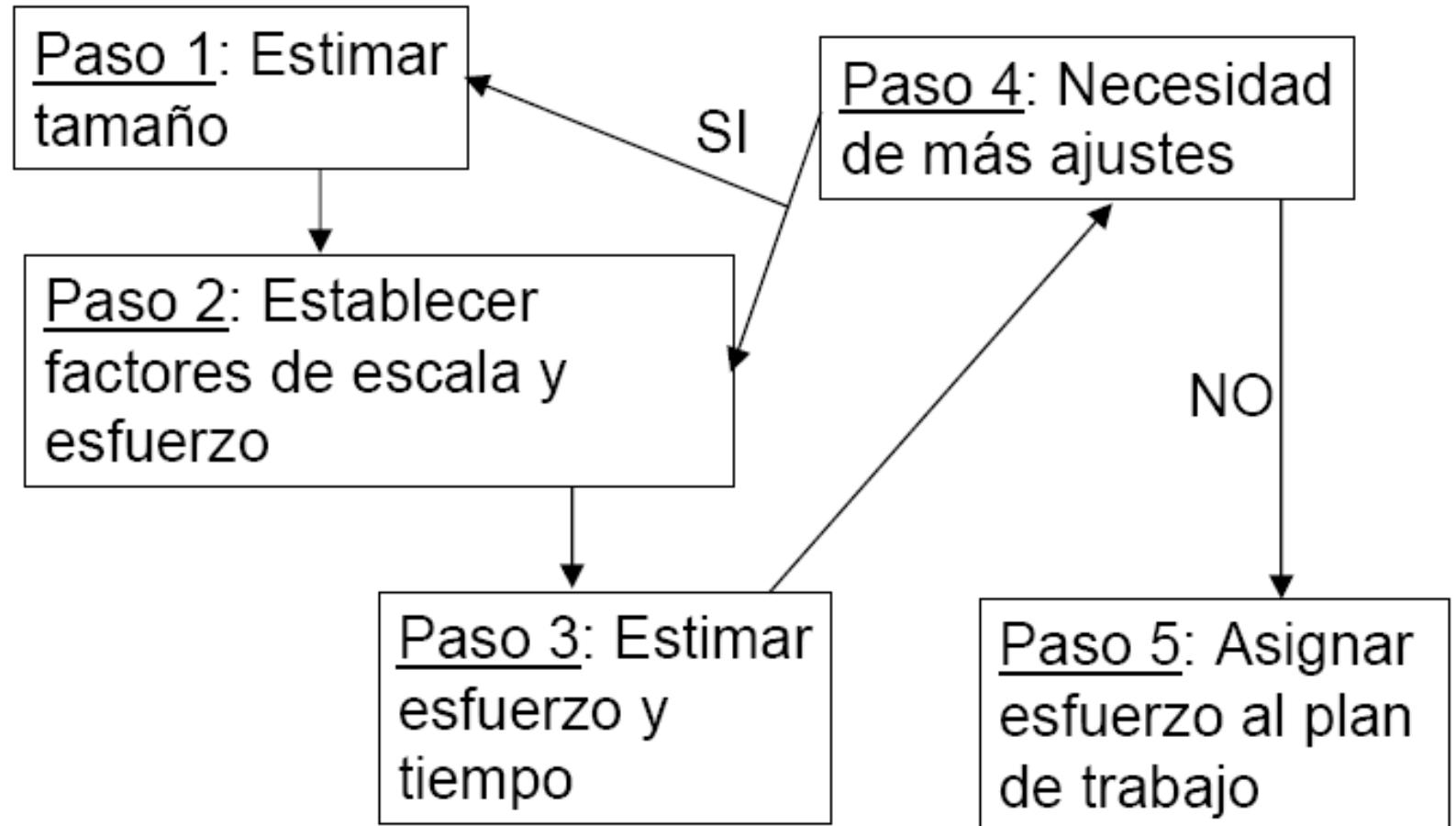


- ▶ Partimos de conocer el número de líneas que tendrá la futura aplicación.
- ▶ Orgánico, hay otros dos
 - $MM\text{-nominal} = 3.2 (KLOC)^{1.5}$
 - $T.\text{desarrollo} = 2.5 (MM)^{0.38}$
- ▶ Determinar los multiplicadores del esfuerzo:
 - Tamaño B.D., experiencia analistas, herramientas, ... (15 en total, varían de 0.75–1.66)
- ▶ Estimación esfuerzo con las correcciones.
- ▶ Estimación de factores relacionados (\$, duración fases,...)

CONCEPTO OPERACIONAL COCOMO II



Proceso de estimación con COCOMO II



Cost drivers y factores de escala de Cocomo II

Atributos del Producto

- ▶ Confiabilidad requerida
- ▶ Tamaño de la base de datos
- ▶ Complejidad del producto
- ▶ Documentación requerida
- ▶ Reuso requerido

Atributos de la Plataforma

- ▶ Limitaciones de tiempo de ejecución
- ▶ Limitaciones de almacenamiento
- ▶ Volatilidad de la plataforma
- ▶ Tiempo de respuesta promedio

Factores de Escala

- ☑ Precedencia (experiencia en aplicaciones similares)
- ☑ Flexibilidad de la especificación
- ☑ Resolución del riesgo
- ☑ Cohesión del grupo
- ☑ Madurez del proceso

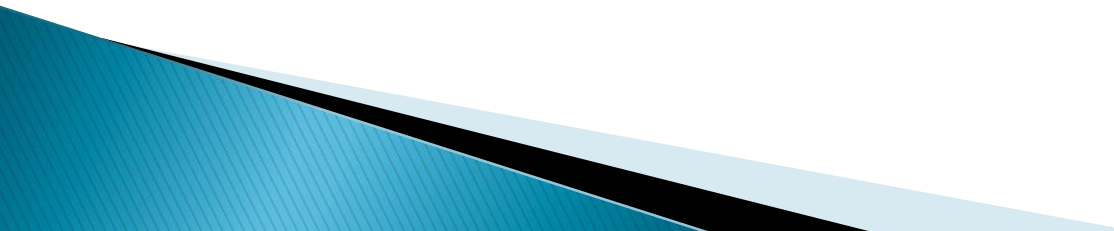
Atributos del Proyecto

- ☑ Prácticas de programación moderna
- ☑ Uso de herramientas de software
- ☑ Limitaciones de tiempo de desarrollo
- ☑ Desarrollo multisitio

Atributos del Personal

- ☑ Capacidad de los analistas
- ☑ Experiencia en aplicaciones similares
- ☑ Capacidad de los programadores
- ☑ Experiencia en la plataforma
- ☑ Experiencia en lenguaje y herramientas
- ☑ Continuidad del personal

Definidos para tres tipos de proyecto

- ▶ **Modo Orgánico:** proyectos pequeños y sencillos, con pocas personas y con buena experiencia en la aplicación.
 - ▶ **Modo semi-acoplado:** proyectos de tamaño y complejidad intermedios, con equipos de distintos niveles de experiencia
 - ▶ **Modo empotrado:** proyectos con fuertes restricciones de hardware, software y funcionales
- 

ECUACIÓN DE CÁLCULO DE ESFUERZO

$$PM_{\text{nominal}} = A \times (\text{Size})^B$$

Donde:

A = constante tomada por defecto del modelo, ajustado en 2.94

Size = tamaño del software (SLOC)

B = factores de escala

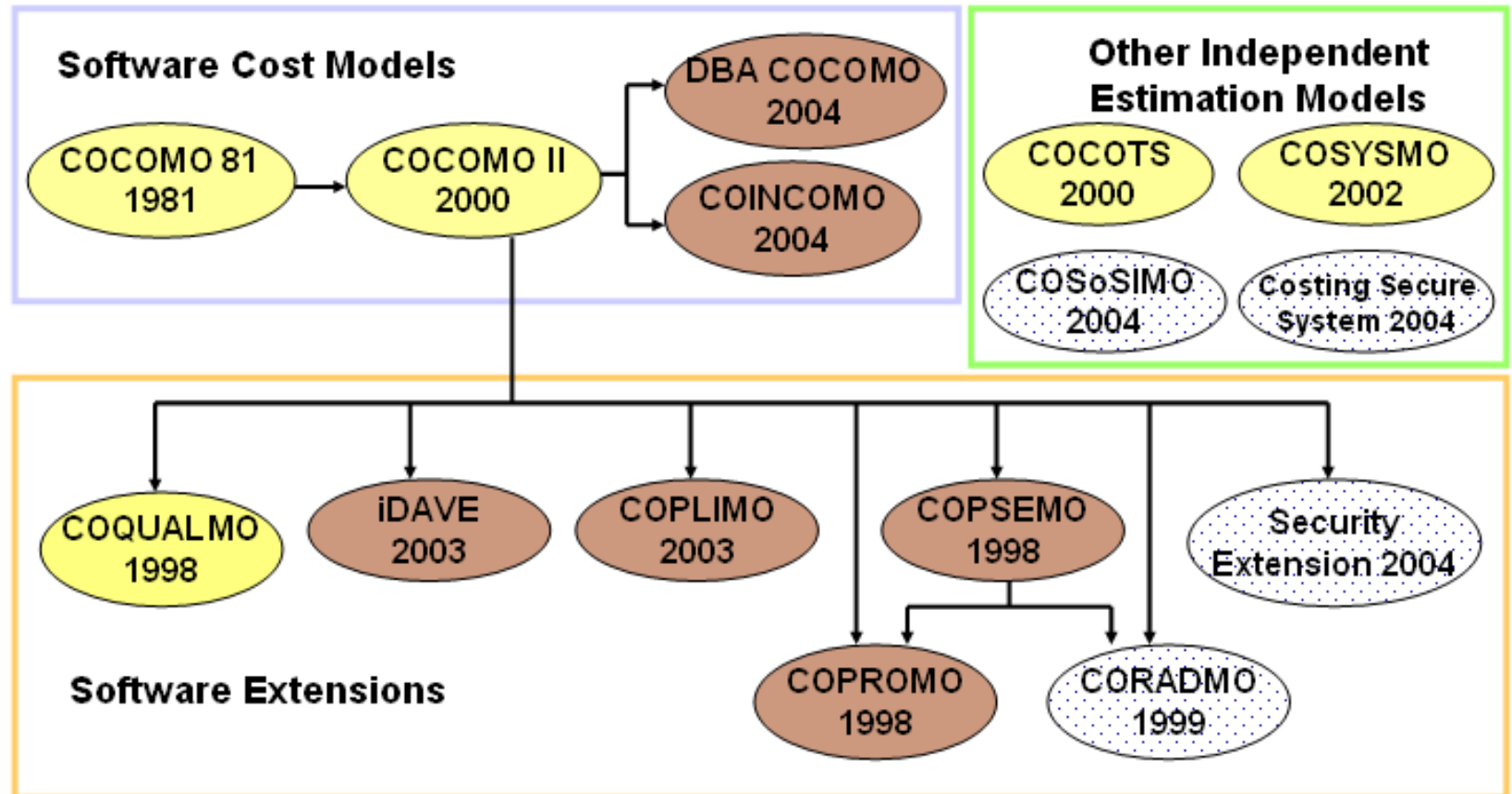
COMO MEDIR EL TAMAÑO DEL SOFTWARE

- ▶ **Líneas de Código (LOC).**
 - Es factible si se tiene código desarrollado.
- ▶ **Puntos de Función (PF)**
 - Permite medir la funcionalidad del sistema desde la perspectiva del usuario.
 - Existen algunos métodos de conteo (Albrecht IFPUG, MKII, NESMA, COSMIC-FFP.)

LIMITACIONES DE COCOMO II

- ▶ Está basado en un ciclo de vida en cascada
- ▶ El modelo NO sirve para proyectos pequeños, esfuerzo < 16 PM, menos de 2 programadores por año .

ALTERNATIVAS DERIVADAS DE COCOMO

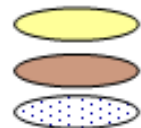


Legend:

Model has been calibrated with historical project data and expert (Delphi) data

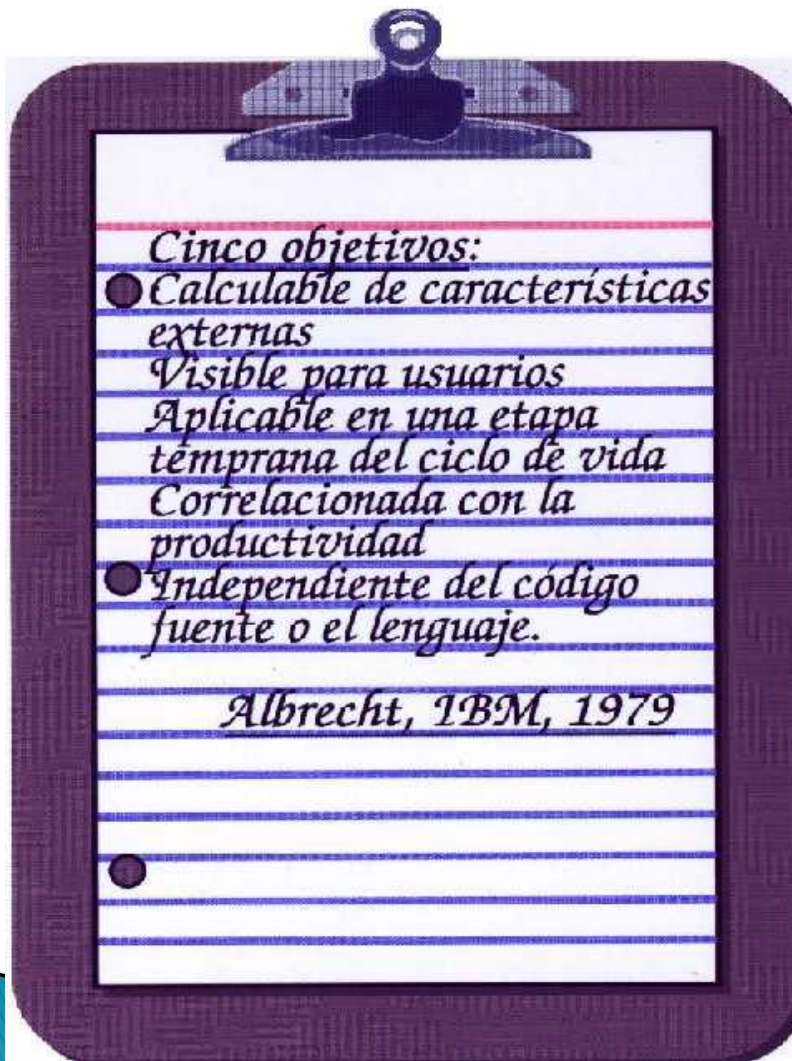
Model is derived from COCOMO II

Model has been calibrated with expert (Delphi) data



Dates indicate the time that the first paper was published for the model

Puntos Función originales



Problemas con LOC

- ✓ No existe definición estándar para todos los lenguajes.
- ✓ Variaciones de tamaño por estilos de programación.
- ✓ Funciones de software liberadas sin producir código.

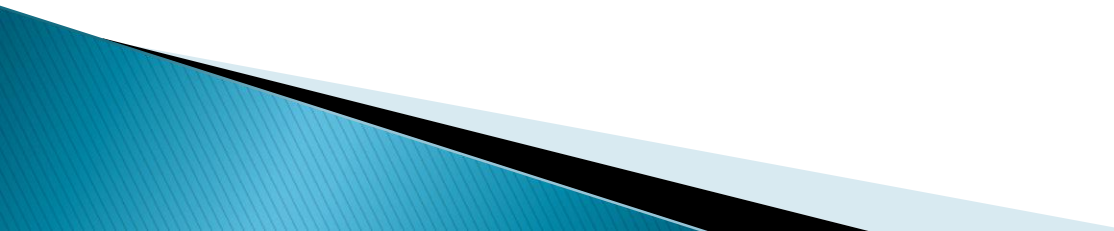
Objetivo

- ✓ Medir la productividad de varios proyectos en distintos lenguajes.

Objetivos de una métrica funcional

1. Facilidades externas, visibles del software.
2. Factores que son importantes para los usuarios.
3. Aplicable temprano en el ciclo de vida.
4. Relación con la productividad económica.
6. Independiente del código fuente o lenguaje. Fácil de aplicar y calcular.
7. Predicción del tamaño de todas las salidas (particularmente del código fuente para cualquier lenguaje).
8. Cálculo reverso sobre software existente (factor de expansión).
9. Proyectos de mantenimiento y mejora.
10. Todo tipo de aplicaciones (incluyendo MIS, software de base, sistemas de tiempo real, software embebido, etc.).
11. Recolección de datos “hard” del proyecto (fechas, personal, esfuerzo, costo, etc.), por medio de un estándar de conteo.
12. Recolección de datos “soft” del proyecto (habilidades, experiencias, métodos, herramientas, etc.), en forma no ambigua.

Métrica de los Puntos de Función

- ▶ Es una métrica que se puede aplicar en las primeras fases de desarrollo.
 - ▶ Se basa en características fundamentalmente “Externas” de la aplicación a desarrollar.
 - ▶ Mide dos tipos de características:
 - Los elementos de función (entradas, salidas, ficheros, etc.)
 - Los factores de Complejidad.
- 

Estimación del Esfuerzo Requerido

Partimos de los datos históricos de la Organización

$\text{Esfuerzo} = \text{PFA} * \text{Promedio (Lenguaje)}$

Nombre Proyecto	Puntos de Función	Lenguaje	Esfuerzo en horas
Sénia	200	COBOL	5.017
Mijares	300	PASCAL	5.410
Palância	150	PASCAL	2.569
Turia	375	4GL	3.011
Albufera	500	PASCAL	9.479
Magro	425	4GL	3.342
Cabriel	800	PASCAL	13.349
Júcar	180	PASCAL	2.800
Serpis	325	4GL	2.541
Montnegre	225	PASCAL	4.528
Vinalopó	310	PASCAL	5.628
Segura	470	COBOL	13.218

- ▶ El método de estimación de costes mediante los puntos de función ha sido denominado FPA o Análisis de Puntos de Función.
- ▶ Este método se basa no en las LDC sino en una métrica que cuantifica la funcionalidad que hay que entregar al usuario al construir una aplicación. Dicha métrica se denomina **puntos de función**.
- ▶ Se caracterizan por:
 - Tener un **componente empírico**, basado en la experiencia de muchos proyectos.
 - Tener **en cuenta la complejidad**, aunque es muy difícil de determinar en un proyecto
 - Ser **independientes del entorno tecnológico** y de las metodologías aplicadas.
 - Utilizar **medidas indirectas**, que se caracterizan por ser subjetivas y difíciles de calcular, sin embargo el resultado obtenido es fácilmente comparable

Definición de Punto Función (octubre 1979)

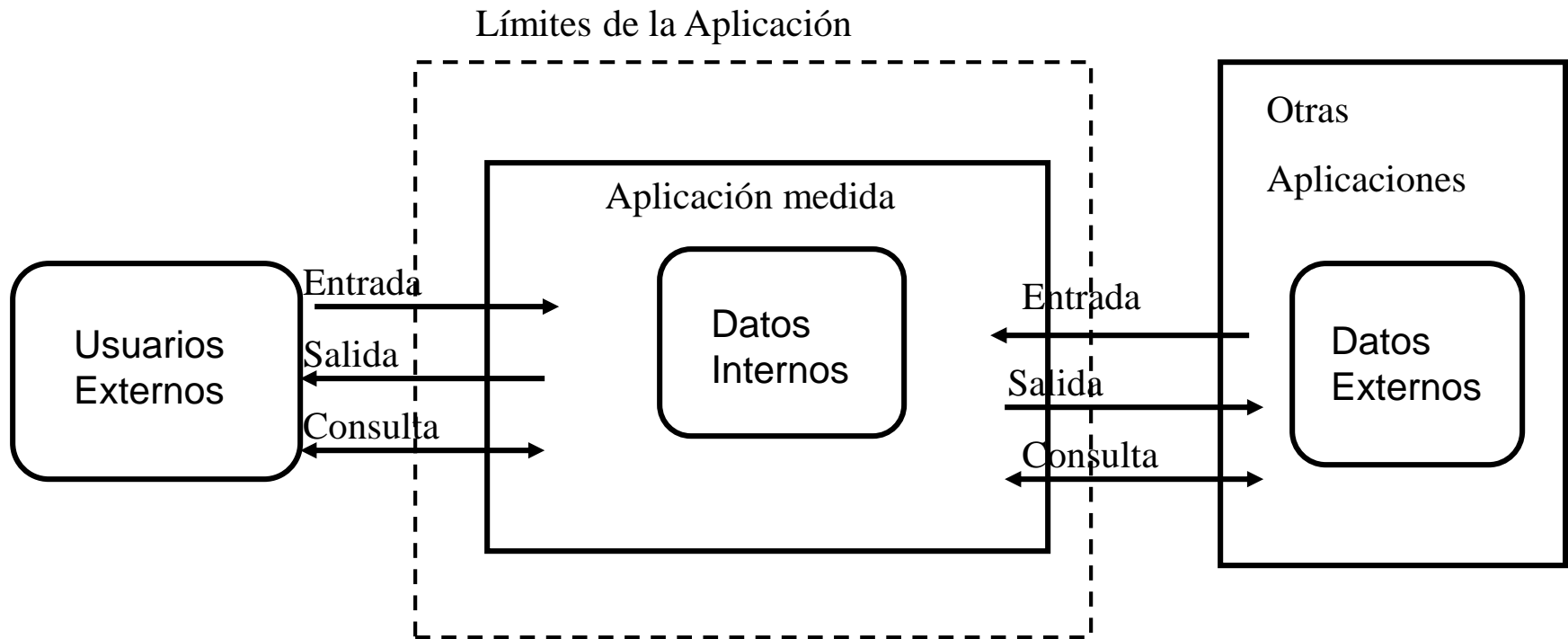
- ✓ Unidad abstracta del “valor económico” de un producto de software
- ✓ No incluye calidad del producto
- ✓ Suma ponderada de cuatro atributos observables:
 - ✓ Nro. de Entradas (x4)
 - ✓ Nro. de Salidas (x5)
 - ✓ Nro. de Consultas (x4)
 - ✓ Nro. de Archivos Maestros (x10)
- ✓ Ajustables por un “factor de complejidad” en + - 25%
- ✓ Problemas: complejidad totalmente subjetiva, rango de variación insuficiente

- Identificación de parámetros y su complejidad. Esto da los PFNA

Parámetros	Cantidad	Complejidad			PF
		<i>simple</i>	<i>media</i>	<i>alta</i>	
<i>Nº de entradas de usuario</i>		x 3	x 4	x 6	
<i>Nº de salidas de usuario</i>		x 4	x 5	x 7	
<i>Nº de peticiones de usuario</i>		x 3	x 4	x 6	
<i>Grupos lógicos de datos internos</i>		x 7	x 10	x 15	
<i>Grupos lógicos de datos externos</i>		x 5	x 7	x 10	
					Σ

- Una vez calculado este valor se debe ajustar a las características del proyecto mediante un factor de complejidad (FA).
- Existen 14 factores que contribuyen a la complejidad de una aplicación.
- Se debe valorar cada uno de ellos dentro de una escala del cero al cinco

Alcance del conteo de PF



Definición: métrica que considera totales ponderados de entradas, salidas, consultas, archivos lógicos e interfaces pertenecientes a una aplicación

Definición de tipos de elementos

- ▶ **Archivos lógicos internos (ALI o ILF)** Grupo de datos lógicos o de información de control, identificables por el usuario, mantenidos a través de procesos elementales de la aplicación dentro de los límites de la misma.
- ▶ **Archivo de interface externa (AIE o EIF)** Grupo de datos lógicos o de información de control, identificables por el usuario, referenciados por la aplicación pero mantenidos a través de procesos elementales de una aplicación diferente.
- ▶ **Entradas externas (EE o EI)** Proceso elemental de la aplicación que procesa datos o información de control que entran desde el exterior del sistema. Los datos procesados mantienen uno o más ALIs, la información de control puede o no mantener un ALI.
- ▶ **Salidas externas (SE o EO)** Es un proceso elemental de la aplicación que genera datos o información de control que es enviado fuera de los límites del sistema. Los datos provienen de uno o más ALIs
- ▶ **Consultas externas (CE o EQ)** Es un proceso elemental de la aplicación que consiste de una combinación de entrada/salida que resulta de un recupero de información desde los ALIs. La parte de la entrada es información de control la cual detalla el requerimiento, especificando qué y/o cómo los datos deben ser recuperados. La parte de la salida contiene los datos recuperados. No se actualizan ALIs durante el proceso.

Definición de tipos de elementos

- ▶ **Tipos elementales de datos (TED o DET)** Son campos/atributos únicos reconocibles por el usuario, no recursivos, y pueden incluir claves foráneas.
- ▶ **Tipos elementales de registros (TER o RET)** Son subgrupos de elementos de datos (mandatorios o no) reconocidos por el usuario, contenidos dentro de los ALIs y de los AIEs.
- ▶ **Tipos de archivos referenciados (TAR o FTR)** Son el total de ALIs mantenidos, leídos o referenciados, y el total de AIEs leídos o referenciados por una transacción de entrada o salida.

Matriz de complejidad (archivos)

Cantidad de grupos lógicos (tipos de registros)	Cantidad de elementos de datos		
	1 - 19	20 -50	>50
1	Baja	Baja	Promedio
2 -5	Baja	Promedio	Alta
> 5	Promedio	Alta	Alta

Matriz de complejidad (entradas externas)

Cantidad de archivos referenciados	Cantidad de elementos de datos		
	1 - 4	5 -15	> 15
1	Baja	Baja	Promedio
2	Baja	Promedio	Alta
> 2	Promedio	Alta	Alta

Matriz de complejidad (salidas externas)

Cantidad de archivos referenciados	Cantidad de elementos de datos		
	1 - 5	6 -19	> 19
1	Baja	Baja	Promedio
2 -3	Baja	Promedio	Alta
> 3	Promedio	Alta	Alta

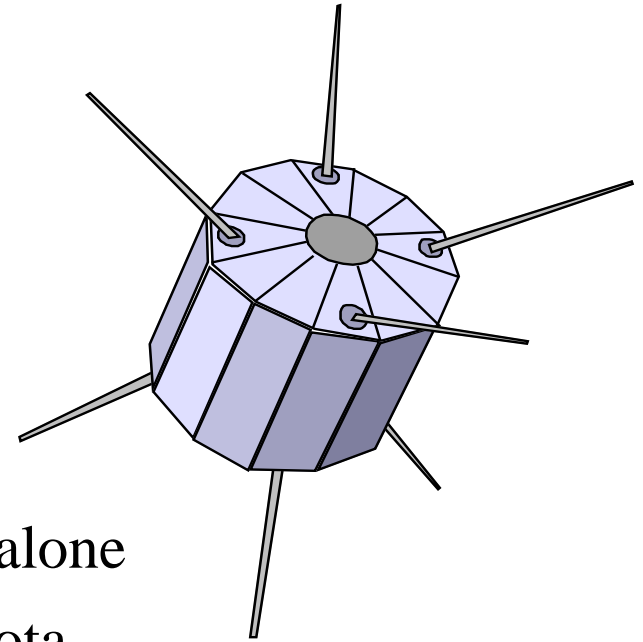
► Factores que contribuyen a la complejidad de una aplicación

Factores de complejidad (FC)	0-5	Factores de complejidad (FC)	0-5
Comunicación de datos		Funciones distribuidas	
Rendimiento		Gran carga de trabajo	
Frecuencia de transacciones		Entrada <i>on-line</i> de datos	
Requisitos de manejo del usuario final		Actualizaciones <i>on-line</i>	
Procesos complejos		Utilización con otros sistemas	
Facilidad de mantenimiento		Facilidad de operación	
Instalación en múltiples lugares		Facilidad de cambio	

Asignación de peso a factores



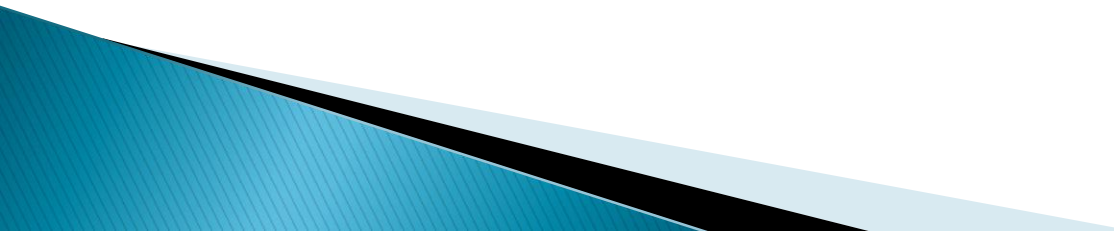
Factores de complejidad







C1 Comunicación de Datos

- 0 Aplicaciones batch o en PC standalone
- 1 Impresión o ingreso de datos remota
- 2 Impresión e ingreso de datos remota
- 3 Front end con teleprocesamiento en la aplicación
- 4 Aplicaciones con teleprocesamiento significativo
- 5 Aplicaciones con teleprocesamiento dominante

C2 Funciones distribuidas

- 0 Aplicaciones totalmente monolíticas
 - 1 Aplicaciones que preparan datos para otras componentes
 - 2 Aplicaciones distribuidas en pocas componentes
 - 3 Aplicaciones distribuidas en más componentes
 - 4 Aplicaciones distribuidas en muchas componentes
 - 5 Aplicaciones dinámicamente ejecutadas en
 muchas componentes
- 

Cómo obtener los puntos función ajustados

-  Sumar los factores (FC)
-  Multiplicar la suma por 0,01
-  Sumarle 0,65
-  Multiplicar por el valor PF "sin ajustar"

$$\text{PF Ajustados} = \text{PF} * ((\text{FC} * 0,01) + 0,65)$$

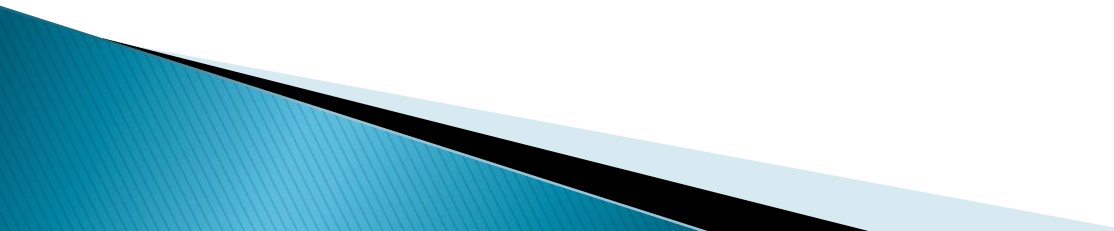
Datos históricos (base de mediciones)

- ▶ **Tamaño**
 - PF y LOC
 - Cálculo directo y cálculo reverso
- ▶ **Estimados**
 - Esfuerzo
 - Duración
- ▶ **Reales (insumidos)**
 - Esfuerzo
 - Duración
- ▶ **Apertura por actividades del ciclo de vida**
 - Requerimientos, desarrollo, implantación (como mínimo)
- ▶ **Staff**
 - Tamaño medio, pico, características del equipo

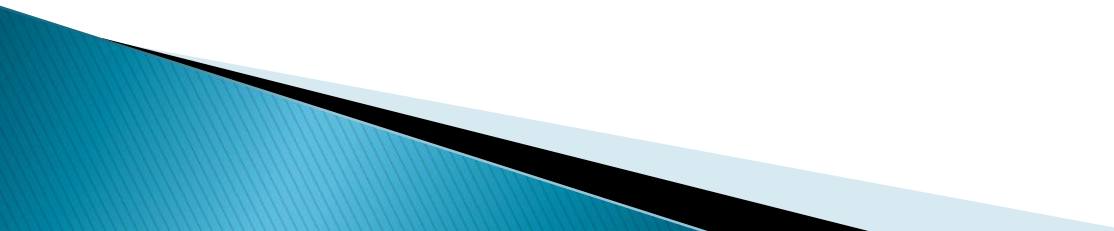
Datos históricos (base de mediciones)

- ▶ Características del proyecto
 - Tipo, Dominio de aplicación, Tecnología
- ▶ Relación tamaño con CU
 - PF por módulo
 - PF promedio por CU
 - PF-UCP
- ▶ Testing
 - Cantidad de casos de prueba diseñados por PF
 - Cantidad de defectos
- ▶ Retrabajo
 - Proporción sobre insumido
 - Por actividad del ciclo de vida

Desarrollar o comprar

- ▶ Especificación de la función y el rendimiento del software y definición de las características que se puedan medir.
 - ▶ Estimación del coste interno de desarrollo y de la fecha de entrega.
 - ▶ Selección de candidatos:
 - Selección de componentes de software reutilizables que puedan ayudar en el desarrollo de la aplicación.
 - Selección de las tres o cuatro aplicaciones que mejor cumplan las especificaciones.
 - ▶ Comparación, una a una, de todas las funciones clave entre los diferentes candidatos.
 - ▶ Evaluación de cada paquete de software o componente según la calidad de productos anteriores, soporte del vendedor, reputación del producto, etc.
 - ▶ Contacto con otros usuarios de dicho software y valoración de las opiniones.
- 

Subcontratación

- ▶ +Ventajas económicas: reducción del número de personas y de la infraestructura necesaria.
 - ▶ +Mayor calidad, por la enorme especialización de las empresas.
 - ▶ –Pérdida de control sobre el software a desarrollar.
- 

Use Case Point

- ▶ Karner, 1993
- ▶ Enfoque semejante a PF
 - Cuenta aspectos claves de los requerimientos para obtener puntos no ajustados
 - Usa varios conjuntos de cuestiones sobre el equipo y su entorno para crear un factor de ajuste
 - Multiplica la cuenta original por el factor para obtener puntos ajustados, que traduce en una estimación de esfuerzo en horas/personas(LOE, *Level of effort*)
- ▶ Ajuste de Kraner
 - Muy parecido a los factores de PF
 - LOE 20 horas /persona por UCP

Use Case Point – Cálculo

- ▶ **1 – Ranquer actores:** Simple (1pto), Medio (2pto), Complejo (3pto)
 - **Simple:** una maquina con una api programable
 - **Medio:** un humano con una interfaz de línea comando o maquina con algún protocolo
 - **Complejo** un humano con una GUI
- ▶ **2 Ranquer caso de usos:** Simple(5pto) Medio(10pto) Complejo (15pto)
 - **Simple:** <4 escenarios claves o caminos de ejecución
 - **Medio** :4 o + escenarios pero < 8
 - **Complejo:** 8 o+ escenarios claves
- ▶ **3 Calcular UCP no ajustados(UUCP) el factor de ajuste y UCP ajustados (AUCP)**
 - Sumar todos los puntos
 - Multiplicar por factor de ajuste técnico y de entorno
- ▶ **4 Convertir el toda AUCP en LOE**
 - Utilizar la calibración propia de su equipo/organización
 - Usar 20 horas persona por AUCP para comenzar
 - Sum 30 hora persona por aucp

Use Case Point- Problemas

▶ **Que es un escenario clave**

- Camino principal que una instancia de un CU puede ejecuta
- Correspondencia con un flujo alternativo principal
- Varios flujos alternativos se combinan en un escenario clave
- Un flujo de excepción particular muy complejo

▶ **Se asume los CU están nivelados**

- Nivel de detalle no demasiado descompuestos, igualmente importancia de un alto nivel
- CU el sistema ,no del negocio

▶ **Los escenarios clave pueden realizar por colaboraciones 7+- 2 clases de análisis**

▶ **Evaluación de complejidad de los casos de uso**

▶ **Granularidad, que es un CU y hasta donde debe refinarse**

El Personal es Decisivo



Experimentos realizados demuestran que el personal tiene diferencias de rendimiento en una proporción de 1:26 ¡o más!

