INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo: "Investigación de modelos de estimación"

Asignatura: Ingeniería de Software

Unidad temática: 1.- Fundamentos

Elaborado por: *Corro Mendoza Onasis Alejandro*

García Hernández Brenda Jacqueline

Herrera Mauricio Pedro Alonso

López de la Rosa Elliot Moisés

Luis Luis Ángel Alexis

Zariñan Gómez Emilio

Docente: Gónzalez Ramírez Marko Alfonso

Fecha: Ciudad de México a 29 de septiembre de 2025

MODELOS DE ESTIMACIÓN

1.- Análisis de punto de función

Los puntos de función miden el tamaño del software basándose en la cantidad de funcionalidad que se entrega al usuario final. Esta métrica ayuda a estimar esfuerzo, costo y tiempo de desarrollo, y comparativamente evaluar la productividad o calidad en proyectos de software.

Los puntos de función fueron propuestos por Allan Albrecht a finales de los años 70 mientras trabajaba en IBM. Él desarrolló esta técnica como un método para medir el tamaño del software basado en la funcionalidad que el software entrega al usuario

Se evalúan cinco componentes funcionales principales:

- Entradas externas (datos que ingresa el usuario)
- Salidas externas (información que el sistema devuelve)
- Consultas externas (interacciones de consulta con datos)
- Archivos lógicos internos (datos almacenados dentro del sistema)
- Interfaces externas (datos compartidos con otros sistemas)

Los puntos de función se aplican mediante un proceso que sigue estos pasos principales:

Identificación de componentes funcionales: Se identifican las funcionalidades del sistema que se van a medir en cinco categorías: entradas externas, salidas externas, consultas externas, archivos lógicos internos y archivos de interfaz externa.

Clasificación de complejidad: Cada componente funcional se clasifica según su complejidad (simple, promedio o compleja) de acuerdo con criterios específicos, asignándoles un valor numérico basado en tablas estándar.

Conteo inicial: Se contabilizan las cantidades de cada tipo de componente con su respectiva complejidad para obtener un conteo total de puntos de función sin ajustar.

Cálculo del factor de ajuste: Se evalúan 14 características generales del sistema (como requerimientos de respaldo, comunicaciones de datos, desempeño, entre otros), asignándoles un peso de 0 a 5 para representar su influencia en la complejidad.

Ajuste del conteo inicial: Se calcula un factor de ajuste basado en la suma ponderada de las características evaluadas, que se utiliza para ajustar el conteo inicial de puntos de función y obtener un valor final.

Uso del valor PF final: Este valor se usa para estimar el esfuerzo, costo y duración del proyecto, así como para predecir el número de errores y líneas de código aproximadas, con base en datos históricos y productividades pasadas.

Los valores de ajuste en el cálculo de puntos de función se determinan mediante la evaluación de 14 características generales del sistema, cada una puntuada en una escala de 0 a 5. Estos valores reflejan la influencia de diversos aspectos técnicos y operativos del sistema en la complejidad del software. Las 14 características y la escala son:

1. Respaldo y recuperación confiables (0 - no importante a 5 - absolutamente esencial)
2. Comunicaciones de datos especializadas
3. Procesamiento distribuido
4. Desempeño crítico
5. Uso de un entorno operativo existente
6. Entrada de datos en línea
7. Entrada de datos en línea con construcción de transacción en múltiples pantallas u operaciones
8. Actualizaciones en línea de archivos lógicos internos
9. Complejidad en entradas, salidas, archivos o consultas
10. Complejidad en procesamiento interno
11. Código diseñado para ser reutilizable
12. Conversión e instalación incluidas en el diseño
13. Diseño para instalaciones múltiples en diferentes organizaciones
14. Diseño para facilitar el cambio y uso por parte del usuario

El uso de puntos de función como métrica de tamaño fue propuesto por Albrecht . La suma de los valores asignados a estas características es utilizada para calcular un factor de ajuste que modifica el conteo total de puntos de función, según la fórmula:

$$PF = conteo\ total \times (0.65 + 0.01 \times \sum Fi)$$

Donde Fi representa cada uno de los valores asignados a las características anteriores

2.- COCOMO I Básico e Intermedio

Boehm introdujo el primer COCOMO en su obra clásica *Software Engineering Economics* [2] en 1981, es un modelo algorítmico ampliamente utilizado para estimar el esfuerzo, el costo y el cronograma de los proyectos de desarrollo de software. El modelo se basa principalmente en el tamaño del proyecto, medido en miles de líneas de código (KLOC), y ajusta las estimaciones utilizando diversos parámetros que describen el entorno del proyecto. Este modelo se estructura en tres niveles para adaptarse a las distintas fases de planificación de un proyecto:

1. Modelo Básico: Ofrece una estimación rápida y aproximada, utilizando únicamente el tamaño estimado del software como entrada principal. Es ideal para estudios de viabilidad y planificación inicial.
2. Modelo Intermedio: Aumenta la precisión al ajustar la estimación del modelo Básico mediante 15 "impulsores de coste" que cuantifican atributos del producto, el hardware, el personal y el proyecto.
3. Modelo Detallado: Incorpora las características del modelo Intermedio, pero aplica el impacto de los impulsores de coste de manera diferenciada a cada fase del proceso de ingeniería de software (análisis, diseño, etc.).

El modelo COCOMO Básico es una herramienta diseñada para obtener una primera aproximación del esfuerzo y el coste de un proyecto de software. Su principal y único dato de entrada es el tamaño estimado del producto, expresado en miles de líneas de código fuente (KLOC). Es más adecuado para proyectos de tamaño pequeño a mediano.

Modos de Desarrollo

Esta clasificación determina el conjunto de constantes empíricas que se utilizarán en las fórmulas de cálculo, reflejando las diferencias de productividad y complejidad entre distintos tipos de proyectos.

- Orgánico: Se aplica a proyectos relativamente pequeños y sencillos, desarrollados por equipos pequeños con una experiencia considerable en aplicaciones similares y en un entorno de trabajo familiar y estable. Los requisitos suelen ser flexibles y el proyecto no presenta grandes innovaciones algorítmicas. El tamaño típico es inferior a 50 KLOC.
- Semiacoplado: Representa un nivel intermedio de complejidad y tamaño. El equipo de desarrollo puede tener una mezcla de personal experimentado y novato. El proyecto combina requisitos tanto rígidos como flexibles y puede interactuar con otros sistemas. El tamaño puede alcanzar hasta 300 KLOC.
- Empotrado: Corresponde a proyectos que operan bajo fuertes restricciones de hardware, software y operativas. Suelen ser complejos, innovadores y con requisitos muy rígidos y difíciles de modificar. La experiencia del equipo en el dominio del problema puede ser limitada.

Fórmulas Fundamentales

El modelo Básico utiliza un conjunto de tres ecuaciones interrelacionadas para estimar el esfuerzo, la duración y el personal necesario:

1. Esfuerzo (E): Calcula la cantidad total de trabajo requerida, medida en Personas-Mes (PM). Un Persona-Mes representa el trabajo que una persona realiza a tiempo completo durante un mes.

$$E = a * (KLDC) * b$$

2. Tiempo de Desarrollo (TDEV): Estima la duración del proyecto en meses calendario, desde el inicio del diseño hasta la finalización de la integración y pruebas.

$$Tdev = c \cdot Ed$$

3. Personal Requerido (P): Calcula el número promedio de personas necesarias para completar el proyecto en el tiempo estimado.

$$P = TdevE$$

Parámetros y Constantes

Los coeficientes (a,b) y los exponentes (c,d) en las fórmulas anteriores no son universales; sus valores dependen directamente del modo de desarrollo seleccionado para el proyecto. Estos valores fueron derivados empíricamente por Boehm a partir de su base de datos de proyectos.

Modo de desarrollo	a	b	c	d
Orgánico	2.4	1.05	2.50	0.38
Semiacoplado	3.0	1.12	2.50	0.35
Empotrado	3.6	1.20	2.50	0.32

Limitaciones del Modelo Básico

La principal fortaleza del modelo Básico, su simplicidad, es también su mayor debilidad. Al basar la estimación únicamente en el tamaño del código, ignora una multitud de factores que influyen significativamente en la productividad y el coste del software. Entre los factores omitidos se encuentran las restricciones de hardware, la experiencia y capacidad del personal, la complejidad del producto y el uso de herramientas de desarrollo modernas. Como resultado, sus estimaciones deben ser consideradas como aproximaciones iniciales y básicas, con un margen de error potencialmente alto. Esta falta de precisión motivó el desarrollo del modelo Intermedio.

El modelo COCOMO Intermedio representa un avance significativo en la precisión de la estimación al complementar la fórmula básica con un análisis detallado del entorno del proyecto. Está diseñado para ser utilizado cuando ya se dispone de un mayor conocimiento sobre los requisitos y las características del sistema, típicamente después

de la fase de especificación. Su innovación clave es la introducción de 15 "impulsores de coste" (cost drivers), que son atributos cualitativos que se cuantifican para ajustar la estimación de esfuerzo nominal.

El Factor de Ajuste del Esfuerzo (EAF)

El corazón del modelo Intermedio es el Factor de Ajuste del Esfuerzo (EAF - Effort Adjustment Factor). Este factor es un multiplicador que se calcula como el producto de los valores asignados a cada uno de los 15 impulsores de coste. El EAF modifica el esfuerzo nominal (calculado de forma similar al modelo Básico) para reflejar cómo las características particulares del proyecto pueden aumentar o disminuir la productividad del equipo. Un EAF mayor que 1.0 indica que el proyecto es más difícil que el promedio, mientras que un EAF menor que 1.0 sugiere que es más sencillo.

Fórmula de Esfuerzo Ajustado

La fórmula del esfuerzo en el modelo Intermedio se modifica para incorporar el EAF. Es crucial señalar que los coeficientes para el cálculo del esfuerzo nominal (a i,b i) en el modelo Intermedio son diferentes de los utilizados en el modelo Básico para los modos orgánico y empotrado. Boehm ajustó estos valores para reequilibrar la ecuación base en torno al efecto multiplicador del EAF.

La fórmula ajustada del esfuerzo es:

$$(E) = a * (KLDC)b * ME$$

Las fórmulas para el Tiempo de Desarrollo (Tdev) y el Personal (P) son las mismas que en el modelo Básico, pero utilizan el valor del esfuerzo ajustado (E) recién calculado.

Tabla 2: Coeficientes para el Esfuerzo Nominal en COCOMO Intermedio

Modo de desarrollo	a	b	c
Orgánico	2.4	1.05	2.50
Semiacoplado	3.0	1.12	2.50
Empotrado	3.6	1.20	2.50

Los 15 impulsores de coste se agrupan en cuatro categorías principales, proporcionando una visión holística de los factores que afectan al desarrollo:

Atributos del Producto:

RELY (Fiabilidad del Software Requerida): Mide el impacto de un fallo del software. Varía desde una simple inconveniencia (muy bajo) hasta un riesgo para la vida humana (muy alto).

DATA (Tamaño de la Base de Datos): Relaciona el tamaño de la base de datos con el tamaño del programa en KLOC.

CPLX (Complejidad del Producto): Evalúa la complejidad de las operaciones, el flujo de control y las estructuras de datos del software.

Atributos del Hardware:

TIME (Restricciones de Tiempo de Ejecución): Porcentaje máximo de la capacidad de la CPU que el software puede utilizar.

STOR (Restricciones de Memoria Principal): Porcentaje máximo de la memoria principal que el software puede ocupar.

VIRT (Volatilidad de la Máquina Virtual): Frecuencia con la que se espera que cambien el hardware y el sistema operativo subyacentes.

TURN (Tiempo de Respuesta Requerido): Rapidez con la que el sistema debe responder a las interacciones, especialmente en entornos interactivos.

Atributos del Personal:

ACAP (Capacidad del Analista): Habilidad, eficiencia y experiencia del equipo de análisis de requisitos.

AEXP (Experiencia en la Aplicación): Experiencia del equipo de desarrollo en el dominio de la aplicación específica.

PCAP (Capacidad del Programador): Habilidad y talento del equipo de programación.

VEXP (Experiencia en la Máquina Virtual): Experiencia del equipo con la plataforma de desarrollo (hardware, SO).

LEXP (Experiencia en el Lenguaje de Programación): Familiaridad del equipo con el lenguaje y las herramientas de programación a utilizar.

Atributos del Proyecto:

MODP (Uso de Prácticas Modernas de Programación): Grado de adopción de prácticas de ingeniería de software, como la programación estructurada.

TOOL (Uso de Herramientas de Software): Nivel de sofisticación y disponibilidad de herramientas de apoyo al desarrollo (ej. herramientas CASE, depuradores avanzados).

SCED (Cronograma de Desarrollo Requerido): Nivel de compresión o relajación del cronograma en comparación con el tiempo nominal estimado por el modelo.

El modelo Intermedio introduce una transformación fundamental en el proceso de estimación: formaliza la incorporación del juicio experto. A diferencia del modelo Básico, que es puramente mecánico, el Intermedio requiere que un gestor de proyectos evalúe cualitativamente cada uno de los 15 impulsores de coste en una escala de seis niveles: Muy Bajo, Bajo, Nominal, Alto, Muy Alto y Extra Alto. El modelo proporciona una tabla de conversión que asigna un multiplicador de esfuerzo numérico a cada una de estas calificaciones subjetivas

	Multiplicadores de esfuerzo (ME)		Valoración					
			Muy bajo	Bajo	Nominal	Alto	Muy alto	Extr. alto
		Atributos del producto						
1.	RELY	Fiabilidad requerida del software	0,75	0,88	1.00	1,15	1,40	
2.	DATA	Tamaño de la base de datos		0,94	1.00	1,08	1,16	
3.	CPLX	Complejidad del producto	0,70	0,85	1.00	1,15	1,30	1,65
		Atributos de la computadora						
4.	TIME	Restricciones del tiempo de ejecución			1.00	1,11	1,30	1,66
5.	STOR	Restricciones del almacenamiento princ.			1.00	1,06	1,21	1,56
6.	VIRT	Inestabilidad de la máquina virtual		0,87	1.00	1,15	1,30	
7.	TURN	Tiempo de respuesta del computador		0,87	1.00	1,07	1,15	
		Atributos del personal						
8.	ACAP	Capacidad del analista	1,46	1,19	1.00	0,86	0,71	
9.	AEXP	Experiencia en la aplicación	1,29	1,13	1.00	0,91	0,82	
10.	PCAP	Capacidad de los programadores	1,42	1,17	1.00	0,86	0,70	
11.	VEXP	Experiencia en S.O. utilizado	1,21	1,10	1.00	0,90		
12.	LEXP	Experiencia en el lenguaje de progr.	1,14	1,07	1.00	0,95		
		Atributos del proyecto						
13.	MODP	Uso de prácticas de programación modernas	1,24	1,10	1.00	0,91	0,82	
14.	TOOL	Uso de herramientas software	1,24	1,10	1.00	0,91	0,83	
15.	SCED	Restricciones en la duración del proy.	1,23	1,08	1.00	1,04	1,10	

3.- COCOMO II Factores de Escala

Según el manual oficial de COCOMO II [1] los factores de escala (Scale Factors, SF) se utilizan para calcular el exponente E en la fórmula del esfuerzo. Este exponente determina si el proyecto presenta economías o deseconomías de escala:

- E < 1.0 ⟶ Economías de escala: el esfuerzo crece menos que proporcional al tamaño.
- E = 1.0 ⟶ Escala lineal: esfuerzo proporcional al tamaño.

- E > 1.0 ⟶ Deseconomías de escala: el esfuerzo crece más que proporcional al tamaño (proyectos grandes tienden a ser más difíciles por coordinación y riesgos)

La fórmula para el exponente es:

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

donde B = 0.91 y los cinco factores de escala determinan el valor final.

2.1.- Los 5 Factores de Escala

2,1.1.- PREC – Precedentedness (Precedencia)

Evalúa qué tanto el proyecto se parece a proyectos anteriores de la organización.

- Muy bajo: el producto es completamente novedoso, sin experiencia previa.
- Muy alto: el producto es casi idéntico a trabajos anteriores.

2.1.2.- FLEX – Development Flexibility (Flexibilidad de Desarrollo)

Mide el grado de rigidez en requerimientos y especificaciones.

- Muy bajo: requisitos rígidos, poco margen de cambio.
- Muy alto: libertad en decisiones técnicas y cambios.

2.1.3.- RESL – Architecture/Risk Resolution (Resolución de Riesgos y Arquitectura)

Considera qué tanto se analizan riesgos y qué tan sólida es la arquitectura antes de iniciar.

- Muy bajo: casi no se identifican riesgos ni se define la arquitectura.
- Muy alto: arquitectura completa y riesgos mitigados tempranamente.

2.1.4.- TEAM – Team Cohesion (Cohesión del Equipo)

Evalúa la cooperación y comunicación entre los miembros del proyecto.

- Muy bajo: interacciones difíciles, choques frecuentes.
- Muy alto: equipo muy cooperativo y alineado.

2.1.5.- PMAT – Process Maturity (Madurez del Proceso)

Basado en el nivel de madurez de procesos de la organización (ej. CMMI).

- Muy bajo: nivel 1 inicial.
- Muy alto: nivel 5 optimizado.

Los factores de escala en COCOMO II representan un elemento crítico para la estimación de esfuerzo, ya que influyen de manera exponencial en el costo y la duración de un proyecto de software. Su importancia radica en que no sólo cuantifican aspectos técnicos, como la precedencia o la madurez del proceso, sino que también capturan dimensiones organizacionales y humanas, como la cohesión del equipo y la gestión de riesgos. En conjunto, estos factores permiten reflejar de manera más realista las economías o deseconomías de escala propias de cada proyecto, ofreciendo una base sólida para mejorar la precisión en la planeación y la toma de decisiones estratégicas.

4.- COCOMO II Multiplicadores de Esfuerzo

En el modelo de Diseño Inicial (Early Design), COCOMO II busca estimar de manera preliminar el esfuerzo de un proyecto cuando todavía no se tienen todos los detalles de arquitectura, requerimientos y equipo.
Por esta razón:

- Se usan menos multiplicadores de esfuerzo (6) que en el modelo Post-Arquitectura (17).
- Se definen de forma agregada, combinando varios de los multiplicadores detallados que se utilizan más adelante.
- La idea es obtener una aproximación realista en etapas tempranas de planeación

Su fórmula general es:

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

4.1.- Los 7 multiplicadores de esfuerzo Diseño Inicial

4.1.1.- RCPX – Product Reliability and Complexity

Agrupa la confiabilidad requerida, la complejidad del software y las interfaces del producto.

- *Muy bajo*: producto simple y tolerante a fallos.
- *Muy alto*: software crítico, complejo, con alta interacción.

4.1.2. RUSE – Required Reusability

Refleja el grado en que el software debe diseñarse para ser reutilizable en futuros proyectos.

- *Nominal*: código para uso puntual.
- *Muy alto*: requiere documentación extensa y generalización.

4.1.3.- PDIF (o PDIF) – Platform Difficulty

Combina las restricciones de tiempo de ejecución, almacenamiento y volatilidad de la plataforma.

- *Muy bajo*: plataforma estable, sin restricciones exigentes.
- *Muy alto*: hardware/software cambiante o con fuertes limitaciones.

4.1.4.- PERS – Personnel Capability

Evalúa las habilidades técnicas del equipo (analistas, programadores, gestores).

- *Muy bajo*: bajo nivel de capacitación o experiencia.

- *Muy alto*: equipo altamente competente.

4.1.5.- PREX – Personnel Experience

Mide la experiencia previa en el dominio de la aplicación, la plataforma y las herramientas.

- *Muy bajo*: equipo inexperto en tecnologías o dominio.
- *Muy alto*: experiencia sólida y directa.

4.1.6.- FCIL – Facilities

Agrupa el uso de herramientas CASE, entornos de desarrollo, infraestructura de comunicación y soporte organizacional.

- *Muy bajo*: herramientas pobres y comunicación limitada.
- *Muy alto*: uso de automatización, integración continua, soporte avanzado.

4.1.7.- SCED – Required Development Schedule

Evalúa si se pide acelerar o extender el cronograma de entrega.

- *Muy bajo*: gran compresión del tiempo (trabajo forzado, menos calidad).
- *Muy alto*: tiempo holgado, sin presiones de calendario.

Los multiplicadores de esfuerzo en el modelo de Diseño Inicial de COCOMO II representan un mecanismo esencial para ajustar las estimaciones en etapas tempranas del proyecto, cuando aún no se dispone de una arquitectura completa. A través de siete dimensiones clave —confiabilidad y complejidad del producto (RCPX), reusabilidad requerida (RUSE), dificultad de la plataforma (PDIF), capacidad del personal (PERS), experiencia del equipo (PREX), facilidades del entorno (FCIL) y compresión del cronograma (SCED)— se logra capturar la influencia real de factores técnicos, organizacionales y de calendario sobre el esfuerzo total. De esta manera, el modelo Early Design no solo permite generar una primera estimación más cercana a la realidad, sino que también sirve como herramienta de análisis para identificar áreas críticas que, si se mejoran, pueden reducir significativamente costos y riesgos en el desarrollo de software.

REFERENCIAS

[1] B. Boehm *et al.*, *COCOMO II Model Definition Manual, Version 2.1*. Center for Software Engineering, USC, 2000.

[2] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.

[3] A. J. Albrecht, "Measuring application development productivity," in *Proc. Joint SHARE/GUIDE/IBM Application Development Symp.*, Monterey, CA, USA, 1979, pp. 83–92.

[4] C. Abts, B. Boehm, and E. B. Clark, *COCOMO II Software Cost Estimation Model*. Upper Saddle River, NJ, USA: Prentice Hall, 2000. *es el libro donde se consolida el modelo.*

[5] S. Chulani, B. Boehm, and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 573–583, Jul. 1999. *es un artículo que lo refina con técnicas estadísticas (Bayes).*

Tabla de porcentaje de aportación

Nombre del Integrante	Porcentaje de Aportación
Corro Mendoza Onasis Alejandro	100%
García Hernández Brenda Jacqueline	100%
Herrera Mauricio Pedro Alonso	100%
López de la Rosa Elliot Moisés	100%
Luis Luis Ángel Alexis	100%
Zariñan Gómez Emilio	100%