Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Análisis de los modelos COCOMO I y II: Estimación de costos aplicada al desarrollo de un sistema bibliotecario

Delgado Pilay, J1, Montalvo Sandoval, S2, Mota Magallanes, R3, Quinde Aspiazu, N4 y Soto Monar, J5

Resumen: El presente artículo expone la aplicación de modelos matemáticos de estimación de costos en el desarrollo de proyectos de software, específicamente en un Sistema de Gestión Bibliotecaria. Se analizan los modelos COCOMO I y COCOMO II, ampliamente reconocidos en ingeniería de software por su utilidad en la planificación de recursos, tiempo y esfuerzo. A través de la estimación práctica con ambos modelos, se comparan factores clave como esfuerzo total, duración del proyecto, personal requerido, productividad esperada y costo económico, considerando un ingreso mensual fijo por persona.

INDEX TERMS COCOMO I, COCOMO II, SISTEMA DE GESTIÓN DE SOFTWARE.

I. INTRODUCCIÓN

El presente documento tiene como objetivo analizar y aplicar dos modelos matemáticos de estimación de costos, esfuerzo y tiempo en el desarrollo de software, específicamente en un Sistema de Gestión Bibliotecaria. En particular, se aborda el uso de los modelos COCOMO I y COCOMO II, ampliamente utilizados en ingeniería de software para respaldar decisiones durante la planificación y ejecución de proyectos.

El modelo COCOMO (Constructive Cost Model), propuesto por Barry Boehm en 1981, es una metodología utilizada para estimar el esfuerzo, costo y tiempo necesarios para desarrollar un proyecto de software. COCOMO I se basa en la cantidad de líneas de código fuente (KDSI o KLOC) como principal métrica para calcular las necesidades del proyecto. Este modelo fue muy útil en su época para proyectos de software tradicionales y sigue siendo una referencia importante en ingeniería de software.

COCOMO I se divide en tres niveles: básico, intermedio y detallado. El nivel básico permite estimaciones rápidas usando solo el tamaño del software; el intermedio incorpora factores de costo adicionales como experiencia del equipo o complejidad del software; y el detallado ofrece un análisis más preciso desglosando el proyecto en módulos. A pesar de su utilidad, este modelo fue limitado con el tiempo, ya que no se adaptaba bien a metodologías modernas de desarrollo ni a proyectos orientados a objetos o con ciclos de vida iterativos.

Para responder a estas nuevas necesidades, Boehm y su equipo desarrollaron COCOMO II en la década de 1990. Este modelo es una evolución del original que incorpora prácticas de desarrollo contemporáneas como el uso de componentes reutilizables, herramientas CASE, desarrollo ágil y enfoques iterativos. Además, COCOMO II emplea estimaciones tempranas basadas en el alcance funcional (como puntos de función) y permite ajustar la estimación conforme el proyecto avanza y se tiene más información.

COCOMO II ofrece más flexibilidad que su predecesor al incluir tres submodelos: el modelo de aplicación (Application Composition Model), el modelo de diseño temprano (Early Design Model) y el modelo de post-arquitectura (Post-Architecture Model), permitiendo realizar estimaciones en distintas fases del ciclo de vida del software. Gracias a estas mejoras, COCOMO II se ha convertido en una herramienta más precisa y adaptable a la complejidad de los proyectos de software actuales.

En el ámbito mundial existen diversos sistemas orientados a la automatización de procesos bibliotecarios. Uno de ellos fue desarrollado por el Departamento de Informática de la Universidad de Cienfuegos, en colaboración con la empresa de tecnología para la defensa XETID. Este sistema permite informatizar la gestión bibliográfica en unidades militares, incluyendo funciones como préstamos y devoluciones, control de inventario, gestión de usuarios y generación de reportes estadísticos [3].

Otro ejemplo es el Sistema de Información Bibliotecario creado por estudiantes de Ingeniería de Sistemas de la Universidad Católica de Colombia. Este sistema automatiza procesos administrativos, gestiona diversos perfiles de usuarios, permite búsquedas en catálogos en línea, y genera informes estadísticos sobre el uso de la biblioteca. También permite a los lectores realizar solicitudes de préstamo a través de la plataforma.

A diferencia de estos sistemas, el proyecto que se presenta en este artículo está orientado exclusivamente a la gestión administrativa por parte del personal autorizado (administradores y bibliotecarios), excluyendo a los lectores de la manipulación directa del sistema. Además, incluye características distintivas como una interfaz intuitiva, un módulo de gestión de sanciones o multas por retraso o daños, y la gestión de cubículos de estudio disponibles en la biblioteca, por último, el proyecto realizado no cuenta con un sistema de facturación debido a que la biblioteca ya cuenta con uno propio y no quiere cambiarlo, siendo este proyecto por así decirlo un sistema administrativo para visualizar cada parte importante de la biblioteca sin recurrir a los libros que se tenía anteriormente. Estas funcionalidades amplían el alcance del sistema, adaptándolo a las necesidades reales de la biblioteca.

El sistema ha sido desarrollado como una aplicación web, utilizando tecnologías como HTML5, CSS3, JavaScript, PHP y MySQL, lo que garantiza su accesibilidad, escalabilidad y facilidad de uso.

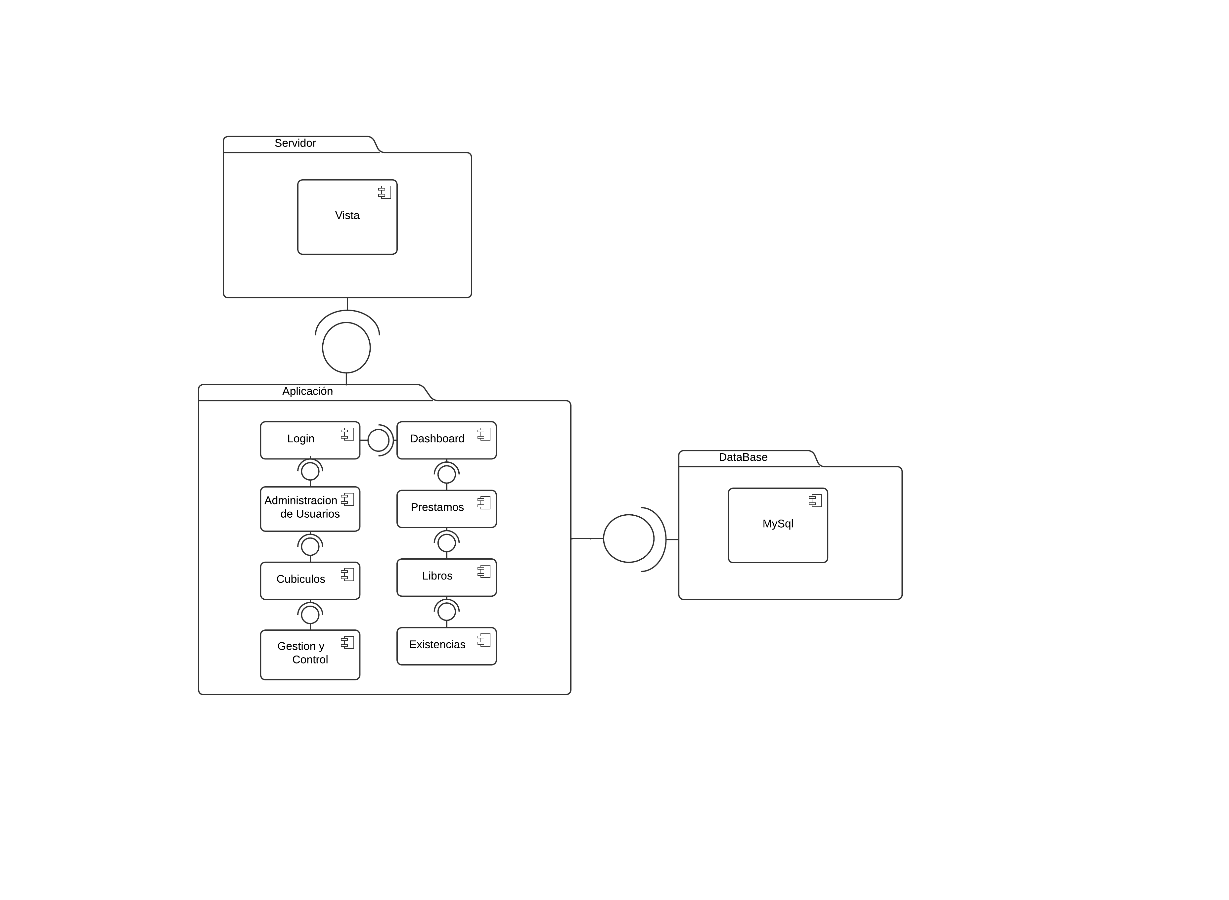
II.  Materials and Methods

Esta sección Proporciona una visión general de cómo está estructurado y diseñado el sistema de Gestión de Biblioteca.

Se incluye una descripción de cada uno de los componentes y las tecnologías seleccionadas para su implementación.

### Arquitectura del Sistema

El sistema está estructurado siguiendo una arquitectura por capas, dividiendo sus funcionalidades en módulos lógicos que facilitan la escalabilidad, el mantenimiento y la reutilización de componentes. La solución ha sido desarrollada como una aplicación web, accesible desde navegadores modernos, y diseñada para su uso exclusivo por parte del personal administrativo de la biblioteca (administradores y bibliotecarios).



**Figura 1.** Diagrama de Componentes del Sistema

### Componentes del Sistema

El proyecto se organiza en tres paquetes principales: Vista, Aplicación, y Base de Datos, como se detalla a continuación:

1. Vista

Corresponde a la capa de presentación, responsable de la interacción con el usuario. Define la interfaz gráfica y controla la visualización de los formularios, paneles de control y pantallas del sistema. Se desarrolló utilizando HTML5, CSS3 y JavaScript, siguiendo principios de diseño intuitivo y accesibilidad.

1. Aplicación

Incluye la lógica de negocio y los controladores de flujo. Este módulo contiene los siguientes subcomponentes:

* Login: Implementa la autenticación de usuarios mediante credenciales validadas contra la base de datos.
* Dashboard: Proporciona navegación entre módulos, accesos directos y resúmenes estadísticos.
* Préstamos: Permite gestionar préstamos de libros, incluyendo creación, modificación, búsqueda, devoluciones y eliminación de registros.
* Libros: Administra el catálogo de libros disponibles en la biblioteca.
* Existencias: Registra copias específicas de libros, sus ubicaciones físicas y su estado actual.
* Administración de Usuarios: Controla los datos de usuarios bibliotecarios y lectores, con funciones CRUD (crear, leer, actualizar, eliminar).
* Cubículos: Administra la disponibilidad, reservas y devoluciones de espacios de estudio (cubículos).
* Gestión y Control: Gestiona sanciones (multas) aplicadas a usuarios lectores por retrasos en devoluciones o daños a materiales, con posibilidad de bloqueo temporal del acceso al sistema.

1. Base de Datos

La persistencia de datos se realiza mediante el sistema gestor de bases de datos MySQL, encargado del almacenamiento de todos los datos relacionados con libros, usuarios, préstamos, cubículos, multas y transacciones del sistema.

## COCOMO I

COCOMO (Constructive Cost Model) es un modelo de estimación propuesto originalmente por Barry Boehm en 1981, diseñado para ayudar en la planificación de costos, cronogramas y recursos necesarios para el desarrollo de software. COCOMO I, la versión inicial del modelo, permite a los equipos de desarrollo estimar con precisión los costos y tiempos de entrega basándose en cronogramas predefinidos, líneas de código y tipo de proyecto. Este modelo también incorpora factores como solicitudes de cambio, compromisos entre costo, funcionalidad y rendimiento, así como decisiones para la mejora de procesos [1].

### Tipos de Proyectos

En el modelo COCOMO, los proyectos de software se clasifican en tres tipos según su complejidad, tamaño y entorno de desarrollo. Estos tipos son:

1. **Orgánico**

Se dice que un proyecto de software es de tipo orgánico si el tamaño del equipo requerido es adecuadamente pequeño, el problema se entiende bien y se ha resuelto en el pasado y también los miembros del equipo tienen una experiencia nominal con respecto al problema.

1. **Semi-desprendido**

Un proyecto de software se dice que es un tipo semi-desprendido si las características vitales tales como el tamaño del equipo, la experiencia y el conocimiento de los diversos entornos de programación se encuentran entre orgánicos e integrados.

Los proyectos clasificados como Semi-Desprendidos son comparativamente menos familiares y difíciles de desarrollar en comparación con los orgánicos y requieren más experiencia, mejor orientación y creatividad. Por ejemplo: Los Compiladores o diferentes Sistemas Incrustados pueden considerarse tipos Semi-Desprendidos.

1. **Empotrado**

Un proyecto de software que requiere el más alto nivel de complejidad, creatividad y requisitos de experiencia cae dentro de esta categoría. Dicho software requiere un tamaño de equipo más grande que los otros dos modelos y también los desarrolladores deben tener la experiencia y la creatividad suficientes para desarrollar modelos tan complejos.

Aquí presento una comparación entre tipos de proyectos

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspectos** | **Orgánico** | **Semi-Acoplado** | **Empotrado** |
| Tamaño del Proyecto | 2 a 50 KLOC | 50-300 KLOC | 300 y por encima de KLOC |
| Complejidad | Bajo | Medio | Alto |
| Experiencia de equipo | Altamente experimentado | Algunos experimentados, así como personal inexperto | Experiencia mixta, incluye expertos |
| Ambiente | Flexible, menos restricciones | Restricciones algo flexibles y moderadas | Requisitos altamente rigurosos y estrictos |

### 2) Tipos de modelo COCOMO

Hay tres tipos de Modelo COCOMO, de los cuales solo vamos a utilizar el Modelo Básico:

**Modelo Básico**

El modelo básico estima el esfuerzo y tiempo de desarrollo en función del tamaño del software (en KLOC, miles de líneas de código).

|  |  |
| --- | --- |
|  | Formula |
| Esfuerzo (E) |  |
| Tiempo de desarrollo (T) |  |
| Personal requerido (P) |  |

Los coeficientes (a, b, c, d) variaran según el tipo de proyecto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de Proyecto** | **a** | **b** | **c** | **d** |
| Orgánico | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-acoplado | 3.0 | 1.12 | 2.5 | 0.35 |
| Empotrado | 3.6 | 1.20 | 2.5 | 0.32 |

## COCOMO II

Posteriormente, y ante la evolución de las metodologías de desarrollo y tecnologías emergentes, surgió COCOMO II, una versión mejorada y adaptada a contextos modernos. Este modelo fue desarrollado en la década de 1990 con el objetivo de estimar proyectos de software más complejos, que utilizan componentes reutilizables, herramientas orientadas a objetos y ciclos de vida iterativos e incrementales [2]. COCOMO II se basa principalmente en el tamaño del software (medido en miles de líneas de código – KLOC – o puntos de función), e incorpora atributos técnicos, del equipo, del producto y del entorno como variables clave. Su enfoque flexible permite clasificar los factores de estimación desde niveles muy bajos hasta extraordinariamente altos, ofreciendo así una herramienta adaptable a diversos contextos de desarrollo.

Para comprender con mayor claridad la evolución y las principales diferencias entre ambas versiones del modelo, se presenta a continuación una tabla comparativa entre COCOMO I y COCOMO II:

Tabla 1. Comparación entre los Modelos COCOMO I y COCOMO II

|  |  |  |
| --- | --- | --- |
| **Aspecto** | **COCOMO I** | **COCOMO II** |
| Año de creación | 1981 | 1997 |
| Desarrollado por | Barry W. Boehm (USC) | Barry W. Boehm y equipo (USC-CSE) |
| Tipo de modelo | Estático, basado en fórmulas empíricas | Basado en submodelos adaptativos según la etapa del proyecto |
| Enfoque principal | Estimación del esfuerzo según el tamaño del software (KLOC) | Estimación flexible basada en KLOC, puntos de función y características del proyecto |
| Etapas del desarrollo que abarca | General para todo el ciclo | Diferentes modelos: Composición de aplicaciones, Diseño temprano, Post-arquitectura |
| Entradas principales | KLOC (líneas de código fuente estimadas) | KLOC, atributos del producto, del personal, de la plataforma y del proyecto |
| Factores de ajuste | 15 multiplicadores de esfuerzo | 17 multiplicadores de esfuerzo + 5 factores de escala |
| Modos de desarrollo | Orgánico, Semi-acoplado, Empotrado | No se limita a modos fijos, más flexible y ajustado al contexto |
| Tecnología moderna considerada | No | Sí (reutilización, programación orientada a objetos, herramientas CASE) |
| Precisión y aplicabilidad | Menor precisión en proyectos modernos | Alta precisión en entornos contemporáneos con metodologías modernas |
| Soporte a reutilización de software | Limitado | Sí, contempla componentes reutilizables desde fases tempranas |
| Uso actual | Académico e histórico | Estándar actual en estimación de costos de software |

Como se aprecia en la tabla anterior, COCOMO II representa una evolución significativa respecto a su predecesor, al incorporar factores de escala, submodelos específicos por etapa de desarrollo y soporte para entornos más dinámicos. Este enfoque resulta especialmente útil en sistemas complejos y con alto grado de reutilización, como el sistema de gestión bibliotecaria desarrollado en el presente estudio.

COCOMO II se estructura en tres submodelos principales, aplicables según la etapa del ciclo de vida del software:

1. Composición de Aplicaciones: Se utiliza en fases iniciales del desarrollo, especialmente cuando se construyen prototipos rápidos a partir de componentes reutilizables o interfaces gráficas.
2. Diseño Temprano: Aplicado cuando ya se dispone de una visión general de la arquitectura del sistema, pero sin detallar todos sus componentes.
3. Post-Arquitectura: Empleado cuando la arquitectura del sistema está completamente definida, incluyendo sus módulos, herramientas de desarrollo y entorno de ejecución.

### Aplicación del Modelo COCOMO I

Con base en los valores de la métrica de estimación de COCOMO I BÁSICO, y usando herramientas como COCOMO I Estimator Tool o plantillas en Excel adaptadas al modelo, se estimó lo siguiente:

Tabla de Resultados Estimados (COCOMO I)

|  |  |
| --- | --- |
| Métrica | Valor Aproximado |
| Esfuerzo total | 10.29 |
| Tiempo de desarrollo (T) | 6.06 |
| Personal requerido (P) | 1.70 |

Nota. Elaboración Propia.

### Aplicación del Modelo COCOMO II

Con el propósito de estimar el esfuerzo requerido para desarrollar este sistema, se utilizó el modelo COCOMO II, específicamente su variante Post-Arquitectura, dado que la arquitectura y componentes del sistema ya estaban completamente definidos.

Variables Consideradas:

* Tamaño estimado del software: Se calculó un total aproximado de 25 KLOC (miles de líneas de código), considerando todos los módulos desarrollados en PHP, HTML, CSS y JavaScript.
* Factores de escala y multiplicadores: Se definieron en función del nivel de experiencia del equipo, la reutilización de componentes, la complejidad del sistema, y la calidad de las herramientas utilizadas.

Herramientas de Apoyo:

* Se utilizaron planillas de cálculo compatibles con COCOMO II (como COCOMO II Estimator Tool) para ingresar los parámetros técnicos del proyecto.
* Además, se consultaron manuales oficiales del modelo publicados por USC-CSE (University of Southern California – Center for Software Engineering) para validar las configuraciones aplicadas.

#### Entorno de Desarrollo

El sistema fue desarrollado en un entorno local con las siguientes herramientas:

* Servidor Web: Apache (XAMPP)
* Lenguajes: HTML5, CSS3, JavaScript, PHP
* Base de Datos: MySQL
* IDE: Visual Studio Code
* Control de Versiones: Git

#### Estimación de Esfuerzo con COCOMO II (Post-Arquitectura)

Para estimar el esfuerzo de desarrollo del sistema de gestión bibliotecaria, se aplicó el modelo COCOMO II en su forma Post-Arquitectura, ya que la estructura del sistema estaba completamente definida.

Se consideraron los siguientes parámetros:

|  |  |  |
| --- | --- | --- |
| Parámetro | Valor Asignado | Justificación |
| Tamaño del roduct (KLOC) | 25 | Basado en líneas de código estimadas para todos los módulos del sistema. |
| Confiabilidad requerida (RELY) | Nominal (1.00) | Nivel estándar de fiabilidad esperado en un sistema administrativo interno. |
| Base de datos (DATA) | Nominal (1.00) | Volumen de datos moderado. |
| Complejidad del roduct (CPLX) | Alta (1.17) | Por el número de módulos y reglas de negocio implementadas. |
| Capacidad del programador (PCAP) | Alta (0.85) | Se asume un equipo con buena experiencia en desarrollo web. |
| Experiencia en lenguaje (LTEX) | Nominal (1.00) | Experiencia media en PHP, JS, y MySQL. |
| Uso de herramientas CASE | Nominal (1.00) | Sin uso intensivo de herramientas automatizadas. |
| Planificación del proyecto | Nominal (1.00) | Cronograma y seguimiento básico. |

Nota: El modelo aplica multiplicadores a estos factores para calcular el esfuerzo en persona-meses y la duración del proyecto.

Con base en los valores anteriores, y usando herramientas como COCOMO II Estimator Tool o plantillas en Excel adaptadas al modelo, se estimó lo siguiente:

Tabla de Resultados Estimados (COCOMO II)

|  |  |
| --- | --- |
| Métrica | Valor Aproximado |
| Esfuerzo total (persona-meses) | 11.6 |
| Duración estimada (meses) | 5.3 |
| Personal promedio necesario | 2-3 personas |
| Productividad esperada | ~2.15 KLOC/persona-mes |

Nota. Elaboración Propia.

### Comparación de COCOMO 1 Y COCOMO II

A continuación, se presenta una comparación entre las estimaciones realizadas con los modelos COCOMO I (modo orgánico, versión básica) y COCOMO II (modelo post-arquitectura). Para calcular el costo de cada modelo, se ha establecido un ingreso mensual por persona de $450. El costo total se obtiene multiplicando el esfuerzo estimado (en persona-mes) por dicho valor.

* COCOMO I: 10.29 persona-mes × $450 = $4,630.50
* COCOMO II: 11.6 persona-mes × $450 = $5,220.00

La siguiente tabla resume los resultados obtenidos:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **COCOMO 1 (Orgánico)** | **COCOMO II (Post-Arquitectura)** |
| **Esfuerzo total (persona-mes)** | 10.29 | 11.6 |
| **Duración total (meses)** | 6.06 | 5.3 |
| **Personal promedio** | 1.7 personas | 2–3 personas |
| **Costo total estimado** | $4,630.50 | $5,220.00 |
| **Productividad (KLOC/persona-mes)** | No estimado | ~2.15 |

En términos generales, COCOMO I presenta una estimación más económica, con menor esfuerzo total y un equipo más pequeño. Sin embargo, la duración del proyecto es ligeramente mayor (6.06 meses). Por otro lado, COCOMO II estima un tiempo de desarrollo más corto (5.3 meses), con una mayor productividad por persona, aunque requiere un equipo más numeroso y representa un costo ligeramente superior.

Conclusión

Aunque el modelo COCOMO I resulta más económico en términos absolutos, su enfoque tradicional no contempla características propias del desarrollo moderno, como el uso de componentes reutilizables, herramientas CASE o enfoques iterativos. Además, no ofrece mecanismos detallados para ajustar estimaciones a medida que el proyecto evoluciona. Esto puede representar una desventaja en proyectos complejos o en contextos que requieren alta adaptabilidad, como el desarrollo de aplicaciones web escalables.

En contraste, COCOMO II se adapta mejor a las necesidades del sistema propuesto, ya que permite realizar estimaciones más precisas en diferentes etapas del ciclo de vida del software y refleja de forma más realista los escenarios actuales de desarrollo. Aunque el costo estimado es ligeramente mayor, la reducción del tiempo de entrega, la mejora en la productividad y la alineación con prácticas modernas justifican plenamente su elección. Por tanto, para el Sistema de Gestión Bibliotecaria desarrollado, COCOMO II representa una herramienta más eficaz y confiable para la planificación del proyecto.

REFERENCES AND FOOTNOTES

A. REFERENCES

# Referencias

|  |  |
| --- | --- |
| [1] | B. Boehm, R. Valerdi, J. A. Lane y A. Winsor Brown, «COCOMO Suite Methodology and Evolution,» de *Software Engineering Technology*, California, 2005, pp. 20-25. |
| [2] | B. Clark, S. Devnani-Chulani y B. Boehm, «alibrating the COCOMO II Post-Architecture model,» de *Actas de la 20ª Conferencia Internacional sobre Ingeniería de Software*, Kyoto, 1998.  [3] B. W. Boehm, C. Abts, S. Chulani, B. Clark, E. Horowitz, R. Madachy y B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall PTR, 2000. ISBN: 0130266922.  Paper IEEE:  [4] J. Smith, A. Johnson, “Application of COCOMO II in Software Cost Estimation,” IEEE Transactions on Software Engineering, vol. 35, no. 2, pp. 123-135, Feb. 2010.  [5] GeeksforGeeks, «Software Engineering | COCOMO Model,» *GeeksforGeeks*, 2023. [En línea]. Disponible: <https://www.geeksforgeeks.org/software-engineering-cocomo-model/>. Accedido: 30 de mayo del 2025. |