Fuente: http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm

El Modelo COCOMO

1. Introducción

El Modelo Constructivo de Costes (Constructive Cost Model) fue desarrollado por B. W. Boehm a finales de los 70 y comienzos de los 80, exponiéndolo detalladamente en su libro "Software Engineering Economics" (Prentice-Hall, 1981). COCOMO es una jerarquía de modelos de estimación de costes software que incluye submodelos *básico*, *intermedio* y *detallado*.

Las ecuaciones de estimación del esfuerzo de desarrollo tienen la forma

$$E = a_i S^{b_i} m(X)$$

con

- S el número de miles de líneas de código fuente
- m(X) es un multiplicador que depende de 15 atributos
- en la siguiente tabla se muestran los coeficientes para los diferentes modos

	Básico		Intermedio	
Modo	a_i	b_i	a_i	b_i
Orgánico	2.4	1.05	3.2	1.05
Semiencajado	3.0	1.12	3.0	1.12
Empotrado	3.6	1.2	2.8	1.2

2. Modelo Básico

Este modelo trata de estimar, de una manera rápida y más o menos burda, la mayoría de proyectos pequeños y medianos. Se consideran tres modos de desarrollo en este modelo: orgánico, semiencajado y empotrado.

2.1. Modo orgánico.

En este modo, un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía de unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles de líneas (medio), mientras que en los otros dos modos el tamaño varía de pequeño a muy grandes (varios cientos de miles de líneas). En este modo, al igual que en los otros, el coste se incrementa a medida que el tamaño lo hace, y el tiempo de desarrollo se alarga.

Se utilizan dos ecuaciones para determinar el esfuerzo de personal y el tiempo de desarrollo. El coste es

$K_m = 2.4\ S_k^{1.05}$

donde K_m se expresa en personas-mes y S_k es el tamaño expresado en miles de líneas de código fuente. El tiempo de desarrollo se da por

$t_d = 2.5\ K_m^{0.38}$

donde K_m se obtiene de la ecuación anterior y t_d es el tiempo de desarrollo en meses. Estas ecuaciones se han obtenido por medio de ajustes de curvas realizado por Boehm en TRW sobre 63 proyectos.

2.2. Modo Empotrado.

En este modo, el proyecto tiene unas fuertes restricciones, que pueden estar relacionadas con el procesador y el interface hardware. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Las estimaciones de tiempo y coste se basan en las mismas ecuaciones que en el modo orgánico, pero con diferentes constantes. Así, el coste se da por

$K_m = 3.6\ S_k^{1.20}$

y el tiempo de desarrollo por

$t_d = 2.5\ K_m^{0.32}$

2.3. Modo Semiencajado.

Es un modo intermedio entre los dos anteriores. Dependiendo del problema, el grupo puede incluir una mezcla de personas experimentadas y no experimentadas.

Fuente: http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm

Las ecuaciones son

$K_m = 3.0\ S_k^{1.12}$

y el tiempo de desarrollo por

$t_d = 2.5\ K_m^{0.35}$.

2.4. Notas al Modelo Básico

Se puede observar que a medida que aumenta la complejidad del proyecto, las constantes aumentan de 2.4 a 3.6, que corresponde a un incremento del esfuerzo del personal. Hay que utilizar con mucho cuidado el modelo básico puesto que se obvian muchas características del entorno.

3. Modelo Intermedio

En este modelo se introducen 15 atributos de coste para tener en cuenta el entorno de trabajo. Estos atributos se utilizan para ajustar el coste nominal del proyecto al entorno real, incrementando la precisión de la estimación.

3.1. Ecuaciones nominales de coste.

Para cada modo de desarrollo, los 15 atributos del coste intervienen como multiplicadores en el coste nominal, K_n, para producir el coste ajustado.

Las ecuaciones nominales de coste para el modelo intermedio son

modo orgánico $K_n = 3.2\ S_k^{1.05}$

modo semiencajado $K_n = 3.0\ S_k^{1.12}$

modo empotrado $K_n = 2.8\ S_k^{1.20}$

Notemos que:

• los exponentes son los mismos que los del modelo básico, confirmando el papel que representa el tamaño;

• los coeficientes de los modos orgánico y empotrado han cambiado, para mantener el equilibrio alrededor del semiencajado con respecto al efecto multiplicador de los atributos de coste.

3.2. Atributos de coste.

Estos atributos tratan de capturar el impacto del entorno del proyecto en el coste de desarrollo. De un análisis estadístico de más de 100 factores que influencian el coste, Boehm retuvo 15 de ellos para COCOMO.

Estos atributos se agrupan en cuatro categorías: atributos del producto, atributos del ordenador, atributos del personal y atributos del proyecto.

(1) Atributos del producto
- RELY: garantía de funcionamiento requerida al software
- DATA: tamaño de la base de datos
- CPLX: complejidad del producto

(2) Atributos del ordenador
- TIME: restricción de tiempo de ejecución
- STOR: restricción del almacenamiento principal
- VIRT: volatilidad de la máquina virtual
- TURN: tiempo de respuesta del ordenador

(3) Atributos del personal
- ACAP: capacidad del analista
- AEXP: experiencia en la aplicación
- PCAP: capacidad del programador
- VEXP: experiencia en máquina virtual
- LEXP: experiencia en lenguaje de programación

(4) Atributos del proyecto
- MODP: prácticas de programación modernas
- TOOL: utilización de herramientas software
- SCED: plan de desarrollo requerido.

Cada atributo se cuantifica para un entorno de proyecto. La escala es

muy bajo -- bajo -- nominal -- alto -- muy alto -- extremadamente alto.

Fuente: http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm

En la tabla se muestran los valores del multiplicador para cada uno de los 15 atributos. Estos 15 valores se multiplican al K_n, y nos proporciona el esfuerzo ajustado al entorno.

3.3. Significado de los atributos.

A. RELY

Indica las posibles consecuencias para el usuario en el caso que todavía existan defectos en el producto. Una puntuación 'muy baja' indica que solamente hace falta eliminar los defectos sin ninguna otra consecuencia.

"very low": el efecto de un fallo del software simplemente trae como consecuencia la inconveniencia de corregir el fallo

"low": el efecto de un fallo software es una pérdida fácilmente recuperable para los usuarios

"nominal": el efecto es una moderada pérdida para los usuarios, pero es una situación de la que se puede salir sin excesiva dificultad

"high": el efecto es una gran pérdida financiera o una inconveniencia masiva humana

"very high": el efecto es una pérdida de vidas humanas.

B. DATA

Indica el tamaño de la base de datos a desarrollar en relación con el tamaño del programa. Tenemos cuatro segmentos con la razón 10-100-1000, que determinan las puntuaciones de 'bajo' a 'muy alto'.

Se define por el cociente

$$\frac{D}{P} = \frac{\text{tamaño de la base de datos en bytes}}{\text{tamaño del programa en DSI}}$$

donde D es la cantidad de datos a ser articulada y almacenada en memoria secundaria (cintas, discos, etc.) hasta el tiempo de entrega del producto software.

C. CPLX

Indica la complejidad de cada módulo y se utiliza para determinar la complejidad compuesta del sistema. Entonces la puntuación puede variar de 'muy bajo' si el módulo está compuesto de expresiones matemáticas simples a 'extremadamente alto' para módulos que utilizan muchos recursos de planificación.

D. TIME

Siempre será más exigente para un programador escribir un programa que tiene una restricción en el tiempo de ejecución. Esta puntuación se expresa en el porcentaje de tiempo de ejecución disponible. Es 'nominal' cuando el porcentaje es el 50%, y 'extremadamente alto' cuando la restricción es del 95%.

E. STOR

Se espera que un cierto porcentaje del almacenamiento principal sea utilizado por el programa. El esfuerzo de programación se incremente si el programa tiene que correr en un volumen menor del almacenamiento principal. STOR captura este esfuerzo extra de 'nominal' cuando la reducción del almacenamiento principal es del 50% a 'extremadamente alto' cuando la reducción es del 95%.

F. VIRT

Durante el desarrollo del software la máquina (hard y soft) en la que el programa se va a desarrollar puede sufrir algunos cambios. VIRT lo refleja desde 'bajo' a 'muy alto'.

G. TURN

Cuantifica el tiempo de respuesta del ordenador desde el punto de vista del programador. Cuanto mayor sea el tiempo de respuesta, más alto será el esfuerzo humano. TURN puede variar desde 'bajo' para un sistema interactivo a 'muy alto', cuando el tiempo medio de respuesta es de más de 12 horas.

H. ACAP

La capacidad del grupo de analistas, en términos de habilidad de análisis, eficiencia y capacidad para cooperar tiene un impacto significativo en el esfuerzo humano. Cuanto más capaz sea el grupo, menos esfuerzo será necesario. ACAP puede variar desde 'muy bajo' a 'muy alto'.

I. AEXP

La experiencia del grupo en una aplicación similar tiene una gran influencia en el esfuerzo. Puede variar desde 'muy bajo' (menos de cuatro meses de experiencia) a 'muy alto' (mayor de 12 años de experiencia).

Fuente: http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm

"very low": ≤ 4 meses experiencia media
"low": 1 año de experiencia media
"nominal": 3 años de experiencia media
"high": 6 años de experiencia media
"very high": ≥ 12 años, o reimplementación de un subsistema

J. PCAP

La cuantificación es similar a la de ACAP, pero en este caso relacionado con los programadores. Se aplica a los programadores como grupo, pero no a los programadores individuales.

K. VEXP

Cuanto mayor sea la experiencia del grupo de programación con el procesador, menor será el esfuerzo necesario. VEXP puede variar desde 'muy bajo', cuando la experiencia es menor de un mes, a 'alto' cuando esta experiencia es mayor de 3 años.

"very low": ≤ 1 mes experiencia media
"low": 4 meses
"nominal": 1 año
"high": ≥ 3 años

L. LEXP

Un grupo de programadores con amplia experiencia en un lenguaje determinado programará de una manera mucho más segura, generando un menor número de defectos y de requerimientos humanos. Puede variar desde 'muy bajo' a 'alto' para un grupo de un mes a tres años de experiencia, respectivamente.

"very low": ≤ 1 mes experiencia media
"low": 4 meses de experiencia media
"nominal": 1 año de experiencia media
"high": ≥ 3 años

M. MODP

Utilización de modernas prácticas de programación. Varía de 'muy bajo' a 'muy alto'. Estas prácticas incluyen, por ejemplo, programación estructurada y desarrollo 'top-down'.

"very low": no se utilizan prácticas modernas de programación -PMP-.
"low": uso experimental de algunas PMP
"nominal": experiencia razonable en el uso de algunas PMP
"high": experiencia razonable en gran parte de PMP
"very high": uso habitual de PMP

N. TOOL

El uso adecuado de herramientas software es un multiplicador de la productividad. La puntuación de TOOL varía desde 'muy bajo' cuando sólo se utilizan herramientas básicas, a 'muy alto' cuando se utilizan herramientas específicas.

O. SCED

El tiempo nominal de desarrollo, tal como se define en el modo básico, es el plazo que requiere menor esfuerzo humano. Cualquier apresuramiento ('muy bajo') o retraso ('muy alto') demandarán más esfuerzo.

3.4. Notas al modelo Intermedio.

Este modelo proporciona una manera potente de capturar la influencia del entorno en el proyecto.

4. Modelo Detallado

Este modelo puede procesar todas las características del proyecto para construir una estimación. Introduce dos características principales

(1) Multiplicadores de esfuerzo sensitivos a la fase. Algunas fases se ven más afectadas que otras por los atributos. El modelo detallado proporciona un conjunto de multiplicadores de esfuerzo para cada atributo. Esto ayuda a determinar la asignación del personal para cada fase del proyecto.

(2) Jerarquía del producto a tres niveles. Se definen tres niveles de producto. Estos son módulo, subsistema y sistema. La cuantificación se realiza al nivel apropiado, esto es, al nivel al que es más susceptible la variación.

Fuente: http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm

4.1. Estimación del esfuerzo.

A. Fases de desarrollo

El desarrollo del software se lleva a cabo a través de cuatro fases consecutivas: requerimientos/planes, diseño del producto, programación y prueba/integración.

Requerimientos/planes. Esta es la primera fase del ciclo de desarrollo. Se analiza el requerimiento, se muestra un Plan de Producto y se general una especificación completa del producto. Esta fase consume del 6% al 8% del esfuerzo nominal K_n, y dura del 10% al 40% del tiempo nominal de desarrollo td. Estos porcentajes dependen del modo y del tamaño (de 2000 LOC a 512000 LOC).

Diseño del producto. La segunda fase del ciclo de desarrollo COCOMO se preocupa de la determinación de la arquitectura del producto y de las especificaciones de los subsistemas. Esta fase requiere del 16% al 18% del esfuerzo nominal K_n, y puede durar del 19% al 38% del tiempo nominal de desarrollo t_d.

Programación. La tercera fase del ciclo de desarrollo COCOMO se subdivide en dos subfases: diseño detallado y prueba del código. Esta fase requiere del 48% al 68% del esfuerzo nominal K_n, y dura del 24% Al 64% del tiempo nominal de desarrollo.

Prueba/Integración. Esta última fase consiste principalmente en unir las diferentes unidades ya probadas. Se utiliza del 16% al 34% del coste nominal K_n y dura del 18% al 34% del t_d.

B. Principio de estimación del esfuerzo.

B.1. *Tamaño equivalente*. Como parte del software puede haber sido ya desarrollado, no se requiere entonces un desarrollo completo. En tales casos se estiman las partes de diseño (D%), código (C%) e integración (I%) a ser modificadas. Se calcula un factor de ajuste A

$A = 0.4\ D + 0.3\ C + 0.3\ I$

El tamaño equivalente, S_{equ} es

$S_{equ} = (S \cdot A) / 100.$

B.2. *Cálculo del esfuerzo*. El tamaño equivalente se calcula para cada módulo. El esfuerzo asignado al desarrollo de cada módulo se obtiene entonces a través de:

(1) seleccionar los valores apropiados de los atributos de coste para cada fase

(2) multiplicar los atributos de coste para cada módulo y fase, obteniendo un conjunto de 4 multiplicadores globales

(3) multiplicar los atributos globales por el esfuerzo nominal en cada fase y sumarlos para obtener el esfuerzo total estimado.

5. Conclusiones sobre COCOMO.

Es uno de los modelos más documentados en la actualidad y es muy fácil de utilizar. Es correcto con referencia a los 63 proyectos utilizados, aunque de ello no se debe desprender que deba ser válido siempre. Una preocupación es la adaptación de las ecuaciones exponenciales a organizaciones específicas, cosa que no parece inmediatamente fácil.

6. Validación independiente del modelo COCOMO (Conte et al.)

El rendimiento de COCOMO en su propia base de datos se indica en la tabla 6.13. El COCOMO Básico no funciona bien en su propia base de datos. El Intermedio lo hace bastante mejor.

Es muy difícil evaluar COCOMO Intermedio en otras bases de datos debido a que hay que proporcionar muchos parámetros. El modelo Básico ha sido evaluado según indica la tabla 6.15, dando unos resultados pobres (quizá debido a la falta de datos). A continuación se indican unas fórmulas mejoradas sobre las originales

$$E = 2.6S^{1.08}m(X) \text{ (modo orgánico)}$$

$$E = 2.9S^{1.12}m(X) \text{ (modo semiencajado)}$$

$$E = 2.9S^{1.20}m(X) \text{ (modo empotrado)}$$