Modelo de Construcción de Costos: **COCOMO**
Ingeniería de Software

Dr. Guillermo Monroy

24-O

1. Introducción

El modelo COCOMO (Constructive Cost Model) fue desarrollado por Barry Boehm a principios de la década de 1980, siendo publicado en su libro "Software Engineering Economics", en el año de 1981.

COCOMO es una jerarquía de 3 modelos incrementales utilizados para estimar costos en un proyecto de Software. Una de las principales motivaciones de Barry Boehm al desarrollar COCOMO consistió en entender el costo de la toma de decisiones hechas en el *comisionamiento*, el *desarrollo* y el *soporte* de un proyecto de Software, ya que explica qué costos se están estimando y el *porqué* se obtienen las estimaciones que se calculan, además de proporcionar capacidades para análisis *trade-off* (intercambio de ventajas por beneficios posteriores) de muchos de los problemas de decisión clásicos en la ingeniería de software.

COCOMO puede ser descrito a su vez como una familia de modelos de algoritmos de costos, y desde su primera publicación, ha sido modificado y actualizado para reflejar las prácticas introducidas por las nuevas tecnologías así como de la *ingeniería cambiante de software*.

De ésta manera, a COCOMO le siguió su predecesor, que inicialmente recibió el nombre de COCOMO 2,0. Sin embargo, después de un debate en cuanto al nombramiento de posteriores versiones, el nombre fue cambiado permanentemente a COCOMO II y fue publicado en el libro "Software Cost Estimation with COCOMO II" (1995). Con el fin de evitar confusiones, el modelo COCOMO original fue designado COCOMO 81 o simplemente COCOMO. Es importante mencionarlo ya que las referencias a COCOMO antes de 1995 se refieren a lo que ahora se llama COCOMO 81, mientras que las mismas a partir de 1995 refieren a COCOMO II.

2. El modelo COCOMO

COCOMO 81 fue un modelo de estimación de costos ampliamente basado en el desarrollo de código original. Es una clasificación consistente de 3 modelos de escala incremental:

1. Un modelo de micro-estimación: **Básico**.
2. Un modelo de estimación media: **Intermedio**.
3. Un modelo de macro-estimación: **Detallado**.

La clasificación se refiere a las escalas en las cuales se categoriza un proyecto en función del tamaño del producto deseado y cada uno posee un conjunto de multiplicadores sensibles a cada fase para cada atributo controlador de costo. A su vez, el modelo de micro estimación y estimación media se encuentran divididos en tres subniveles dependientes del desarrollo del trabajo (*modo de desarrollo*).

Los tres modelos que componen a COCOMO manejan básicamente las mismas ecuaciones descritas a continuación:

$$MM = a * KDSI^b \quad \text{Esfuerzo de desarrollo} \tag{1}$$

$$TDEV = 2{,}5 * MM^c \quad \text{Tiempo de Esfuerzo y Desarrollo} \tag{2}$$

Donde MM es una unidad hombre-mes (man-month), persona-mes (person-month) o equipo-mes (staff-month) y se refiere al esfuerzo mensual de una persona (o unidad de individuo). En COCOMO, hay 152 horas por persona-mes, sin embargo de acuerdo a diferentes organizaciones este valor puede diferir del estándar por 10 % o 20 %. Y finalmente los coeficientes a, b y c dependen del *modo de desarrollo*.

Modos de desarrollo A continuación se muestran los tres modos de desarrollo con las características que a cada proyecto atañen.

Modo de Desarrollo	Características del Proyecto			
	Tamaño	**Innovación**	**Restricciones**	**Ambiente de desarrollo**
Orgánico	Pequeño	Poca	No restringido	Estable
Semi-disjunto	Medio	Medio	Medio	Medio
Embebido	Grande	Alta	Restringido	Hardware complejo

2.1. Modelo Básico

Se trata de un modelo estático basado en un único valor, el cual calcula el esfuerzo y costo del desarrollo del software como una función del tamaño del programa, expresando éste en *millares de instrucciones entregadas* (KDSI, Thousand delivered source instructions).

Se utilizan las Ecuaciones 1 y 2 sin considerar muco detalle en las características del producto, por lo que se utilizan los siguientes valores para los coeficientes a, b y c.

Modo de Desarrollo para el modo **Básico**	a	b	c
Orgánico	2.4	1.05	0.38
Semi-disjunto	3.0	1.12	0.35
Embebido	3.6	1.20	0.32

2.2. Modelo Intermedio

El modelo intermedio calcula el esfuerzo y el costo del desarrollo del software como una función del tamaño del programa y un conjunto de quince "*controladores de costo*" (cost drivers), los cuales incluyen valoraciones objetivas del producto, del hardware, del personal y atributos del proyecto.

Las mismas Ecuaciones 1 y 2 son utilizadas, sin embargo quince nuevos *controladores de costo* son insertados y valorados desde una escala de 'muy baja' a 'muy alta' para con ello calcular el multiplicador del *esfuerzo específico* y cada uno regresa un factor de ajuste el cual multiplicado, entrega en total el *Factor de Ajuste de Esfuerzo* (EAF, Effort Adjustment Factor). El factor de ajuste es 1 para un controlador de costo considerado 'normal'.

En adición al EAF, el parámetro a del modelo de desarrollo es ligeramente distinto del presentado en el modelo básico, mientras que el parámetro b y c permanecen sin cambios como se muestra:

Modo de Desarrollo para el modo **Intermedio**	a	b	c
Orgánico	3.2	1.05	0.38
Semi-disjunto	3.0	1.12	0.35
Embebido	2.8	1.20	0.32

Y ahora la corrección del MM cambia a ser la mostrada en la Ecuación 3:

$$MM_{corregido} = EAF * MM_{nominal} \tag{3}$$

Controladores de Costo Para determinar los *Multiplicadores de Esfuerzo*, se necesita calificar cada uno de los 15 *Atributos de Controladores de Costo* a través de una escala, así como a un conjunto de *Multiplicadores de Esfuerzo* que indiquen cuánto del esfuerzo nominal estimado debe ser multiplicado para contabilizar al proyecto el nivel de atributo calificado.

Los *Atributos de Controladores de Costo* son:

1. **RELY** *Required Software Reliability*. Confiabilidad requerida en el software.
2. **DATA** *Database size*. Tamaño de la base de datos
3. **CPLX** *Product complexity*. Complejidad del producto
4. **TIME** *Execution time constraint*. Restricción en el tiempo de ejecución
5. **STOR** *Main Storage constraint*. Restricción en el almacenamiento principal
6. **VIRT** *Virtual Machine Volatility*. Volatilidad de la máquina virtual. Para algún producto de software dado, la máquina virtual fundamental es el complejo del hardware y el software (Sistema Operativo, Manejador de Base de datos, etc) que éste llama para cumplir sus funciones.
7. **LEXP** *Programming Language Experience*. Experiencia en el lenguaje de programación.
8. **MODP** *Use of Modern programming practices*. Utilización de prácticas modernas de programación.
9. **TOOL** *Use of software tools*. Utilización de herramientas de software.
10. **SCED** *Required development schedule*. Calendarización del desarrollo requerido.

Dichos *Atributos de Controladores de Costo* así como sus correspondientes multiplicadores se muestran en la siguiente Tabla

Controlador de costo	Escala					
	Muy baja	Baja	Nominal	Alta	Muy alta	Extra alta
Atributos de producto						
RELY	0.75	0.88	1.0	1.15	1.40	
DATA		0.94	1.0	1.08	1.16	
CPLX	0.70	0.85	1.0	1.15	1.30	1.65
Atributos de Computadora						
TIME			1.0	1.11	1.3	1.66
STOR			1.0	1.06	1.21	1.56
VIRT		0.87	1.0	1.15	1.3	
TURN		0.87	1.0	1.07	1.15	
Atributos Personales						
ACAP	1.46	1.19	1.0	0.86	0.71	
AEXP	1.29	1.13	1.0	0.91	0.82	
PCAP	1.42	1.17	1.0	0.86	0.70	
VEXP	1.21	1.10	1.0	0.90		
LEXP	1.14	1.07	1.0	0.95		
Atributos de Proyecto						
MODP	1.24	1.10	1.0	0.91	0.82	
TOOL	1.24	1.10	1.0	0.91	0.83	
SCED	1.23	1.08	1.0	1.04	1.10	

Mientras que la Figura 2.2 muestra el sumario de las escalas para cada atributo de controlador de costo (excepto la escala de complejidad, la cual es mostrada en la Figura 2.2.

TABLE VII
COCOMO SOFTWARE COST DRIVER RATINGS

Cost Driver	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
RELY	Effect: slight inconvenience	Low, easily recoverable losses	Moderate, recoverable losses	High financial loss	Risk to human life	
DATA		$\frac{\text{DB bytes}}{\text{Prog. DSI}} < 10$	$10 < \frac{D}{P} < 100$	$100 < \frac{D}{P} < 1000$	$\frac{D}{P} > 100$	
CPLX	See Table VIII	See Table VIII	See Table VIII	See Table VIII	See Table VIII	See Table VIII
Computer attributes						
TIME			< 50% use of available execution time	70%	85%	95%
STOR			< 50% use of available storage	70%	85%	95%
VIRT		Major change every 12 months Minor: 1 month	Major: 6 months Minor: 2 weeks	Major: 2 months Minor: 1 week	Major: 2 weeks Minor: 2 days	
TURN		Interactive	Average turnaround < 4 hours	4–12 hours	>12 hours	
Personnel attributes						
ACAP	15th percentile*	35th percentile	55th percentile	75th percentile	90th percentile	
AEXP	< 4 months experience	1 year	3 years	6 years	12 years	
PCAP	15th percentile*	35th percentile	55th percentile	75th percentile	90th percentile	
VEXP	< 1 month experience	4 months	1 year	3 years		
LEXP	< 1 month experience	4 months	1 year	3 years		
Project attributes						
MODP	No use	Beginning use	Some use	General use	Routine use	
TOOL	Basic microprocessor tools	Basic mini tools	Basic midi/maxi tools	Strong maxi programming, test tools	Add requirements, design, management, documentation tools	
SCED	75% of nominal	85%	100%	130%	160%	

*Team rating criteria: analysis (programming) ability, efficiency, ability to communicate and cooperate.

Figura 1: Escalas de Controladores de Costos

TABLE VIII
COCOMO MODULE COMPLEXITY RATINGS VERSUS TYPE OF MODULE

Rating	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations
Very low	Straightline code with a few nonnested SP* operators: DOs, CASEs, IFTHENELSEs. Simple predicates	Evalution of simple expressions: e.g., A=B+C* (D–E)	Simple read, write statements with simple formats	Simple arrays in main memory
Low	Straightforward nesting of SP operators. Mostly simple predicates	Evaluation of moderate-level expressions, e.g., D = SQRT (B**2–4*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level. No cognizance of overlap	Single file subsetting with no data structure changes, no edits, no intermediate files
Nominal	Mostly simple nesting. Some intermodule control. Decision tables	Use of standard math and statistical routines. Basic matrix/vector operations	I/O processing includes device selection, status checking and error processing	Multi-file input and single file output. Simple structural changes, simple edits
High	Highly nested SP operators with many compound predicates. Queue and stack control. Considerable intermodule control.	Basic numerical analysis (NA) multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns	Operations at physical I/O level (physical storage address translations, seeks, reads, etc.). Optimized I/O overlap	Special purpose subroutines activated by data stream contents. Complex data restructuring at record level
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling	Difficult but structured NA.: near singular matrix equations, partial differential equations	Routines for interrupt diagnosis, servicing, masking. Communication line handling	A generalized, parameter-driven file structuring routine. File building, command processing, search optimization
Extra high	Multiple resource scheduling with dynamically changing priorities. Microcode-level control	Difficult and unstructured NA.: highly accurate analysis of noisy, stochastic data	Device timing-dependent coding, micro-programmed operations	Highly coupled, dynamic relational structures. Natural language data management

*SP structured programming

Figura 2: Escalas de complejidad de módulos COCOMO contra el tipo de módulo

Posteriormente se calcula el *Esfuerzo Estimado de Desarrollo*; éste se obtiene calculando el esfuerzo nominal de desarrollo y multiplicarlo por el producto de los multiplicadores de esfuerzo para los quince atributos de los controladores de costo. Éstos últimos se presentan en la Figura 2.2

Cost Driver	Situation	Rating	Effort Multiplier
RELY	Serious financial consequences of software faults	High	1.15
DATA	20,000 bytes	Low	0.94
CPLX	Communications processing	Very High	1.30
TIME	Will use 70% of available time	High	1.11
STOR	45K of 64K store (70%)	High	1.06
VIRT	Based on commercial microprocessor hardware	Nominal	1.00
TURN	Two-hour average turnaround time	Nominal	1.00
ACAP	Good senior analysts	High	0.86
AEXP	Three years	Nominal	1.00
PCAP	Good senior programmers	High	0.86
VEXP	Six months	Low	1.10
LEXP	Twelve months	Nominal	1.00
MODP	Most techniques in use over one year	High	0.91
TOOL	At basic minicomputer tool level	Low	1.10
SCED	Nine months	Nominal	1.00
Effort adjustment factor (product of effort multipliers)			1.35

Figura 3: Escala de controladores de costo COCOMO

El último paso consiste en determinar los *Factores Estimados Relacionados con el Proyecto*; ya que COCOMO tiene relaciones estimadas de costo adicional para calcular el costo resultante en dolares del proyecto y para el desmembre del costo y esfuerzo por cada fase del ciclo de vida (requerimientos, diseño, etc) y por tipo de actividad de proycto (programación, planeación de tests, administración, etc). Relaciones posteriores apoyan el estimado de la calendarización del proyecto así como la distribución de las fases.

Como se mencionó con anterioridad, COCOMO soporta así mismo los tipos más comunes de análisis de Sensibilidad e intercambio de ventajas por posteriores beneficios (*trade-off*) involucrados en el alcance proyectado del proyecto.

2.3. Modelo Detallado

El último modelo incorpora todas las características anteriormente integradas en el modelo Intermedio , con una evaluación del impacto de los *controladores de costo* en cada etapa (análisis, diseño, ...) del proceso de la ingeniería de software.

Éste modelo calcula el esfuerzo como una función del tamaño del programa y un conjunto de controladores de costo determinado de acuerdo a cada fase del ciclo de vida del software. El modelo Detallado aplica el modelo Intermedio a nivel de componentes y posteriormente utiliza un enfoque basado en las fases para consolidar finalmente el estimado.

Con ello, las 4 fases utilizadas en el modelo Detallado son:

- Planeación de Requerimientos y Diseño de Producto (RPD, *requirements planning and product design*)

- Diseño Detallado (DD, *Detailed design*)
- Prueba de código y unidad (CUT, *code and unit test*)
- Integración y Pruebas (IT, *Integration and test*)

Cada controlador de costo es segmentado en cada fase. Se muestra a continuación un ejemplo:

Controlador de costo	Calificación	RPD	DD	CUT	IT
ACAP	Muy bajo	1.80	1.35	1.35	1.50
	Bajo	0.85	0.85	0.85	1.20
	Nominal	1.0	1.0	1.0	1.0
	Alto	0.75	0.90	0.90	0.85
	Muy alto	0.55	0.75	0.75	0.70

2.4. Ventajas de COCOMO

- COCOMO es transparente, se puede verificar cómo trabaja
- Los controladores son particularmente útiles al estimador para entender el impacto de diferentes factores que afectan los costos del proyecto

El modelo COCOMO ha sido validado con respecto a muestras de 63 proyectos que representan una amplia variedad de negocios, aplicaciones científicas, tiempo real y apoyo a proyectos de software.

2.5. Desventajas de COCOMO

- Es difícil estimar con exactitud los KDSI antes de desarrollar el proyecto cuando la mayoría de las estimaciones son requeridas
- Los KDSI en realidad no son una medida de tamaño, sino de lontigtud
- Es extremadamente vulnerable a una mala clasificación del modo de desarrollo
- Su éxito depende en su mayoría del ajuste del modelo a las necesidades de la organización empleando datos históricos, los cuales no siempre se encuentran disponibles.

3. El modelo COCOMO II

COCOMO II fue desarrollado a partir de COCOMO 81 tomando en cuenta enfoques más modernos de desarrollo de proyectos tales como el rápido desarrollo de programas utilizando lenguajes dinámicos de programación, desarrollo por composición de componentes y utilización de programación de bases de datos. COCOMO II soporta el modelo espiral de desarrollo y produce submodelos que generan estimados incrementalmente detallados.

COCOMO II proporciona tres estapas de modelos para la estimación de proyectos de software:

- **Modelo de composición de aplicaciones**. Utilizado para fases tempranas o ciclos en espiral (prototipado y cualquier otra ocurrencia de prototipos que se puedan presentar posteriormente en el ciclo de vida del proyexto)

- **Modelo de diseño temprano**. Empleado para las fases siguientes o ciclos en espiral. Involucra exploración de alternativas arquitecturales o estrategias de desarrollo de incrementos. El nivel de detalle consiste en niveles de información disponible y del nivel general de la estimación de exactitud requerida en esta etapa.
- **Modelo post-arquitectura**. Una vez que el proyecto está listo para su desarrollo e involucra un campo categorizado, se debería contar con una arquitectura de ciclo de vida, el cual proporcona información más precisa en las entradas de los controladores de costos y permite estimados más exactos.

3.1. Controladores de gastos en COCOMO II

COCOMO II tiene siete de diecisiete factores multiplicativos que determinan el esfuerzo requerido para completar un proyecto de software. Todos los controladores de gastos tienen niveles de escala cualitativos (desde *extra bajo* a *extra alto*) que expresan el impacto del controlador y un conjunto correspondiente de multiplicadores de esfuerzo. El nivel nominal siempre tiene un multiplicador de esfuerzo de 1.0 el cual no cambia el esfuerzo estimado. Así una escala cualitativa del controlador de costo es traducida a una cuantitativa para su uso en el modelo. El modelo COCOMO II puede ser utilizado para calcular el esfuerzo estimado y calendarizar todo el proyecto o para un proyecto que consista de múltiples módulos. El tamaño y escala de los controladores de costos pueden diferir para cada módulo con la excepción de los controladores de costos del *Calendario de Desarrollo Requerido* (SCED, Required Development Schedule) y factores de escala.

En el *Modelo de Diseño Temprano* se utiliza un conjunto pequeño de multiplicadores de controladores de costo , ya que sus controladores son obtenidos de la combinación de los controladores de costo del modelo *Post-arquitectura*.

3.2. Pasos de COCOMO II

Existen en general siete pasos que modelan su desarrollo.

1. Análisis de la Literatura existente
2. Revisión de la literatura de modelado para costos de software
3. Revisión de parámetros significativos potencialmente de formas funcionales
4. Problemas de definición de parámetros (por ejemplo, tamaño)
5. Identificación de parámetros potencialmente nuevos para la *Maduración del Proceso y Desarrollo en Multisitios*
6. Continuación de un conjunto de parámetros provenientes de COCOMO I
7. Obtención de dichos parámetros de COCOMO I como el tiempo de retroceso y prácticas de programación modernas.

3.3. Ventajas de COCOMO II

- COCOMO II es un estándar industrial
- Información muy profunda es fácilmente disponible
- Cuenta con un proceso de calibración claro y efectivo a través de la combinación de delphi con técnicas algorítmicas de estimación de costos (método Bayesiano)
- Compatibilidad de retroceso (backward) con el Rosetta Stone

- Existen diversas extensiones para casi todos los propósitos
- Soporte de la herramienta de software (así como para sus diversas extensiones)

3.4. Desventajas de COCOMO II

- El corazón de COCOMO II está aún basado en el modelo de predicción cascada (predilección)
- La mayoría de las extensiones están aún en su fase experimental y no se encuentran completamente calibradas (al menos hasta el año 2003)
- El cálculo de la duración para proyectos pequeños es irrazonable.

4. Comparativa entre COCOMO'81 y COCOMO II

Las diferencias más importantes entre los dos modelos COCOMO'81 y COCOMO II son:

- COCOMO'81 necesita el tamaño del software en $KDSI$ como una entrada, pero COMOMO II está basado en $KSLOC$ que es código lógico. La diferencia más importante entre DSI y SLOC es que una sola linea de código fuente pueden ser muchas lineas físicas. Para mostrar un ejemplo, una estructura de control *if-then-else* puede ser contada como una SLOC, pero puede ser contada como muchas DSI.
- COCOMO II aborda las siguientes 3 fases de el ciclo de vida de espiral (postulado por el Dr. Boehm): *application development*, *early design* y *post architecture*.
- COCOMO'81 proporciona estimaciones puntuales de esfuerzo y calendarización, pero COMOMO II proporciona rangos probables de estimaciones que representan una desviación estándar alrededor de la estimación más probable.
- El exponente ecuación de estimación se determina por cinco factores de escala (en lugar de los tres modos de desarrollo).
- Los cambios en factores de coste son:
  - factores de costo agregados: DOCU, RUSE, PVOL, PLEX, LTEX, PCON, SITE.
  - factores de costo eliminados: VIRT, TURN, VEXP, LEXP, MODP.
  - modificar las metricas mantenidas para reflejar practicas de software actuales.
- Entradas en COCOMO'81: 63 y COCOMO II:161
- COCOMO II se ajusta para los casos donde existe reutilización del software y reingeniería donde se utilizan herramientas automatizadas para traducción de software existente, pero COCOMO'81 hace solo algunos ajustes para estos factores.
- COCOMO II representa los requisitos de volatilidad en sus estimaciones.

5. Referencias

- MERLO–SCHETT, Nancy; GLINZ, Martin; MUKHIJA, Arun. Seminar on Software Cost Estimation WS 2002/2003.
- BOEHM, Barry W. Software engineering economics. 1981.