

# - COCOMO -

## UN MODELO DE ESTIMACION DE PROYECTOS DE SOFTWARE

Adriana Gómez, María del C.López,  
Silvina Migani, Alejandra Otazú

### RESUMEN

Como se conoce, una de las tareas de mayor importancia en la planificación de proyectos de software es la estimación, la cual consiste en determinar, con cierto grado de certeza, los recursos de hardware y software, costo, tiempo y esfuerzo necesarios para el desarrollo de los mismos.

Este trabajo describe un modelo de estimación, propuesto por Barry Boehm, llamado COCOMO II. Este modelo permite realizar estimaciones en función del tamaño del software, y de un conjunto de factores de costo y de escala. Los factores de costo describen aspectos relacionados con la naturaleza del producto, hardware utilizado, personal involucrado, y características propias del proyecto. El conjunto de factores de escala explica las economías y deseconomías de escala producidas a medida que un proyecto de software incrementa su tamaño.

COCOMO II posee tres modelos denominados *Composición de Aplicación*, *Diseño Temprano* y *Post-Arquitectura*. Cada uno de ellos orientados a sectores específicos del mercado de desarrollo de software y a las distintas etapas del desarrollo de software.

<b>1</b>	<b><i>Introducción</i></b> .....	<b>4</b>
<b>2</b>	<b><i>Breve Historia</i></b> .....	<b>5</b>
<b>3</b>	<b><i>COCOMO 81</i></b> .....	<b>6</b>
	<b>3.1 Modos de Desarrollo</b> .....	<b>7</b>
	<b>3.2 Modelo Básico</b> .....	<b>9</b>
	<b>3.3 Modelo Intermedio</b> .....	<b>12</b>
	<b>3.4 Modelo Detallado</b> .....	<b>14</b>
	3.4.1 Procedimiento de estimación de esfuerzo .....	14
	3.4.2 Procedimiento de estimación del cronograma.....	25
<b>4</b>	<b><i>COCOMO II</i></b> .....	<b>26</b>
	<b>4.1 Definición del modelo</b> .....	<b>26</b>
	<b>4.2 Estimación del Esfuerzo</b> .....	<b>28</b>
	4.2.1 Modelo Composición de Aplicación .....	28
	4.2.2 Modelo Diseño Temprano.....	28
	4.2.3 Modelo Post-Arquitectura .....	30
	<b>4.3 Estimación del Cronograma</b> .....	<b>30</b>
	<b>4.4 Métricas de Software</b> .....	<b>31</b>
	4.4.1 Puntos Objeto .....	31
	4.4.2 Puntos Función.....	32
	4.4.3 Líneas de Código Fuente.....	35
	4.4.4 Conversión de Puntos Función a Líneas de Código Fuente (SLOC).....	36
	4.4.5 Desperdicio de Código (Breakage). .....	37
	4.4.6 Modelo de Reuso.....	37
	4.4.7 Reingeniería y Conversión .....	40
	<b>4.5 Factor Exponencial de Escala</b> .....	<b>42</b>
	4.5.1 Precedencia y Flexibilidad en el Desarrollo (PREC Y FLEX ).....	43
	4.5.2 Arquitectura y Determinación del Riesgo (RESL).....	44
	4.5.3 Cohesión del Equipo (TEAM) .....	45
	4.5.4 Madurez del Proceso (PMAT) .....	45
	<b>4.6 Factores Multiplicadores de Esfuerzo ( Effort Multipliers EM ).</b> .....	<b>46</b>
	4.6.1 Factores del producto .....	48
	4.6.2 Factores de la plataforma .....	50
	4.6.3 Factores del personal.....	51
	4.6.4 Factores del proyecto .....	52
	<b>4.7 Consideraciones destacables del modelo</b> .....	<b>53</b>
<b>5</b>	<b><i>Un Ejemplo Práctico</i></b> .....	<b>54</b>

<b>6</b>	<b><i>Conclusiones</i></b> .....	<b>59</b>
<b>7</b>	<b><i>Anexo I</i></b> .....	<b>60</b>
<b>8</b>	<b><i>Acrónimos y Abreviaturas</i></b> .....	<b>63</b>
<b>9</b>	<b><i>Referencias</i></b> .....	<b>66</b>

# 1 Introducción

Una de las tareas de mayor importancia en la administración de proyectos de software es la estimación de costos. Si bien es una de las primeras actividades, inmediatamente posterior al establecimiento de los requerimientos, se ejecuta regularmente a medida que el proyecto progresa con el fin de ajustar la precisión en la estimación.

La estimación de costos de software tiene dos usos en la administración de proyectos:

- Durante la etapa de planeamiento: Permite decidir cuantas personas son necesarias para llevar a cabo el proyecto y establecer el cronograma adecuado.
- Para controlar el progreso del proyecto: Es esencial evaluar si el proyecto está evolucionando de acuerdo al cronograma y tomar las acciones correctivas si fuera necesario. Para esto se requiere contar con métricas que permitan medir el nivel de cumplimiento del desarrollo del software.

En el ámbito de la ingeniería de software, la estimación de costos radica básicamente en estimar la cantidad de personas necesarias para desarrollar el producto. A diferencia de otras disciplinas de la ingeniería, en las cuales, el costo de los materiales es el principal componente a ser estimado.

La estimación de costos de software posibilita relacionar conceptos generales y técnicas del análisis económico en el mundo particular de la ingeniería de software. Aunque no es una ciencia exacta no podemos prescindir de ella puesto que hoy en día un error en las predicciones puede conducir a resultados adversos.

Es importante reconocer la fuerte relación entre costo, cronograma y calidad. Estos tres aspectos están íntimamente relacionados y confrontados entre sí. De esta manera, se hace difícil incrementar la calidad sin aumentar el costo y/o el cronograma del software a desarrollar. Similarmente, el cronograma de desarrollo no puede reducirse dramáticamente sin deteriorar la calidad del producto de software y/o incrementar el costo de desarrollo. Los modelos de estimación juegan un papel importante ya que permiten equilibrar estos tres factores.

Se han propuesto numerosos métodos de estimación. Entre ellos se pueden contar:

- *Juicio de Expertos*: Este método implica la consulta a expertos, quienes usan su experiencia y conocimiento del proyecto propuesto para lograr una estimación de sus costos.
- *Analogía*: Este método implica una estimación por analogía con proyectos similares, que ya han finalizado, de manera de relacionar los costos reales con la estimación del costo del nuevo proyecto. La principal virtud de la estimación por analogía es que está basada en la experiencia real de un proyecto. Esta experiencia puede ser estudiada para determinar las diferencias específicas con un proyecto nuevo y el impacto de los cambios en los costos. Por otra parte, la principal desventaja es que no está claro hasta que punto es realmente representativo el proyecto previo, en lo que se refiere a restricciones, técnicas, personal y funcionalidad requerida.
- *Parkinson*: Este método intenta adaptar la estimación del costo a los recursos disponibles. En general, es extremadamente inadecuado.
- *Tasar para ganar*: Estima los costos en función del presupuesto adecuado para ganar el trabajo, o el cronograma necesario para estar primero en el mercado con el nuevo producto.

- *Estimación top-down*: A partir de las propiedades globales del producto de software se deriva el costo de todo el proyecto. Después, el costo total es dividido entre las diversas componentes.
- *Estimación bottom-up*: El costo de cada componente de software es estimado por separado, generalmente por la persona responsable del desarrollo de la misma, y luego sumados para obtener el costo total del proyecto. Las técnicas de estimación bottom-up y top-down pueden ser usadas en conjunción con cualquiera de los métodos discutidos en esta sección.
- *Modelos Algorítmicos*: Estos métodos proveen uno o más algoritmos que estiman el costo del software en función de un número de variables que se consideran los principales factores de costo. Los valores de los factores se establecen a partir del análisis de regresión de datos confiables recopilados en proyectos anteriores. Comparados con otros métodos una de sus ventajas es la objetividad, ya que están calibrados a partir de experiencias anteriores. Esto mismo constituye la principal desventaja, por no poder asegurar que estas experiencias sean realmente representativas de proyectos futuros, en especial si se desarrollan en nuevas áreas de aplicación, con nuevas técnicas y arquitecturas. Como sucede en cualquier modelo de estimación, no hay forma de compensar la falta o calidad de los datos de entrada y/o precisión de los valores de los factores de costo. El modelo COCOMO es un ejemplo de modelo algorítmico.

## 2 Breve Historia

El modelo COCOMO ha evolucionado debido a los constantes avances en el mercado de desarrollo de software.

En el año 1981 Barry Boehm publica el modelo COCOMO, acorde a las prácticas de desarrollo de software de aquel momento [Boehm 1981]. Durante la década de los 80, el modelo se continuó perfeccionando y consolidando, siendo el modelo de estimación de costos más ampliamente utilizado en el mundo.

Al aparecer las computadoras personales y generalizarse su uso, surgieron algunas implementaciones. Varias empresas comenzaron a comercializar herramientas de estimación computarizadas.

En el año 1983 se introduce el lenguaje de programación Ada (American National Standard Institute) para reducir los costos de desarrollo de grandes sistemas. Algunos aspectos de Ada provocaron un gran impacto en los costos de desarrollo y mantenimiento, así Barry Boehm y Walker Royce definieron un modelo revisado, llamado Ada COCOMO [Boehm 1989].

En los 90, las técnicas de desarrollo de software cambiaron dramáticamente, surgieron la necesidad de reusar software existente, la construcción de sistemas usando librerías, etc. Estos cambios comenzaron a generar problemas en la aplicación del modelo COCOMO. La solución fue reinventar el modelo. Después de algunos años y de un esfuerzo combinado de USC-CSE (University of Southern California- Center For Software Engineering), IRUS at UC Irvine y organizaciones privadas, aparece COCOMO II. Las incorporaciones a este modelo lo reforzaron e hicieron apto para ser aplicado en proyectos vinculados a tecnologías como orientación a objetos, desarrollo incremental, composición de aplicación, y reingeniería. COCOMO II consta de tres modelos, cada uno de los cuales ofrece una precisión acorde a cada etapa de desarrollo del proyecto. Enunciados en orden creciente de fidelidad son, modelo de *Composición de Aplicación*, *Diseño Temprano* y *Post Arquitectura*.

El USC- CSE implementó los dos últimos modelos en una herramienta de software. Esta herramienta le permite al planificador hacer rápidamente una exploración de las posibilidades de un proyecto, analizando qué efectos provoca el ajuste de requerimientos, recursos y staff sobre la estimación de costos y tiempos.

Para evitar confusión el modelo COCOMO original fue redesignado con el nombre COCOMO' 81. Así todas las referencias de COCOMO encontradas en la literatura antes de 1995 se refieren a lo que ahora llamamos COCOMO'81. La mayoría de las referencias publicadas a partir de 1995 se refieren a COCOMO II.

Existe una nomenclatura para distinguir el modelo teórico, de la implementación, esto es, se denomina COCOMO II al modelo y USC COCOMOII a la herramienta de software. La designación del primer release de la implementación fue USC COCOMO II.1997.0. El componente del año calendario identifica la calibración. Dentro de cualquier año calendario sólo una versión oficial es liberada por USC. Sin embargo, en un mismo año pueden existir más de un release del software, así por ejemplo USC COCOMO II.1997.0 y USC COCOMO II.1997.1 tienen los mismos valores de parámetros sólo los diferencia mejoras incorporadas en la interfase.

### 3 COCOMO 81

COCOMO' 81 está compuesto por tres modelos que corresponden a distintos niveles de detalle y precisión. Mencionados en orden creciente son: *Modelo Básico*, *Intermedio* y *Detallado*. La estimación es más precisa a medida que se toman en cuenta mayor cantidad de factores que influyen en el desarrollo de un producto de software.

COCOMO'81 permite estimar cómo se distribuye el esfuerzo y el tiempo en las distintas fases del desarrollo de un proyecto y dentro de cada fase, en las actividades principales. Las fases consideradas por COCOMO'81 son:

- *Diseño del Producto* (PD)

Se define la arquitectura del hardware, software y las estructuras de datos y control. También se desarrolla un bosquejo del manual del usuario y los planes de aceptación y testeo.

- *Diseño Detallado* (DD)

- *Codificación y Testeo de Unidades* (CT)

En estas dos fases el diseño global de la fase anterior es implementado, creando las componentes de software, que son testeadas y evaluadas individualmente.

- *Integración y Testeo* (IT)

Se fusionan todas las componentes de software desarrolladas con el fin de lograr que el producto de software funcione correctamente. Los requerimientos definidos son usados para controlar las aptitudes del producto liberado.

Los costos y tiempos de las fases excluidas (Requerimientos y Mantenimiento) deben ser estimados en forma separada empleando otros modelos.

Se distinguen las siguientes actividades principales:

- *Análisis de Requerimientos*

Determinación, especificación, revisión y actualización de la funcionalidad, performance e interfase del software

- *Diseño del producto*  
Determinación, especificación, revisión y actualización de la arquitectura de hardware y software, diseño del programa y diseño de la base de datos.
- *Programación (Diseño detallado + Unit Test)*  
Comprende el diseño detallado, codificación, testeos unitarios e integración de las componentes individuales de software. Incluye planificación del personal de programación, adquisición de herramientas, desarrollo de la base de datos, y documentación al nivel de componente.
- *Planificación del Testeo*  
Especificación, revisión y actualización de los planes de testeo del producto y de aceptación del mismo. Adquisición de herramientas de testeo y lotes de datos de prueba.
- *Verificación y Validación*  
Ejecución de la validación y verificación de requerimientos, diseño, testeos del producto y testeos de aceptación. Adquisición de herramientas de V&V. Se hacen las preguntas ¿estaremos construyendo el producto correcto? y ¿estaremos construyendo correctamente el producto?
- *Actividades de oficina*  
Tareas referidas al gerenciamiento del proyecto. Incluye la administración de contratos y subcontratos, relaciones con los clientes, etc.
- *Administración de la Configuración y Aseguramiento de la Calidad (CM/QA):*  
La Administración de la Configuración incluye la identificación del producto, control de cambios, administración de las librerías de soporte, etc. Aseguramiento de la Calidad abarca el desarrollo y monitoreo de estándares, auditorías técnicas del producto de software y el proceso de desarrollo.
- *Manuales:*  
Desarrollo y actualización de los manuales de usuario, operador y mantenimiento.

### 3.1 Modos de Desarrollo

En el modelo COCOMO' 81 uno de los factores más importantes que influye en la duración y el costo de un proyecto de software es el *Modo de Desarrollo*. Todo proyecto corresponde a uno de los siguientes tres modos:

***Modo Orgánico (Organic):*** En esta clasificación se encuentran proyectos desarrollados en un ambiente familiar y estable. El producto a elaborar es relativamente pequeño y requiere pocas innovaciones tecnológicas en lo que refiere a algoritmos, estructuras de datos e integración de hardware.

La mayoría de las personas conectadas con el proyecto tienen gran experiencia en sistemas relacionados dentro de la organización, y un entendimiento acabado de cómo el sistema contribuirá a los objetivos de la organización. Esto significa que todo el equipo de desarrollo podrá contribuir en las etapas iniciales del proyecto sin generar confusión en las comunicaciones debido a que todos conocen que tarea deben realizar.

Además, un proyecto clasificado dentro del modo orgánico, es relativamente flexible en el cumplimiento de los requerimientos, especificaciones de interfase y tiempos de entrega.

Muy pocos proyectos de modo orgánico han desarrollado productos con más de 50 KSLOC<sup>1</sup> de nuevo software. En los casos de productos más extensos, han sido construidos frecuentemente a partir de software existente.

Estas características permiten decir que los proyectos que se encuentran en este modo tienen una gran productividad y una pequeña deseconomía<sup>2</sup> de escala.

Ejemplos de software que se encuentran bajo esta clasificación son:

- Modelos de negocios
- Modelos científicos
- Sistemas operativos de pequeña escala

**Modo Semiacoplado (Semidetached):** Es un modelo para productos de software de tamaño y complejidad media. Las características de los proyectos se consideran intermedias a las de los modos *Orgánico* y *Empotrado*. Esto implica:

Que el equipo de desarrollo:

- Tiene un nivel intermedio de experiencia y conocimiento del sistema en desarrollo.
- Está conformado por algunas personas con vasta experiencia y otras inexpertas en el campo de aplicación.
- Está constituido por personas con amplios conocimientos sólo en algunos aspectos.

Con respecto al cumplimiento de especificaciones de interfase y funcionalidad:

- Son sistemas que presentan niveles variados de exigencia, algunas interfases rigurosas (auditadas por el gobierno) y otras interfases muy flexibles (mensajes de display al operador).

Los productos tienen un tamaño que llega a 300 KSLOC.

Ejemplos de software que se encuentran en esta clasificación son:

- Sistemas de control de producción
- Sistemas de procesamiento de transacciones
- Administradores de Bases de Datos

**Modo Empotrado (Embedded):** En esta clasificación están incluidos proyectos de gran envergadura que operan en un ambiente complejo con altas restricciones de hardware, software y procedimientos operacionales, tales como los sistemas de tráfico aéreo.

Se espera que el software no sólo conforme las especificaciones sino también que sea estable frente a cambios y dificultades producidas en el ambiente. Es decir, estos proyectos no tienen opción de negociar cambios y/o arreglos provocados por modificaciones en los requerimientos y/o en las especificaciones de interfase. El proyecto dedica un gran esfuerzo para adaptarse a los cambios y arreglos, en asegurar que el software cumpla verdaderamente las especificaciones y que los cambios se efectúen correctamente. Esto implica altos costos en los procesos de Verificación y Validación y en la Administración de la Configuración, contribuyendo así a la disminución de la productividad y al aumento de las deseconomías de escala en grandes proyectos.

---

<sup>1</sup> KSLOC es una unidad de medida, donde 1 KSLOC equivale a mil líneas de código fuente

<sup>2</sup>Concepto tratado en la sección 4.5, página 40.

El equipo que interviene en el proyecto tiene un conocimiento general de los objetivos del mismo y una moderada experiencia en el tema. En general, el líder del proyecto destina en las primeras etapas a un grupo pequeño del equipo a las tareas de análisis, evitando así los problemas acarreados por la sobrecarga de comunicaciones. Una vez que se completa el diseño global del producto, la mejor estrategia es delegar a un gran equipo de programadores las tareas de diseño detallado, codificación y testeo.

Ejemplos de software que se encuentran en esta clasificación son:

- Sistemas complejos de procesamiento de transacciones
- Sistemas operativos de gran escala

### 3.2 Modelo Básico

El *Modelo Básico* de COCOMO'81 estima el esfuerzo y el tiempo empleado en el desarrollo de un proyecto de software usando dos variables predictivas denominadas factores de costo (cost drivers): el tamaño del software y el modo de desarrollo. Las ecuaciones básicas son:

*Esfuerzo:*

$$PM = A \times (KSLOC)^B$$

Donde:

- *PM* es el esfuerzo estimado. Representa los meses-persona<sup>3</sup> necesarios para ejecutar el proyecto
- *KSLOC* es el tamaño del software a desarrollar en miles de líneas de código
- *A* y *B* son coeficientes que varían según el Modo de Desarrollo (*Orgánico, Semiacoplado, Empotrado*)

*Cronograma:*

$$TDEV = C \times (PM)^D$$

Donde:

- *TDEV* representa los meses de trabajo que se necesitan para ejecutar el proyecto
- *C* y *D* son coeficientes que varían según el Modo de Desarrollo (*Orgánico, Semiacoplado, Empotrado*)

La Tabla 1 muestra la variación de la fórmula de estimación de esfuerzo y cronograma para los tres Modos de Desarrollo:

Modo de Desarrollo	Esfuerzo	Cronograma
<b>Orgánico</b>	$PM=2.4 \times (KSLOC)^{1.05}$	$TDEV=2.5 \times (PM)^{0.38}$
<b>Semiacoplado</b>	$PM=3.0 \times (KSLOC)^{1.12}$	$TDEV=2.5 \times (PM)^{0.35}$
<b>Empotrado</b>	$PM=3.6 \times (KSLOC)^{1.20}$	$TDEV=2.5 \times (PM)^{0.32}$

Tabla 1: Ecuaciones del Modelo Básico de COCOMO 81. [Boehm 1981]

Este modelo es adecuado para una estimación rápida y temprana, pero su precisión es muy limitada debido a que no contempla factores que tienen significativa influencia en los costos, como

---

<sup>3</sup> Un mes-persona equivale a 152 horas de trabajo y corresponde a la cantidad de tiempo que una persona dedica durante un mes a trabajar en un proyecto de desarrollo de software. Este valor tiene en cuenta los fines de semana pero excluye feriados y vacaciones.

por ejemplo, restricciones de hardware, experiencia y calidad del equipo de trabajo, y uso de técnicas y herramientas modernas.

La Tabla 2 presenta los porcentajes que determinan cómo se distribuye el esfuerzo y el cronograma, según el tamaño y el modo de desarrollo, en cada fase del ciclo de vida del proyecto.

Distribución del esfuerzo según		Tamaño									
		Pequeño 2 KSLOC		Intermedio 8 KSLOC		Medio 32 KSLOC		Grande 128 KSLOC		Muy Grande 512 KSLOC	
Modo	Fase										
Orgánico	Requerimientos (%)		6		6		6		6		
	Diseño del Producto	16		16		16		16			
	Programación	68		65		62		59			
	Diseño Detallado		26		25		24		23		
	Codificación y Testeo		42		40		38		36		
	Integración y Testeo	16		19		22		25			
Semiacoplado	Requerimientos (%)		7		7		7		7		7
	Diseño del Producto	17		17		17		17		17	
	Programación	64		61		58		55		52	
	Diseño Detallado		27		26		25		24		23
	Codificación y Testeo		37		35		33		31		29
	Integración y Testeo	19		22		25		28		31	
Empotrado	Requerimientos (%)		8		8		8		8		8
	Diseño del Producto	18		18		18		18		18	
	Programación	60		57		54		51		48	
	Diseño Detallado		28		27		26		25		24
	Codificación y Testeo		32		30		28		26		24
	Integración y Testeo	22		25		28		31		34	
Distribución del cronograma según		Pequeño 2 KSLOC		Intermedio 8 KSLOC		Medio 32 KSLOC		Grande 128 KSLOC		Muy Grande 512 KSLOC	
Orgánico	Requerimientos (%)		10		11		12		13		
	Diseño del Producto	19		19		19		19			
	Programación	63		59		55		51			
	Integración y Testeo	18		22		26		30			
Semiacoplado	Requerimientos (%)		16		18		20		22		24
	Diseño del Producto	24		25		26		27		28	
	Programación	56		52		48		44		40	
	Integración y Testeo	20		23		26		29		32	
Empotrado	Requerimientos (%)		24		28		32		36		40
	Diseño del Producto	30		32		34		36		38	
	Programación	48		44		40		36		32	
	Integración y Testeo	22		24		26		28		30	

Tabla 2: Distribución del Esfuerzo y Cronograma en las distintas fases. [Boehm 1981]

En cuanto a las actividades, el esfuerzo se distribuye en función del tamaño y del modo de desarrollo, como se puede apreciar en [Boehm 1981], páginas 99, 100 y 101.

### 3.3 Modelo Intermedio

Comparado con el modelo anterior, éste provee un nivel de detalle y precisión superior, por lo cual es más apropiado para la estimación de costos en etapas de mayor especificación. *COCOMO Intermedio* incorpora un conjunto de quince variables de predicción que toman en cuenta las variaciones de costos no consideradas por *COCOMO Básico*.

Existen diversos factores a considerar en el desarrollo de un buen modelo de estimación de costos de un proyecto de software. Para reducir el número a una cantidad relativamente manejable se utilizaron fundamentalmente dos principios:

- Eliminar aquellos factores que son significativos solamente en una fracción relativamente pequeña o en situaciones especiales.
- Eliminar los factores que están altamente correlacionados con el tamaño y comprimir aquellos factores correlacionados entre sí

Los factores seleccionados se agrupan en cuatro categorías:

- *Atributos del producto de software*
  - RELY      Confiabilidad Requerida
  - DATA      Tamaño de la Base de Datos
  - CPLX      Complejidad del Producto
- *Atributos del hardware*
  - TIME      Restricción del Tiempo de Ejecución
  - STOR      Restricción del Almacenamiento Principal
  - VIRT      Volatilidad de la Máquina Virtual\*
  - TURN      Tiempo de Respuesta de la computadora expresado en horas
- *Atributos del personal involucrado en el proyecto*
  - ACAP      Capacidad del Analista
  - AEXP      Experiencia en Aplicaciones Similares
  - PCAP      Capacidad del Programador
  - VEXP      Experiencia en la máquina virtual
  - LEXP      Experiencia en el Lenguaje de Programación
- *Atributos propios del proyecto*
  - MODP      Prácticas Modernas de Programación
  - TOOL      Uso de Herramientas de Software
  - SCED      Cronograma de Desarrollo Requerido

El proceso de estimación del esfuerzo puede sintetizarse en los siguientes pasos:

---

\* *Nota:* Se entiende por máquina virtual subyacente al conjunto de hardware y software que el sistema necesita para cumplimentar su tarea.

- Se calcula el esfuerzo nominal  $PM_{Nominal}$ , al igual que en el *modelo Básico*, donde los únicos factores de costo son el tamaño y el modo de desarrollo.
- Se determina el *Factor de Ajuste del Esfuerzo (EAF, Effort Adjustment Factor)* según la

$$\text{fórmula: } EAF = \prod_{i=1}^{15} EM_i$$

Donde cada EM, llamado factor multiplicador de esfuerzo, es el valor que corresponde a cada atributo de acuerdo al grado de influencia (Muy Bajo, Bajo, Nominal, Alto, Muy Alto, Extra Alto) en el esfuerzo del desarrollo del software. La tabla 8-3 de [Boehm 1981] muestra las características que determinan en que nivel se puede clasificar cada factor de costo, y la tabla 8-2 de [Boehm 1981] establece el valor de cada factor para cada nivel.

- Finalmente, se ajusta el esfuerzo nominal aplicando el EAF.

$$PM = A \times EAF \times (KSLOC)^B$$

La Tabla 3 muestra la variación de la ecuación de estimación de esfuerzo y cronograma según los tres modos de desarrollo.

Modo de Desarrollo	Esfuerzo Nominal	Esfuerzo Ajustado	Cronograma
<b>Orgánico</b>	$PM_{nominal} = 3.2 \times (KSLOC)^{1.05}$	$PM = 3.2 \times EAF \times (KSLOC)^{1.05}$	$TDEV = 2.5 \times (PM)^{0.38}$
<b>Semiacoplado</b>	$PM_{nominal} = 3.0 \times (KSLOC)^{1.12}$	$PM = 3.0 \times EAF \times (KSLOC)^{1.12}$	$TDEV = 2.5 \times (PM)^{0.35}$
<b>Empotrado</b>	$PM_{nominal} = 2.8 \times (KSLOC)^{1.20}$	$PM = 2.8 \times EAF \times (KSLOC)^{1.20}$	$TDEV = 2.5 \times (PM)^{0.32}$

Tabla 3: Ecuaciones del Modelo Intermedio de COCOMO 81. [Boehm 1981]

Los porcentajes de distribución de esfuerzo y cronograma por fase se obtienen en función del modo y tamaño del producto, según Tabla 2, página 11.

En lo referente a la distribución del esfuerzo por fase y por actividades, también se tiene en cuenta el tamaño del software y el modo de desarrollo. Las tablas correspondientes se encuentran en [Boehm 1981], páginas 99, 100 y 101.

*COCOMO Intermedio* posibilita estimar el esfuerzo de un proyecto al nivel de componentes. De esta manera, cada componente individual tendrá sus propios parámetros representativos del tamaño y de los 15 factores de costo. En consecuencia, es factible experimentar con diferentes estrategias de desarrollo que hacen posible encontrar el plan que mejor se ajuste a las necesidades y recursos existentes.

Es importante destacar que este modelo tiene dos limitaciones importantes a la hora de estimar grandes proyectos de software:

- La estimación de la distribución del esfuerzo para cada fase resulta imprecisa.
- No es muy práctico si el producto de software tiene un gran número de componentes.

### 3.4 Modelo Detallado

El *Modelo Detallado* provee los medios para generar estimaciones con mayor grado de precisión y detalle. Difiere del *Modelo Intermedio* en dos aspectos principales que ayudan a superar las limitaciones mencionadas en 3.3:

- **Jerarquía de niveles del producto.** En el *Modelo Intermedio* se pueden calcular valores diferentes de los factores de costo para cada componente de software. Este proceso puede resultar muy tedioso e innecesariamente repetitivo si las componentes están agrupadas en subsistemas de características generales similares. Para subsanar este problema el *Modelo Detallado* aplica al producto de software una descomposición jerárquica de tres niveles. En el nivel inferior, nivel de módulo, la estimación se basa en el número de líneas de código del módulo (SLOC) y aquellos factores que tienden a variar en ese nivel: complejidad del módulo y adaptación del software existente, nivel de capacidad y experiencia del programador, con el lenguaje y la máquina virtual sobre la que se construirá el software. El segundo nivel, nivel de subsistema, está descrito por el resto de los factores de costo que pueden variar de un subsistema a otro, pero que tienden a ser los mismos para todos los módulos dentro de un subsistema. Entre ellos se encuentran: restricciones de tiempo y espacio, capacidad del analista, herramientas, etc. El nivel superior, nivel de sistema, se usa para aplicar las ecuaciones de esfuerzo nominal y cronograma y calcular las estimaciones tanto para todo el proyecto como para cada fase.
- **Multiplicadores de Esfuerzo (EM Effort Multipliers) sensitivos a las fases.** El *modelo Detallado* provee un conjunto de multiplicadores diferentes para cada factor de costo, según la fase del ciclo de desarrollo que se considere. De esta forma los multiplicadores se utilizan para determinar el esfuerzo requerido para completar cada fase, ver Tabla 4 y Tabla 5 que muestran los factores que afectan al proyecto al nivel de módulo y subsistema respectivamente.

Las ecuaciones fundamentales de este modelo son similares a las del *modelo COCOMO Intermedio*, la única diferencia reside en el cálculo del Factor de Ajuste del Esfuerzo (EAF). El procedimiento incluye el cálculo de un Factor de Ajuste del Esfuerzo al nivel de módulo (EAF<sub>M</sub>) y otro al nivel de subsistema (EAF<sub>S</sub>).

#### 3.4.1 Procedimiento de estimación de esfuerzo

El *modelo COCOMO Detallado* provee un conjunto de procedimientos que permiten estimar esfuerzo y costo al nivel de módulo, subsistema y sistema tanto para cada fase de desarrollo como para el proyecto en su totalidad. Para registrar los datos necesarios se emplean dos formularios tipo llamados SHEF<sup>4</sup>. Ver Anexo I.

##### Formularios

El procedimiento involucra el uso de la Tabla 4, la cual muestra los valores de los multiplicadores de esfuerzo que afectan a los módulos en las distintas etapas de desarrollo, y la Tabla 5 donde se observan de igual modo los multiplicadores de esfuerzo que influyen sobre los subsistemas.

Las Figura 1 y Figura 2 muestran un ejemplo del uso de los formularios SHEF para la Estimación de Esfuerzo Jerárquica de un Sistema de Administración de Trabajo para Estudiantes (STUJOB) [Boehm 1981], el cual consta de tres subsistemas:

---

<sup>4</sup> Software Hierarchy Estimating Form

1. **Query:** Subsistema a través del cual los alumnos pueden consultar en la base de datos los trabajos ofrecidos y a su vez los empleadores buscar los alumnos disponibles y de esta forma encontrar alguien que reúna los requisitos. Este subsistema está formado por tres módulos:
  - Qedit:** Módulo de entrada y edición de datos.
  - Search:** Módulo de búsquedas.
  - Output:** Módulo para reportar los resultados de las búsquedas.
2. **Update:** Subsistema encargado de las altas, bajas y modificaciones en la base de datos de los empleos ofrecidos y los alumnos disponibles. Conformado por dos módulos:
  - Upedit:** Módulo de entrada y edición de datos.
  - Modify:** Módulo de modificación del archivo.
3. **Utilities:** Subsistema que abarca todas las facilidades extras que provee el sistema como por ejemplo: chequeos de integridad, backup de archivos, análisis estadísticos, etc. Compuesto por un solo módulo, el cual es la adaptación de un módulo existente.

La intención es implementar el sistema **STUJOB** en una computadora mediana de propósito general. El hardware, sistema operativo, compilador, administrador de archivos y generador de reportes a ser usados por el sistema son estables y se cuenta con un gran número de herramientas útiles para asistir el desarrollo del software. La aplicación no requiere un alto nivel de confiabilidad ni está sometida a grandes presiones en lo referente a tiempo de ejecución y uso de memoria. La base de datos con la información de trabajos y estudiantes no es muy extensa.

El administrador del proyecto planea hacer el análisis preliminar y trabajar con los programadores en el análisis y diseño de los módulos que cada uno programará. El módulo **Search** a pesar de ser el más pequeño es el que requiere un entendimiento más profundo del sistema operativo subyacente y del software de administración de datos, por esta razón será el administrador en persona quien se encargue de su programación. También será primariamente responsable de la integración final y testeo del software. Un programador del equipo con capacidad y experiencia razonable será el encargado de trabajar en los módulos **Qedit** y **UpEdit**. Un grupo de programadores nuevos recién incorporados al equipo trabajará en los otros tres módulos de menor complejidad, **Output**, **Modify** y **Utilities**.



Proyecto: <i>Sistema de Administración de Trabajo para Estudiantes</i>			Analista: <i>Juan Pérez</i>											Fecha: <i>25/10/85</i>								
1 Nro SS	02 Subsiste ma SS	8 SLOC	Producto		Atributos de la Computadora				Personal		Proyecto			32 EAF SS	20 PM MOD	33 PM EST	34 Costo Mes- Person a	35 Costo	36 Totales Prom.			
			21 RELY	22 DATA	23 TIME	24 STOR	25 VIRT	26 TURN	27 ACAP	28 AEXP	29 MODP	30 TOOL	31 SCED									
1	<i>Query</i>	3700	PD 1.00 DD 1.00 CT 1.00 IT 1.00	1.00	1.00	1.00	1.00	0.95 0.90 0.85 0.80	1.00	0.75 0.75 0.90 0.85	0.75 0.80 0.85 0.85	1.00	0.98 0.95 0.90 0.85	1.00	0.52 0.58 0.53 0.41	2.0 3.2 5.5 2.5	1.0 1.9 2.9 1.0	6.0 5.0 5.0 5.5	6.0 9.5 14.5 5.5	6.8 <sup>1</sup> 35.5 <sup>2</sup> 544 <sup>3</sup> 10 <sup>4</sup>		
2	<i>Update</i>	2600	1.00	1.00	1.00	1.00	0.95 0.90 0.85 0.80	1.00	0.75 0.75 0.90 0.85	0.75 0.80 0.85 0.85	1.00	0.98 0.95 0.90 0.85	1.00	0.52 0.58 0.53 0.41	1.4 2.4 4.1 1.9	0.7 1.4 2.2 0.8	6.0 5.0 5.0 5.5	4.2 7.0 11.0 4.4	5.1 26.6 510 10			
3	<i>Utilities</i>	1700	1.00	1.00	1.00	1.00	0.95 0.90 0.85 0.80	1.00	1.00	1.00	1.00	0.98 0.95 0.90 0.85	1.00	0.93 0.81 0.69 0.56	0.7 1.4 2.5 1.2	0.7 1.1 1.7 0.7	5.0 5.0 5.5 5.5	3.5 5.5 9.4 3.8	4.2 22.2 405 13			
de Desarrollo		8.000	Total SLOC																		Product. Estimada	
10	Orgánico	28.4	Esfuerzo Nominal PM <sub>Nominal</sub>																			SLOC/PM 8000/16.1= 496.8—497
11		282	Productividad Nominal (SLOC/PM) <sub>Nominal</sub>																			Costo Total p/Instruc. 84.3/8 —11

12	% de Distribución del PM por Fase	
	PD	16
	DD	25
	CT	40
	IT	19

1	Esfuerzo Estimado por Subsistema	38
2	Costo Total del Subsistema en \$	39
3	Productividad	
4	Costo por instrucción	

PM <sub>Estimado</sub>	Cronograma TDEV	Costo Total del Sistema
38	16.1	84.3

Figura 1: Formulario para la Estimación Jerárquica de Software al nivel de Subsistema (SHEF). [Boehm 1989]

Proyecto: <i>Sistema de Administración de Trabajo para Estudiantes</i>						Analista: <i>Juan Pérez</i>				Fecha: <i>25/10/85</i>			
3 Nro. SS	4 Nro. Módulo	5 Módulo	6 SLOC	7 AAF	14 CPLX	15 PCAP	16 VEXP	17 LEXP	18 EAF Módulo	13 PM Nominal por Módulo	19 PM Estimado por Módulo	37 PM Estimado	40 % de Distribución del PM por Fase
1	1	<i>QEdit</i>	1800	100	1.00	1.00	1.00	1.00	1.0 1.0 1.0 1.0	1.0 1.6 2.6 1.2	1.0 1.6 2.6 1.2	0.5 0.9 1.4 0.5	15 27 42 15
1	2	<i>Search</i>	700	100	1.00	1.00 0.83 0.83 0.83	0.90 0.90 0.90 0.90	1.00 0.98 0.92 0.92	0.90 0.73 0.69 0.69	0.4 0.6 1.0 0.5	0.4 0.4 0.7 0.3	0.2 0.2 0.4 0.1	22 22 45 12
1	3	<i>Output</i>	1200	100	0.85 0.85 0.85 0.85	1.00 1.20 1.20 1.20	1.05 1.05 1.15 1.15	1.00 1.05 1.10 1.10	0.89 0.12 1.29 1.29	0.7 1.1 1.7 0.8	0.6 1.2 2.2 1.0	0.3 0.7 1.2 0.4	12 27 46 15
1	Total		3700								2.0 3.2 5.5 2.5	1.0 1.8 3.0 1.0	15 26 44 15
2	1	<i>UpEdit</i>	1700	100	1.00	1.00	1.00	1.00	1.0 1.0 1.0 1.0	1.0 1.5 2.4 1.1	1.0 1.5 2.4 1.1	0.5 0.9 1.3 0.5	16 28 40 16
2	2	<i>Modify</i>	900	100	0.85 0.85 0.85 0.85	1.00 1.20 1.20 1.20	1.05 1.05 1.15 1.15	1.00 1.05 1.10 1.10	0.89 1.12 1.29 1.29	0.5 0.8 1.3 0.6	0.4 0.9 1.7 0.8	0.2 0.5 0.9 0.3	11 26 47 16
2	Total		2600								1.4 2.4 4.1 1.9	0.7 1.4 2.2 0.8	14 27 43 16
3	1	<i>Utilities</i>	1700	34	0.70 0.70 0.70 0.70	1.00 1.20 1.20 1.20	1.05 1.05 1.15 1.15	1.00 1.05 1.10 1.10	0.74 0.93 1.06 1.06	1.0 1.5 2.4 1.1	0.7 1.4 2.5 1.2	0.7 1.1 1.7 0.7	17 26 40 17

Figura 2: Formulario para la Estimación Jerárquica de Software al nivel de Módulo (SHEF). [Boehm 1989]

En el procedimiento a seguir en la estimación de costo y esfuerzo se utilizan los dos formularios mencionados y los pasos a seguir son:

1. Identificar los subsistemas que componen el sistema, asignarles un número y un nombre e ingresarlos en las columnas **1** y **2**, respectivamente. Ej: **Subsistema 2: Update**.
2. Identificar los módulos que contiene cada subsistema registrado en el paso anterior, asignarles un número en la columna **4** y un nombre en la columna **5**. Ej. **Módulo 2 del Subsistema 2: Modify**.
3. Después de ingresar todos los módulos dejar una fila adicional para acumular los totales por subsistema.
4. Determinar el tamaño de cada módulo expresado en SLOC, líneas de código fuentes liberadas, y registrarlo en la columna **6**. Si algún módulo va a ser adaptado a partir de un módulo existente se consideran las ESLOC, líneas de código fuentes equivalentes, calculadas a partir de la siguiente ecuación:

$$ESLOC = SLOC \text{ adaptadas} \times AAF / 100$$

$$AAF = 0.4 \times DM + 0.3 \times CM + 0.3 \times IM$$

Donde:

DM es el Porcentaje de Diseño Modificado para satisfacer los nuevos requerimientos

CM es el Porcentaje de Código Modificado para adaptarse a los nuevos requerimientos

IM es el Porcentaje de Esfuerzo requerido para Integrar el software adaptado al producto global

Registrar el factor AAF en la columna **7**.

Ej: Para el **Módulo Utilities**, adaptado a partir de un módulo de **5000 SLOC**, considerando un DM igual a 25%, un CM de 50 % y un IM de 30%, resulta un valor ESLOC igual a 1700.

$$AAF = 0.4 \times 25 + 0.3 \times 50 + 0.3 \times 30 = 34$$

$$ESLOC(\text{Utilities}) = 5000 \times 34 / 100 = 1700$$

5. Determinar el tamaño en SLOC de los subsistemas, sumando el tamaño de los módulos que componen cada subsistema. Registrarlo en la columna **8**.

Ej: **Tamaño del Subsistema Update: 900+1700=2600 SLOC**

6. Determinar el tamaño en SLOC del Sistema, sumando el tamaño de los subsistemas que componen el sistema. Anotarlo en la celda **9**.

Ej: **Tamaño del Sistema: 3700+2600+1700 = 8000 SLOC**

7. Calcular el Esfuerzo Nominal requerido para desarrollar el sistema,  $PM_{\text{Nominal}}$ , en la celda **10** y la Productividad del Proyecto  $(KSLOC/PM)_{\text{Nominal}}$  en la celda **11**. Teniendo en cuenta el **Modo de Desarrollo** seleccionar la ecuación correspondiente de la Tabla 3.

Ej: Para el **Modo Orgánico**

$$PM_{\text{nominal}} = 3.2 \times (KSLOC)^{1.05} = 3.2 \times (8)^{1.05} = 28.4$$

$$(KSLOC/PM)_{\text{Nominal}} = (8000/28.4) = 281.69 \cong 282$$

8. Seleccionar los porcentajes de Distribución por Fase en la Tabla 2, según el **Modo de Desarrollo** y el **Tamaño** del sistema, y completar el cuadro **12**.
9. Calcular y registrar en la columna **13** el Esfuerzo Nominal por Módulo y Fase ( $PM_{\text{Nominal, Módulo, Fase}}$ ), aplicando los porcentajes del cuadro **12** al Esfuerzo Nominal por Módulo ( $PM_{\text{Nominal, Módulo}}$ ). Este último valor se obtiene del cociente entre el tamaño del módulo (columna **5**) y la Productividad del Proyecto (celda **11**).

Ej: Para el módulo **Modify**  $PM_{Nominal,Módulo} = 900 / 282 = 3.1914 \cong 3.2$

Este valor se distribuye por fase de la siguiente forma:

Fase		%	$PM_{Nominal,Módulo,Fase}$
Fase Diseño del Producto	PD	16%	<b>0.5</b>
Fase Diseño Detallado	DD	25%	<b>0.8</b>
Fase Codificación y Testeo	CT	40%	<b>1.28 <math>\cong</math> 1.3</b>
Fase Integración y Testeo	IT	19%	<b>0.6</b>

10. Analizar las características de cada módulo y determinar en que nivel (Muy Bajo, Bajo, Nominal, Alto, Muy Alto) se encuentra cada uno de los siguientes factores de costo: CPLX, PCAP, VEXP y LEXP. Según el nivel determinado encontrar los valores de los multiplicadores de esfuerzo correspondientes a cada fase, en la Tabla 4 y completar las columnas **14** a **17**.

Ej: Considerando el módulo **Modify**, de escasa complejidad y a cargo de un equipo de programadores principiantes, los factores se evalúan en un nivel **Bajo**, por lo tanto los valores para cada fase según la Tabla 4 son:

		PD	DD	CT	IT
<b>CPLX</b>	<b>Bajo</b>	<b>0.85</b>	<b>0.85</b>	<b>0.85</b>	<b>0.85</b>
<b>PCAP</b>	<b>Bajo</b>	<b>1.00</b>	<b>1.20</b>	<b>1.20</b>	<b>1.20</b>
<b>VEXP</b>	<b>Bajo</b>	<b>1.05</b>	<b>1.05</b>	<b>1.15</b>	<b>1.15</b>
<b>LEXP</b>	<b>Bajo</b>	<b>1.00</b>	<b>1.05</b>	<b>1.10</b>	<b>1.10</b>

11. Multiplicar los multiplicadores de esfuerzo de la columna **14** a la **17** para cada fila y así obtener el Factor de Ajuste del Esfuerzo  $EAF_M$  para cada módulo y fase. Ingresar los resultados en la columna **18**.

Ej: Para el **Módulo Modify**, considerando la fase **PD** el cálculo es:

$$EAF_M = 0.85 \times 1.0 \times 1.05 \times 1.00 = 0.89$$

12. Calcular el Esfuerzo Estimado por Módulo y Fase,  $PM_{Estimado,Módulo,Fase}$ , en la columna **19**, multiplicando el valor de  $PM_{Nominal,Módulo,Fase}$ , columna **13**, por el correspondiente Factor de Ajuste  $EAF_M$  de la columna **18**.

Ej: Para el **Módulo Modify**, considerando la fase **PD** el cálculo es:

$$PM_{Estimado,Módulo,Fase} = PM_{Nominal,Módulo,Fase} \times EAF_M = 0.5 \times 0.89 = 0.445 \cong 0.4$$

13. Para cada Subsistema sumar los valores de  $PM_{Estimado,Módulo,Fase}$  correspondientes a cada Fase, de todos los módulos que lo componen y registrarlos en la columna **20**.

Ej: Para el **Subsistema Update**, constituido por los módulos **UpEdit** y **Modify**, considerando la fase **PD** el cálculo es:

$$PM_{Estimado,M,F(Update,PD)} = PM_{Estimado,M,F(UpEdit,PD)} + PM_{Estimado,M,F(Modify,PD)} = 1.0 + 0.4 = 1.4$$

14. Analizar las características de cada subsistema y determinar en que nivel (Muy Bajo, Bajo, Nominal, Alto, Muy Alto) se encuentra cada uno de los siguientes factores de costo: RELY, DATA, TIME, STOR, VIRT, TURN, ACAP, AEXP, MODP, TOOL, SCED. Según el nivel determinado encontrar los valores de los multiplicadores de esfuerzo correspondientes a cada Fase, según Tabla 5, y completar las columnas **21** a **31**.

Ej: En el **Subsistema Query** al considerar la participación del líder del proyecto, el factor **ACAP** se evalúa en un nivel **Alto**, por lo tanto los valores que muestra para cada fase la Tabla 5 son:

		PD	DD	CT	IT
ACAP	Alto	0.75	0.90	0.90	0.85

15. Multiplicar los multiplicadores de esfuerzo de la columna **21** a la **31** para cada fila y así obtener el Factor de Ajuste del Esfuerzo **EAF<sub>S</sub>** para cada subsistema y fase. Ingresar los resultados en la columna **32**.

Ej: Para el **Subsistema Update**, considerando la fase **PD** el cálculo es:

$$\mathbf{EAF_S = 1.0 \times 1.0 \times 1.0 \times 0.95 \times 1.0 \times 0.75 \times 0.75 \times 1.0 \times 0.98 \times 1.0 = 0.52}$$

16. Calcular el Esfuerzo Estimado por Subsistema y Fase, **PM<sub>Estimado,Subsistema,Fase</sub>**, multiplicando el valor de **PM<sub>Estimado,Módulo,Fase</sub>**, columna **20**, por el correspondiente Factor de Ajuste **EAF<sub>S</sub>** de la columna **32** y completar la columna **33**.

Ej: Para el **Subsistema Update**, considerando la fase **PD** el cálculo es:

$$\mathbf{PM_{Estimado,Subsistema,Fase} = PM_{Estimado,Módulo,Fase} \times EAF_S = 1.4 \times 0.52 = 0.728 \cong 0.7}$$

17. Determinar el Esfuerzo Estimado por Fase para el Sistema en su totalidad **PM<sub>Estimado</sub>**, sumando los **PM<sub>Estimado,Subsistema,Fase</sub>**, para cada fase y para todos los subsistemas. Registrar estos valores en la celda **38**.

Ej: Considerando la Fase **PD**

$$\mathbf{PM_{Estimado,F} = PM_{Estimado,S,F(Query)} + PM_{Estimado,S,F(Update)} + PM_{Estimado,S,F(Utilities)} = 1.0 + 0.7 + 0.7 = 2.4}$$

18. Sumar los 4 valores calculados en el ítem anterior para determinar el Esfuerzo Estimado para el Sistema Total **PM<sub>Estimado</sub>**, registrar este valor en la celda **39**.

Ej:  $\mathbf{PM_{Estimado} = 3 \times PM_{Estimado,Fase} = 2.4 + 4.4 + 6.8 + 2.5 = 16.1}$

Se puede observar que el valor obtenido es un poco más de la mitad del Esfuerzo Nominal **PM<sub>Nominal</sub>** calculado originariamente (**28.4**). La distribución por fase también se ha modificado debido a la influencia de los multiplicadores de esfuerzo, de la siguiente forma:

Fase		% Original	% Ajustado
Fase Diseño del Producto	PD	16%	2.4/16.1 * 100 = <b>15</b>
Fase Diseño Detallado	DD	25%	4.4/16.1 * 100 = <b>27</b>
Fase Codificación y Testeo	CT	40%	6.8/16.1 * 100 = <b>42</b>
Fase Integración y Testeo	IT	19%	2.5/16.1 * 100 = <b>16</b>

19. Determinar el Cronograma Global Estimado del proyecto **TDEV**. Analizando el **Modo de Desarrollo** seleccionar la ecuación correspondiente en la Tabla 3, anotar el resultado en la última celda de la columna **34**.

Ej: Para el **Modo Orgánico**

$$\mathbf{TDEV_{Estimado} = 2.5 \times (PM_{Estimado})^{0.38} = 2.5 \times (16.1)^{0.38} = 7.1867 \cong 7.2 \text{ meses}}$$

20. Anotar en la columna **34** el Costo del Mes-Persona para cada Fase de cada Subsistema, expresado en miles de dólares. Posteriormente multiplicar estos costos por los **PM<sub>Estimado,Subsistema,Fase</sub>** correspondientes (columna **33**), para encontrar el Costo Estimado, en miles de dólares, para cada Subsistema y Fase y registrarlos en la columna **35**.

Ej: Para el **Subsistema Query**, al considerar la fase **PD** se asume un costo más elevado debido a la participación del líder del proyecto:

$$\mathbf{Costo \text{ Mes-Persona} = 6.0}$$

$$\text{Costo}_{\text{Estimado,Subsistema,Fase}} = \text{Costo Mes-Persona} \times \text{PM}_{\text{Estimado,Subsistema,Fase}} = 6.0 \times 0.7 = 4.2$$

21. Obtener el Costo Total del Sistema por Fase, mediante la sumatoria de los valores correspondientes a cada Fase (columna **35**) y registrarlo en la penúltima celda de la misma columna.

Ej: Considerando la fase **PD** los cálculos son:

$$\text{Costo}_{\text{Estimado,Fase}} = 6.0 + 4.2 + 3.5 = 13.7$$

22. Calcular el Costo Total del Sistema sumando los cuatro valores obtenidos en el ítem anterior y registrarlo en la última celda de la columna **35**.

Ej:  $\text{Costo}_{\text{Estimado}} = 13.7 + 22.0 + 34.9 + 13.7 = 84.3$

23. Para cada Subsistema determinar y registrar en la columna **36** los siguientes valores totalizadores:

Esfuerzo Estimado: suma de los valores correspondientes de la columna **33**

Costo de Desarrollo en \$: suma de los valores correspondientes de la columna **35**

Productividad: cociente entre el Tamaño del Subsistema (columna **8**) y el Esfuerzo Estimado

Costo por instrucción en \$: cociente entre el Costo de Desarrollo y el Tamaño del Subsistema (columna **8**)

Ej: Para el **Subsistema Update**

$$\text{Esfuerzo Estimado} = 0.7 + 1.4 + 2.2 + 0.8 = 5.1$$

$$\text{Costo Total en \$} = 4.2 + 7.0 + 11.0 + 4.4 = 26.6$$

$$\text{Productividad} = 2600 / 5.1 = 509.80 \cong 510$$

$$\text{Costo por instrucción en miles de \$} = 26.6 / 2600 = 0.01023 \cong 10 \text{ dólares}$$

24. Multiplicar el Esfuerzo Estimado por Módulo y por Fase  $\text{PM}_{\text{Estimado,Módulo,Fase}}$ , columna **19** por el factor de ajuste  $\text{EAF}_s$  columna **32** para obtener un resultado más preciso y registrarlo en la columna **37**.

25. Sumar los  $\text{PM}_{\text{Estimado,Módulo,Fase}}$  calculados en el punto anterior, para un mismo subsistema y colocar este valor en la fila correspondiente al Total por Subsistema de la columna **37**. Comparar estos resultados con los valores estimados para el Subsistema completo,  $\text{PM}_{\text{Estimado,Subsistema,Fase}}$ , registrados en la columna **33**. Estos resultados deberían ser iguales, tal vez con alguna diferencia en el último dígito debido a errores de redondeo, de no ser así deberían revisarse los cálculos hasta encontrar y corregir la diferencia.

Ej: Para el **Subsistema Update** comparar las celdas con contorno doble.

26. Por último se agrega la columna **40**, que muestra los porcentajes de distribución de esfuerzo resultantes para cada fase y cada módulo. Se puede apreciar como varían con respecto a la distribución nominal, considerada originalmente, cuadro **12**. Las variaciones se producen a causa de los factores sensitivos a las fases. Por ejemplo en el **módulo Search**, desarrollado por el líder del proyecto, se observa un aumento de la fracción de esfuerzo dedicada al Diseño del Producto, de **16%** a **22%** y como contrapartida una disminución de la fracción de esfuerzo dedicada a la fase de Integración y Testeo, del **19%** al **12%**.

Atributo	Nivel	PD	DD	CT	IT
<b>CPLX</b>	Muy bajo	0.70	0.70	0.70	0.70
	Bajo	0.85	0.85	0.85	0.85
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.15	1.15	1.15	1.15
	Muy alto	1.30	1.30	1.30	1.30
	Extra alto	1.65	1.65	1.65	1.65
<b>PCAP</b>	Muy bajo	1.00	1.50	1.50	1.50
	Bajo	1.00	1.20	1.20	1.20
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.00	0.83	0.83	0.83
	Muy alto	1.00	0.65	0.65	0.65
<b>VEXP</b>	Muy bajo	1.10	1.10	1.30	1.30
	Bajo	1.05	1.05	1.15	1.15
	Nominal	1.00	1.00	1.00	1.00
	Alto	0.90	0.90	0.90	0.90
<b>LEXP</b>	Muy bajo	1.02	1.10	1.20	1.20
	Bajo	1.00	1.05	1.10	1.10
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.00	0.98	0.92	0.92

Tabla 4: Multiplicadores de Esfuerzo al nivel de módulo. [Boehm 1981]

Atributo	Nivel	PD	DD	CT	IT
<b><u>Producto</u></b>					
<b>RELY</b>	Muy bajo	0.80	0.80	0.80	0.60
	Bajo	0.90	0.90	0.90	0.80
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.10	1.10	1.10	1.30
	Muy alto	1.30	1.30	1.30	1.70
<b>DATA</b>	Bajo	0.95	0.95	0.95	0.90
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.10	1.05	1.05	1.15
	Muy alto	1.20	1.10	1.10	1.30
<b><u>Computadora</u></b>					
<b>TIME</b>	Nominal	1.00	1.00	1.00	1.00
	Alto	1.10	1.10	1.10	1.15
	Muy alto	1.30	1.25	1.25	1.40
	Extra alto	1.65	1.55	1.55	1.95
<b>STOR</b>	Nominal	1.00	1.00	1.00	1.00

	Alto	1.05	1.05	1.05	1.10
	Muy alto	1.20	1.15	1.15	1.35
	Extra alto	1.55	1.45	1.45	1.85
<b>VIRT</b>	Bajo	0.95	0.90	0.85	0.80
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.10	1.12	1.15	1.20
	Muy alto	1.20	1.25	1.30	1.40
<b>TURN</b>	Bajo	0.98	0.95	0.70	0.90
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.00	1.00	1.10	1.15
	Muy alto	1.02	1.05	1.20	1.30
<b><u>Personal</u></b> <b>ACAP</b>	Muy bajo	1.80	1.35	1.35	1.50
	Bajo	1.35	1.15	1.15	1.20
	Nominal	1.00	1.00	1.00	1.00
	Alto	0.75	0.90	0.90	0.85
	Muy alto	0.55	0.75	0.75	0.70
<b>AEXP</b>	Muy bajo	1.40	1.30	1.25	1.25
	Bajo	1.20	1.15	1.10	1.10
	Nominal	1.00	1.00	1.00	1.00
	Alto	0.87	0.90	0.92	0.92
	Muy alto	0.75	0.80	0.85	0.85
<b><u>Proyecto</u></b> <b>MODP</b>	Muy bajo	1.05	1.10	1.25	1.50
	Bajo	1.00	1.05	1.10	1.20
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.00	0.95	0.90	0.83
	Muy alto	1.00	0.90	0.80	0.65
<b>TOOL</b>	Muy bajo	1.02	1.05	1.35	1.45
	Bajo	1.00	1.02	1.15	1.20
	Nominal	1.00	1.00	1.00	1.00
	Alto	0.98	0.95	0.90	0.85
	Muy alto	0.95	0.90	0.80	0.70
<b>SCED</b>	Muy bajo	1.10	1.25	1.25	1.25
	Bajo	1.00	1.15	1.15	1.10
	Nominal	1.00	1.00	1.00	1.00
	Alto	1.10	1.10	1.00	1.00
	Muy alto	1.15	1.15	1.05	1.05

Tabla 5: Multiplicadores de Esfuerzo al nivel de subsistema. [Boehm 1981]

Para la estimación del esfuerzo distribuida por actividades se usan las mismas técnicas de cálculo que en los *Modelos Básico e Intermedio*. Ver Tablas de las páginas 99, 100 y 101 [Boehm 1981].

### 3.4.2 Procedimiento de estimación del cronograma

Analizando los porcentajes de distribución por fase del esfuerzo estimado (ver Figura 2 columna **40**) se puede observar que difieren de la distribución nominal usada en los *Modelos Básico e Intermedio* (Tabla 2). Claramente esta variación debería reflejarse en la distribución por fase del cronograma estimado. Esto se logra modificando los porcentajes nominales correspondientes en igual proporción en que cambian los porcentajes de distribución del esfuerzo nominal y estimado. Por lo tanto, el valor para cada fase se calcula de acuerdo a la siguiente ecuación:

Donde:

$$PDC_{Ajus} = \frac{PDC_{Nom} \times PDE_{Ajus}}{PDE_{Nom}}$$

$PDC_{Nom}$ : Porcentaje de Distribución Nominal del Cronograma

$PDE_{Nom}$ : Porcentaje de Distribución Nominal del Esfuerzo

$PDE_{Ajus}$ : Porcentaje de Distribución Ajustada del Esfuerzo

Finalmente, aplicando estos porcentajes obtenidos al Cronograma Global Estimado, se calcula el cronograma para cada fase de desarrollo.

Ej: Teniendo en cuenta el **Modo de Desarrollo (Modo Orgánico)** y el tamaño del sistema (**8000 SLOC**), seleccionar los porcentajes de distribución nominal del cronograma ( $PDC_{Nom}$ ) y del Esfuerzo ( $PDE_{Nom}$ ), de la Tabla 2 y calcular los nuevos porcentajes de distribución de esfuerzo como en el paso 18 del procedimiento explicado anteriormente.

Fase	PDCNom	PDENom	PDEAjus
Fase Diseño del Producto PD	19%	16%	15%
Fase Programacion DD y CT	59%	65%	69%
Fase Integración y Testeo IT	22%	19%	16%

De acuerdo a lo ya expuesto, los nuevos Porcentajes de Distribución del Cronograma serán:

Fase PD:  $PDC_{Ajus} = 19 \times 15 / 16 = 17.8 \cong \mathbf{18}$

Fase PG:  $PDC_{Ajus} = 59 \times 69 / 65 = 62.6 \cong \mathbf{63}$

Fase IT:  $PDC_{Ajus} = 22 \times 16 / 19 = 18.5 \cong \mathbf{19}$

y para cada fase el Cronograma Global de **7.2 meses** se distribuirá:

Fase PD:  $TDEV_F = 7.2 \times 0.18 = \mathbf{1.3 meses}$

Fase PG:  $TDEV_F = 7.2 \times 0.63 = \mathbf{4.5 meses}$

Fase IT:  $TDEV_F = 7.2 \times 0.19 = \mathbf{1.4 meses}$

## 4 COCOMO II

### 4.1 Definición del modelo

Los objetivos principales que se tuvieron en cuenta para construir el modelo COCOMO II fueron:

- Desarrollar un modelo de estimación de costo y cronograma de proyectos de software que se adaptara tanto a las prácticas de desarrollo de la década del 90 como a las futuras.
- Construir una base de datos de proyectos de software que permitiera la calibración continua del modelo, y así incrementar la precisión en la estimación.
- Implementar una herramienta de software que soportara el modelo.
- Proveer un marco analítico cuantitativo y un conjunto de herramientas y técnicas que evaluaran el impacto de las mejoras tecnológicas de software sobre los costos y tiempos en las diferentes etapas del ciclo de vida de desarrollo.

COCOMO II está compuesto por tres modelos denominados: *Composición de Aplicación*, *Diseño Temprano* y *Post-Arquitectura*.

Éstos surgen en respuesta a la diversidad del mercado actual y futuro de desarrollo de software. Esta diversidad podría representarse con el siguiente esquema (Figura 3).

<b>Aplicaciones desarrolladas por usuarios finales</b>		
<b>Generadores de Aplicaciones</b>	<b>Aplicaciones con Componentes</b>	<b>Sistemas Integrados</b>
<b>Infraestructura</b>		

Figura 3: Distribución del Mercado de Software Actual y Futuro. [Boehm 1995/1]

- **Aplicaciones desarrolladas por Usuarios Finales:** En este sector se encuentran las aplicaciones de procesamiento de información generadas directamente por usuarios finales, mediante la utilización de generadores de aplicaciones tales como planillas de cálculo, sistemas de consultas, etc. Estas aplicaciones surgen debido al uso masivo de estas herramientas, conjuntamente con la presión actual para obtener soluciones rápidas y flexibles.
- **Generadores de Aplicaciones:** En este sector operan firmas como Lotus, Microsoft, Novell, Borland con el objetivo de crear módulos pre-empaquetados que serán usados por usuarios finales y programadores.
- **Aplicaciones con Componentes:** Sector en el que se encuentran aquellas aplicaciones que son específicas para ser resueltas por soluciones pre-empaquetadas, pero son lo suficientemente simples para ser construidas a partir de componentes interoperables. Componentes típicas son constructores de interfases gráficas, administradores de bases de datos, buscadores inteligentes de datos, componentes de dominio-específico (medicina, finanzas, procesos industriales, etc.). Estas aplicaciones son generadas por un equipo reducido de personas, en pocas semanas o meses.
- **Sistemas Integrados:** Sistemas de gran escala, con un alto grado de integración entre sus componentes, sin antecedentes en el mercado que se puedan tomar como base. Porciones de

estos sistemas pueden ser desarrolladas a través de la composición de aplicaciones. Entre las empresas que desarrollan software representativo de este sector, se encuentran grandes firmas que desarrollan software de telecomunicaciones, sistemas de información corporativos, sistemas de control de fabricación, etc.

- **Infraestructura:** Área que comprende el desarrollo de sistemas operativos, protocolos de redes, sistemas administradores de bases de datos, etc. Incrementalmente este sector direccionará sus soluciones, hacia problemas genéricos de procesamiento distribuido y procesamiento de transacciones, a soluciones middleware. Firmas representativas son Microsoft, Oracle, SyBase, Novell y NeXT.

Los tres modelos de COCOMO II se adaptan tanto a las necesidades de los diferentes sectores descritos, como al tipo y cantidad de información disponible en cada etapa del ciclo de vida de desarrollo, lo que se conoce por granularidad de la información.

Se puede afirmar que para las aplicaciones desarrolladas por usuarios finales no se justifica la utilización de un modelo de estimación de costos. Estas aplicaciones normalmente se construyen en poco tiempo, por lo tanto requieren solamente una estimación basada en actividades.

El modelo *Composición de Aplicación*, es el modelo de estimación utilizado en los proyectos de software que se construyen a partir de componentes pre-empaquetadas. En este caso, se emplean Puntos Objeto<sup>5</sup> para estimar el tamaño del software, lo cual está acorde al nivel de información que generalmente se tiene en la etapa de planificación, y el nivel de precisión requerido en la estimación de proyectos de esta naturaleza.

Para los demás sectores del mercado se aplica un modelo mixto, combinación de los tres modelos.

El modelo *Composición de Aplicación* se emplea en desarrollos de software durante la etapa de prototipación.

El modelo *Diseño Temprano* se utiliza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto. En estas etapas se tiene poca información, lo que concuerda con el uso de Puntos Función<sup>6</sup>, para estimar tamaño y el uso de un número reducido de factores de costo.

El modelo *Post-Arquitectura* se aplica en la etapa de desarrollo propiamente dicho, después que se define la arquitectura del sistema, y en la etapa de mantenimiento. Este modelo utiliza:

- Puntos Función y/o Líneas de Código Fuente<sup>7</sup> para estimar tamaño, con modificadores que contemplan el reuso, con y sin traducción automática, y el "desperdicio" (breakage)<sup>8</sup>.
- Un conjunto de 17 atributos, denominados factores de costo<sup>9</sup>, que permiten considerar características del proyecto referentes al personal, plataforma de desarrollo, etc., que tienen injerencia en los costos.
- Cinco factores que determinan un exponente, que incorpora al modelo el concepto de deseconomía y economía de escala<sup>10</sup>. Estos factores reemplazan los modos Orgánico, Semiacoplado y Empotrado del modelo COCOMO '81.

---

<sup>5</sup> Técnica de estimación de tamaño de software, tratada en la sección 4.4.1, página 31.

<sup>6</sup> Técnica de estimación de tamaño de software, tratada en la sección 4.4.2, página 32.

<sup>7</sup> Técnica de estimación de tamaño de software, tratada en la sección 4.4.3, página 35.

<sup>8</sup> Concepto tratado en la sección 4.4.5, página 37.

<sup>9</sup> Conceptos tratados en la sección 4.6, página 46.

## 4.2 Estimación del Esfuerzo

El esfuerzo necesario para concretar un proyecto de desarrollo de software, cualquiera sea el modelo empleado, se expresa en meses/persona (PM) y representa los meses de trabajo de una persona fulltime, requeridos para desarrollar el proyecto.

### 4.2.1 Modelo Composición de Aplicación

La fórmula propuesta en este modelo es la siguiente:

$$PM = NOP / PROD$$

Donde:

**NOP** (Nuevos Puntos Objeto): Tamaño del nuevo software a desarrollar expresado en Puntos Objeto y se calcula de la siguiente manera:

$$NOP = OP \times (100 - \%reuso)/100$$

**OP** (Puntos Objeto): Tamaño del software a desarrollar expresado en Puntos Objeto

**%reuso**: Porcentaje de reuso que se espera lograr en el proyecto

**PROD**: Es la productividad promedio determinada a partir del análisis de datos de proyectos en [Banker 1994], mostrada en Tabla 6.

Experiencia y capacidad de los desarrolladores	Muy Bajo	Bajo	Normal	Alto	Muy Alto
Madurez y Capacidad del ICASE	Muy Bajo	Bajo	Normal	Alto	Muy Alto
<b>PROD</b>	4	7	13	25	50

Tabla 6: Productividad para el modelo Composición de Aplicación. [Boehm 1995/2]

### 4.2.2 Modelo Diseño Temprano

Este modelo se usa en las etapas tempranas de un proyecto de software, cuando se conoce muy poco del tamaño del producto a ser desarrollado, de la naturaleza de la plataforma, del personal a ser incorporado al proyecto o detalles específicos del proceso a utilizar. Este modelo podría emplearse tanto en productos desarrollados en sectores de Generadores de Aplicación, Sistemas Integrados o Infraestructura.

El modelo de Diseño Temprano ajusta el esfuerzo nominal usando siete factores de costo. La fórmula para el cálculo del esfuerzo es la siguiente:

$$PM_{estimado} = PM_{nominal} \times \prod_{i=1}^7 EM_i$$

<sup>10</sup> Conceptos tratados en la sección 4.5, página 42.

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

$$B = 1.01 + 0.01 \times \sum_{j=1}^5 W_j$$

Donde:

- $PM_{Estimado}$  es el esfuerzo Nominal ajustado por 7 factores, que reflejan otros aspectos propios del proyecto que afectan al esfuerzo necesario para la ejecución del mismo.
- $KSLOC$  es el tamaño del software a desarrollar expresado en miles de líneas de código fuente.
- $A$  es una constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, ( $A=2.94$ ).
- $B$  es el factor exponencial de escala, toma en cuenta las características relacionadas con las economías y diseconomías de escala producidas cuando un proyecto de software incrementa su tamaño. Ver sección 4.5, página 42.
- $EM_i$  corresponde a los factores de costo que tienen un efecto multiplicativo sobre el esfuerzo, llamados Multiplicadores de Esfuerzo (Effort Multipliers). Cada factor se puede clasificar en seis niveles diferentes que expresan el impacto del multiplicador sobre el esfuerzo de desarrollo. Esta escala varía desde un nivel Extra Bajo hasta un nivel Extra Alto. Cada nivel tiene un peso asociado. El peso promedio o nominal es 1.0. Si el factor provoca un efecto nocivo en el esfuerzo de un proyecto, el valor del multiplicador correspondiente será mayor que 1.0, caso contrario el multiplicador será inferior a 1.0. La Figura 4 muestra una pantalla del software COCOMO II.1999.0, donde se aprecian los valores de los factores de acuerdo a cada nivel, según la calibración efectuada para el año 1999.

Clasificados en categorías, los 7 Multiplicadores de Esfuerzo son:

Del Producto

RCPX: Confiabilidad y Complejidad del producto

RUSE: Reusabilidad Requerida

De la Plataforma

PDIF: Dificultad de la Plataforma

Del Personal

PERS: Aptitud del Personal

PREX: Experiencia del Personal

Del Proyecto

FCIL: Facilidades

SCED: Cronograma de Desarrollo Requerido

	XLO	VLO	LO	NOM	HI	VHI	XHI
RCPX	0.73	0.81	0.98	1.00	1.30	1.74	2.38
RUSE	XXXX	XXXX	0.95	1.00	1.07	1.15	1.24
PDIF	XXXX	XXXX	0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	XXXX	1.43	1.14	1.00	1.00	1.00	XXXX
USR1	XXXX	1.00	1.00	1.00	1.00	1.00	XXXX
USR2	XXXX	1.00	1.00	1.00	1.00	1.00	XXXX

Figura 4: Multiplicadores de Esfuerzo del Modelo de Diseño Temprano. [COCOMO II.0]<sup>11</sup>

#### 4.2.3 Modelo Post-Arquitectura

Es el modelo de estimación más detallado y se aplica cuando la arquitectura del proyecto está completamente definida. Este modelo se aplica durante el desarrollo y mantenimiento de productos de software incluidos en las áreas de Sistemas Integrados, Infraestructura y Generadores de Aplicaciones.

El esfuerzo nominal se ajusta usando 17 factores multiplicadores de esfuerzo. El mayor número de multiplicadores permite analizar con más exactitud el conocimiento disponible en las últimas etapas de desarrollo, ajustando el modelo de tal forma que refleje fielmente el producto de software bajo desarrollo. La fórmula para el cálculo del esfuerzo es la siguiente:

$$PM_{estimado} = PM_{nominal} \times \prod_{i=1}^{17} EM_i$$

Los 17 factores de costo correspondientes a este modelo se explicarán en detalle en la sección 4.6., página 46.

#### 4.3 Estimación del Cronograma

La versión inicial de COCOMO II provee un modelo de estimación del cronograma similar al presentado en COCOMO' 81 y ADA COCOMO. La ecuación inicial para los tres modelos de COCOMO II es:

<sup>11</sup> El Software COCOMO II.1999.0 permite el uso de dos factores del usuario (USR1, USR2) para poder contemplar particularidades de cada proyecto.

$$TDEV = \left[ 3.0 \times PM_*^{(0.33+0.2 \times (B-1.01))} \right] \times \frac{SCED\%}{100}$$

Donde:

**TDEV** es el tiempo calendario en meses que transcurre desde la determinación de los requerimientos a la culminación de una actividad que certifique que el producto cumple con las especificaciones.

**PM\*** es el esfuerzo expresado en meses personas, calculado sin tener en cuenta el multiplicador de esfuerzo **SCED**. Ver Tabla 21.

**B** es el Factor de Escala

**SCED%** es el porcentaje de compresión/expansión del cronograma.

Las futuras versiones de COCOMO II ofrecerán un modelo de estimación de cronograma más completo que refleje los diferentes modelos de procesos que se puede usar en el desarrollo de un proyecto, los efectos del reuso de software y la composición de aplicaciones.

#### 4.4 Métricas de Software

En la estimación del tamaño de software COCOMO II utiliza tres técnicas: *Puntos Objeto*, *Puntos Función No Ajustados* y *Líneas de Código Fuente*. Además se emplean otros parámetros relativos al tamaño que contemplan aspectos tales como: reuso, reingeniería, conversión, y mantenimiento.

Es necesario unificar criterios de medición de tamaño, tanto para poder planificar y controlar proyectos, como para realizar estudios y análisis entre proyectos en pro de la mejora de procesos [Park 1992].

##### 4.4.1 Puntos Objeto

A pesar de que la estimación a través de Puntos Objeto es un enfoque de medición de tamaño de software relativamente nuevo, es apropiado para las aplicaciones con componentes y para estimar esfuerzos en las etapas de prototipación. En estas circunstancias, se lo ha comparado con la estimación de Puntos Función. Un experimento, diseñado por Kaufman y Kumar en 1993, involucró a 4 administradores de proyecto experimentados usando Puntos Objeto y Puntos Función, para estimar el esfuerzo requerido de dos proyectos terminados en 3.5 y 6 meses-persona respectivamente. Como base, se emplearon las descripciones disponibles al comienzo de tales proyectos. El experimento permitió determinar que:

- Los Puntos Objeto y los Puntos Función produjeron resultados igualmente precisos (ligeramente más exacto con Puntos Objetos, pero no estadísticamente significativo).
- El tiempo promedio para producir una estimación con Puntos Objeto fue alrededor del 47% del tiempo promedio necesario para las estimaciones con Puntos Función. Además, los administradores consideraron que el método de Puntos Objeto era más fácil de usar.

De esta manera, aunque estos resultados no están respaldados estadísticamente, parecen suficientemente prometedores como para justificar el uso de Puntos Objeto como punto de partida en el modelo de estimación de *Composición de Aplicación* de COCOMO II.

A continuación se describe el procedimiento para determinar Puntos Objeto en un proyecto de software:

**Primero:** Determinar Cantidad de Objetos: Estimar la cantidad de pantallas, reportes, componentes de 3GL que contendrá la aplicación.

**Segundo:** Clasificar cada instancia de un objeto según sus niveles de complejidad (simple, media o difícil) de acuerdo a la Tabla 7.

**Tercero:** Dar el peso a cada objeto según el nivel de complejidad. Los pesos reflejan el esfuerzo relativo requerido para implementar una instancia de ese nivel de complejidad. Tabla 8.

**Cuarto:** Determinar la cantidad de Puntos Objeto, sumando todos los pesos de las instancias de los tipos de objetos especificados.

Para Pantallas			
Cantidad de Vistas Contenidas	Cantidad y fuente de las tablas de datos		
	Total < 4 ( < 2 servidor < 3 cliente)	Total < 8 ( < 2 - 3 servidor < 3 - 5 cliente)	Total 8 + ( > 3 servidor < 5 cliente)
< 3	Simple	Simple	Media
3 - 7	Simple	Media	Difícil
> 8	Media	Difícil	Difícil
Para Reportes			
Cantidad de Vistas Contenidas	Cantidad y fuente de las tablas de datos		
	Total < 4 ( < 2 servidor < 3 cliente)	Total < 8 ( < 2 - 3 servidor < 3-5 cliente)	Total 8 + ( > 3 servidor < 5 cliente)
0 o 1	Simple	Simple	Media
2 o 3	Simple	Media	Difícil
4 +	Media	Difícil	Difícil

Tabla 7: Esquema de Clasificación de Puntos Objetos. [Boehm 1995/2]

Tipo de Objeto	Complejidad - Peso		
	Simple	Media	Difícil
Pantalla	1	2	3
Reporte	2	5	8
Componente 3GL			10

Tabla 8: Peso de un Punto Objeto. [Boehm 1995/2]

#### 4.4.2 Puntos Función

El modelo COCOMO II usa Puntos Función y/o Líneas de Código Fuente (SLOC) como base para medir tamaño en los modelos de estimación de *Diseño Temprano* y *Post-Arquitectura*

Las métricas para puntos función están basadas en las guías proporcionadas por el "International Function Point User Group"-IFPUG [IFPUG 1994][Behrens 1983][Kunkler 1985].

Los Puntos Función procuran cuantificar la funcionalidad de un sistema de software. La meta es obtener un número que caracterice completamente al sistema. Son útiles estimadores ya que están basados en información que está disponible en las etapas tempranas del ciclo de vida del desarrollo de software. COCOMO II considera solamente UFP (Puntos Función no ajustados).

La fórmula de Albretch [Albretch 1979] para calcular los puntos función, es la siguiente:

$$FP = UFP \times TCF$$

Donde **UFP**: Puntos Función no Ajustados

**TCF**: Factor de Complejidad Técnica

Para calcular los UFP, se deben identificar los siguientes tipos de ítems:

- Entradas Externas (Inputs): Entrada de datos del usuario o de control que ingresan desde el exterior del sistema para agregar y/o cambiar datos a un archivo lógico interno.
- Salidas Externas (Outputs): Salida de datos de usuario o de control que deja el límite del sistema de software.
- Archivo Lógicos Internos (Archivos): Incluye cada archivo lógico, es decir cada grupo lógico de datos que es generado, usado, o mantenido por el sistema de software.
- Archivos Externos de Interfase (Interfases): Archivos transferidos o compartidos entre sistemas de software.
- Solicitudes Externas (Queries): Combinación única de entrada-salida, donde una entrada causa y genera una salida inmediata, como un tipo de solicitud externa.

Una vez identificados los ítems se clasifican de acuerdo al grado de complejidad en: bajo, promedio o alto. Se asigna un peso a cada ítem según el tipo y el grado de complejidad correspondiente. Finalmente los UFP son calculados mediante la sumatoria de los pesos de todos los ítems identificados.

$$UFP = \sum_{i=1}^{15} (Cantidad\_Items\_Tipo_i) \times (Peso_i)$$

La Tabla 9 muestra como se determinan los niveles de complejidad de cada tipo de ítem en función del número y tipo de elementos de datos y archivos involucrados.

Para archivos lógicos internos y archivos externos de interfase				Para salidas y consultas externas				Para entradas externas			
Elementos de Registro	Elementos de datos			Tipos de archivos	Elementos de datos			Tipos de archivos	Elementos de datos		
	1-19	20-50	51+		1-5	6-19	20+		1-4	5-15	16+
1	Bajo	Bajo	Prom.	0 ó 1	Bajo	Bajo	Prom.	0 ó 1	Bajo	Bajo	Prom.
2-5	Bajo	Prom.	Alto	2-3	Bajo	Prom.	Alto	2-3	Bajo	Prom.	Alto
6+	Prom.	Alto	Alto	4+	Prom.	Alto	Alto	3+	Prom.	Alto	Alto

Tabla 9: Puntos Función. Determinación del Peso. [Boehm 1995/2]

La Tabla 10 muestra las ponderaciones asociadas a cada tipo de ítem. Estas ponderaciones han sido derivadas y validadas empíricamente mediante la observación de una gran variedad de proyectos.

Tipo de función	Peso del Factor de Complejidad		
	Bajo	Promedio	Alto
<b>Entradas Externas (Inputs)</b>	3	4	6
<b>Salidas Externas (Outputs)</b>	4	5	7
<b>Archivo Lógicos Internos (Archivos)</b>	7	10	15
<b>Archivos Externos de Interfase (Interfases)</b>	5	7	10
<b>Consultas Externas (Queries)</b>	3	4	6

Tabla 10: Peso del Factor de Complejidad. [Boehm 1995/2]

Para el cálculo del Factor de Complejidad Técnica, TCF, se considera la siguiente fórmula:

$$TCF = 0.65 + 0.01 \times \sum_{i=1}^{14} F_i$$

Donde los  $F_i$  corresponden a los pesos asignados a los siguientes factores:

F1: Mecanismos de recuperación y back-up confiables

F2: Comunicación de Datos

F3: Funciones de Procesamiento Distribuido

F4: Performance

F5: Configuración usada rigurosamente

F6: Entrada de datos on-line

F7: Factibilidad Operativa

F8: Actualización de archivos on-line

F9: Interfases Complejas

F10: Procesamiento Interno Complejo

F11: Reusabilidad

F12: Fácil Instalación

F13: Soporte de múltiples instalaciones

F14: Facilidad de cambios y amigabilidad

Los pesos se consideran dentro de una escala de 0 a 5, descripta a continuación:

0: Sin influencia

1: Incidental

- 2: Moderado
- 3: Medio
- 4: Significativo
- 5: Esencial

Estas 14 características consideran aspectos como reusabilidad, performance, complejidad, confiabilidad, etc., contemplados por COCOMO II a través de los factores de costo. Es por ello que este modelo utiliza los UFP como métrica de determinación de tamaño.

#### **4.4.3 Líneas de Código Fuente**

COCOMO II considera a la sentencia fuente lógica como línea standard de código. Ahora bien, definir una línea de código es difícil debido a que existen diferencias conceptuales cuando se cuentan sentencias ejecutables y de declaraciones de datos en lenguajes diferentes. El objetivo es medir la cantidad de trabajo intelectual puesto en el desarrollo de un programa.

Para minimizar esos problemas, se usa el checklist de definición desarrollado por el SEI, que permite unificar criterios en la definición de una línea de código fuente [Park 1992], [Goethert et al. 1992]. Ver Figura 5. A los efectos de COCOMO II, se han efectuado cambios que consisten en eliminar las categorías de software que insumen poco esfuerzo. Así no están incluidas librerías de soporte de lenguajes, sistemas operativos, librerías comerciales, etc., ni tampoco el código generado con generadores de código fuente.

Existen herramientas automatizadas para medir la cantidad de líneas de código fuente, como por ejemplo Amadeus [Amadeus 1994] [Selby et al. 1991]. Para realizar un análisis de mayor especificidad, Amadeus automáticamente recolecta medidas adicionales como total de líneas fuente, de comentarios, declaraciones, interfases, anidamientos, sentencias ejecutables y otras. Esta herramienta provee varias medidas de tamaño, incluyendo métricas aplicables a tecnologías de objetos de [Chidamber and Kemerer 1994].

### Definition Checklist for Source Statements Counts

Definition name: Logical Source Statements Date: \_\_\_\_\_  
 (basic definition) Originator: COCOMO II

Measurement unit	Physical source lines				
	Logical source statements	<input checked="" type="checkbox"/>			
<b>Statement type</b>	Definition <input checked="" type="checkbox"/>	Data Array <input type="checkbox"/>		Includes	Excludes
When a line or statement contains more than one type, classify it as the type with the highest precedence.					
1 Executable	Order of precedence →		1	<input checked="" type="checkbox"/>	
2 Nonexecutable					
3 Declarations			2	<input checked="" type="checkbox"/>	
4 Compiler directives			3	<input checked="" type="checkbox"/>	
5 Comments					
6 On their own lines			4		<input checked="" type="checkbox"/>
7 On lines with source code			5		<input checked="" type="checkbox"/>
8 Banners and non-blank spaces			6		<input checked="" type="checkbox"/>
9 Blank (empty) comments			7		<input checked="" type="checkbox"/>
10 Blank lines			8		<input checked="" type="checkbox"/>
11					
12					
<b>How produced</b>	Definition <input checked="" type="checkbox"/>	Data array <input type="checkbox"/>		Includes	Excludes
1 Programmed				<input checked="" type="checkbox"/>	
2 Generated with source code generators					<input checked="" type="checkbox"/>
3 Converted with automated translators				<input checked="" type="checkbox"/>	
4 Copied or reused without change				<input checked="" type="checkbox"/>	
5 Modified				<input checked="" type="checkbox"/>	
6 Removed					<input checked="" type="checkbox"/>
7					
8					
<b>Origin</b>	Definition <input checked="" type="checkbox"/>	Data array <input type="checkbox"/>		Includes	Excludes
1 New work: no prior existence				<input checked="" type="checkbox"/>	
2 Prior work: taken or adapted from					
3 A previous version, build, or release				<input checked="" type="checkbox"/>	
4 Commercial, off-the-shelf software (COTS), other than libraries					<input checked="" type="checkbox"/>
5 Government furnished software (GFS), other than reuse libraries					<input checked="" type="checkbox"/>
6 Another product					<input checked="" type="checkbox"/>
7 A vendor-supplied language support library (unmodified)					<input checked="" type="checkbox"/>
8 A vendor-supplied operating system or utility (unmodified)					<input checked="" type="checkbox"/>
9 A local or modified language support library or operating system					<input checked="" type="checkbox"/>
10 Other commercial library					<input checked="" type="checkbox"/>
11 A reuse library (software designed for reuse)				<input checked="" type="checkbox"/>	
12 Other software component or library				<input checked="" type="checkbox"/>	
13					
14					

Figura 5: Checklist para la definición de una línea de código fuente. [COCOMO II.0]

#### 4.4.4 Conversión de Puntos Función a Líneas de Código Fuente (SLOC)

Para determinar el esfuerzo nominal en el modelo COCOMO II los puntos función no ajustados tienen que ser convertidos a líneas de código fuente considerando el lenguaje de implementación (assembler, lenguajes de alto nivel, lenguajes de cuarta generación, etc.). Esto se realiza para los modelos Diseño Temprano y Post Arquitectura teniendo en cuenta la Tabla 11 propuesta por Jones.

Lenguaje	SLOC/ Pto. Función
Ada	71
AI Shell	49
APL	32
Assembler	320
Assembler (macro).	213
ANSI/Quick/Turbo Basic.	64
Basic - Compilado	91
Basic – Interpretado	128
C	128
C++	29
ANSI Cobol 85	91
Fortran 77	105
Forth	64
Jovial	105
Lisp	64
Modula 2	80
Pascal	91
Prolog	64
Generador de Reportes	80
Planilla de Cálculo	6

Tabla 11: Conversión de UFP a SLOC. [COCOMO II.0]

#### 4.4.5 Desperdicio de Código (Breakage)

Se considera como *Desperdicio* al porcentaje de código que se debe eliminar debido a la volatilidad de los requerimientos. Por ejemplo, un proyecto con 100.000 instrucciones liberadas que descartó el equivalente de 20.000 instrucciones tiene un valor de *Desperdicio (BRAK)* del 20%. Éste se usa para ajustar el tamaño efectivo del software a ser desarrollado a los efectos del proceso de estimación. De este modo la ecuación del esfuerzo nominal modificada, para contemplar este aspecto, es la siguiente:

$$PM_{nominal} = A \times \left[ \left( 1 + \frac{BRAK}{100} \right) \times KSLOC \right]^B$$

#### 4.4.6 Modelo de Reuso

COCOMO II usa un modelo no lineal para estimar el tamaño del software cuando éste incluye componentes reusables. El análisis de 3.000 casos de reuso de módulos realizado en el Laboratorio

de Ingeniería de Software de la NASA indica que el costo asociado al reuso es una función no lineal debido a dos razones [Selby 1988]:

- Existe un costo base, de alrededor de un 5%, que contempla la evaluación, selección, y asimilación del componente reusable.
- Pequeñas modificaciones generan desproporcionadamente grandes costos. Esto se debe al esfuerzo por comprender el software a ser modificado, testear y chequear las interfases.

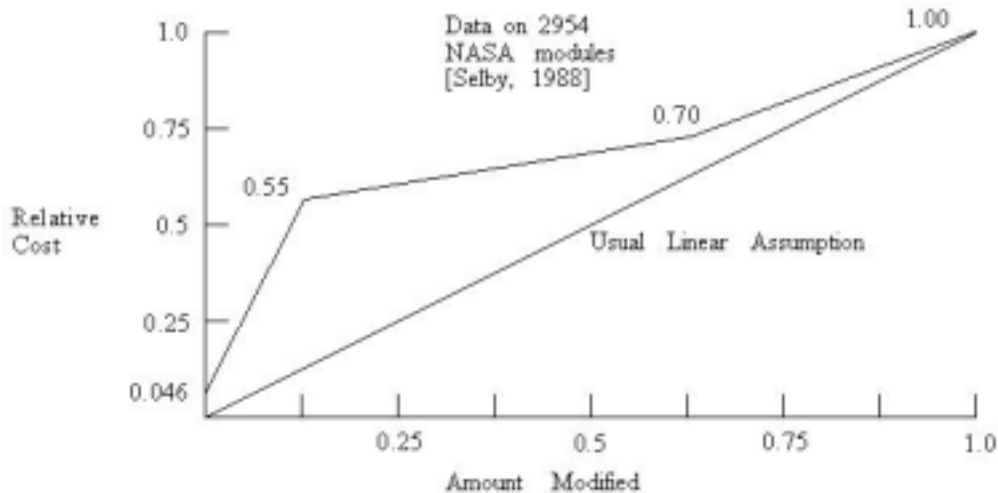


Figura 6: Efectos no lineales del reuso

En [Parikh and Zvegintzov 1983] se indica que el 47% del esfuerzo en el mantenimiento de software está relacionado con la tarea que implica entender el software que va a ser modificado. Además, en [Gerlich and Denskat 1994] se muestra que existe un efecto no lineal en el costo del reuso debido al chequeo de interfases que debe realizarse durante el desarrollo del software modificado. Estos inconvenientes pueden reducirse si el software está apropiadamente estructurado.

El modelo COCOMO II permite tener en cuenta si un proyecto de software va a ser construido a partir de componentes existentes. Para ello, reemplaza en la ecuación de estimación de esfuerzo el parámetro *KSLOC* por el *KESLOC*, que representa la cantidad equivalente de nuevas líneas de código a desarrollar.

*ESLOC* se calcula de la siguiente forma:

$$AAF = 0.4(DM) + 0.3(CM) + 0.3(IM)$$

$$ESLOC = \frac{ASLOC[AA + AAF(1 + 0.02(SU)(UNFM))]}{100}, AAF \leq 0.5$$

$$ESLOC = \frac{ASLOC[AA + AAF + (SU)(UNFM)]}{100}, AAF > 0.5$$

Donde:

*ASLOC* Cantidad de líneas de código fuente del software existente usadas para desarrollar el nuevo producto

*DM* Porcentaje del diseño del software que requiere modificación para alcanzar los objetivos del nuevo software a desarrollar

- CM** Porcentaje del código del software que requiere modificación para lograr los objetivos del nuevo software a desarrollar
- IM** Porcentaje del esfuerzo requerido para integrar y testear el software adaptado al producto global
- SU** Porcentaje de comprensibilidad del software existente. Se determina en función a tres características: estructura, claridad y descriptividad. Ver Tabla 12.
- AA** Grado de Evaluación y Asimilación. Porcentaje de esfuerzo necesario para determinar si un módulo de software a adaptar es apropiado a la aplicación, como así también para integrar su descripción a la descripción total del producto. Ver Tabla 13.
- UNFM** Nivel de familiaridad del programador con el software. Ver Tabla 14.

	<b>Muy Bajo</b>	<b>Bajo</b>	<b>Nominal</b>	<b>Alto</b>	<b>Muy Alto</b>
<b>Estructura</b>	Cohesión muy baja, alto acoplamiento, código espagueti	Cohesión moderadamente Baja, alto acoplamiento	Razonablemente bien estructurado, Algunas áreas débiles	Alta cohesión, bajo acoplamiento	Fuerte modularidad Ocultamiento de la implementación
<b>Claridad en la aplicación</b>	Ninguna correspondencia con el dominio de aplicación	Alguna correspondencia con el dominio	Moderada correspondencia con el dominio	Buena correspondencia con el dominio	Clara correspondencia con el dominio
<b>Descriptividad</b>	Código oscuro, documentación faltante, oscura, u obsoleta	Algunas líneas comentario, y alguna documentación útil	Nivel moderado de líneas comentario, y documentación	Buen nivel de líneas comentario, y documentación útil; debilidad en algunas áreas	Código autodescriptivo, documentación al día, bien organizada con racional diseño
<b>SU</b>	<b>50</b>	<b>40</b>	<b>30</b>	<b>20</b>	<b>10</b>

Tabla 12: Niveles del Incremento por Entendimiento del Software (SU). [COCOMO II.0]

<b>Incremento AA</b>	<b>Nivel de esfuerzo de AA</b>
0	Ninguno
2	Búsqueda del módulo básico y documentación
4	Algunos módulos de testeo y evaluación, documentación
6	Considerable cantidad de módulos de testeo y evaluación, documentación
8	Gran cantidad de módulos de testeo y evaluación, documentación

Tabla 13: Niveles del Incremento por Asimilación y Evaluación (AA). [COCOMO II.0]

UNFM Incremento	Nivel de Desconocimiento
0.0	Completamente familiar
0.2	Familiar en su mayor parte
0.4	Algo familiar
0.6	Considerablemente familiar
0.8	Desconocido en su mayor parte
1.0	Completamente Desconocido

Tabla 14: Niveles del Incremento por Desconocimiento (UNFM). [COCOMO II.0]

#### 4.4.7 Reingeniería y Conversión

El modelo de Reuso de COCOMO II necesita un refinamiento adicional para estimar el costo de reingeniería y de conversión. La principal diferencia entre reingeniería y conversión está dada por la eficiencia de las herramientas automatizadas utilizadas para reestructurar el software. Pueden producirse situaciones en las que a un muy alto nivel de porcentaje de código a modificar (*CM*) le corresponda un bajo nivel de esfuerzo. En un caso de estudio de reingeniería analizado en [Ruhl and Gunn 1991], el 80% del código (13.131 sentencias fuentes COBOL) fue reutilizado por medio de la traducción automática con un esfuerzo real de 35 meses/personas, casi cuatro veces menos de lo que se estimó usando el modelo de COCOMO II (152 meses/persona).

El enfoque de reingeniería y conversión involucra la estimación de un nuevo parámetro *AT*, que representa el porcentaje de código que sufre un proceso de reingeniería mediante el uso de una herramienta de traducción automática. Del análisis de los datos del proyecto anterior surge que la productividad de la traducción automática es de 2400 líneas de código fuente por mes-persona, este valor podrá variar con las diferentes tecnologías y es designado en el modelo COCOMO II como *ATPROD*.

La ecuación siguiente muestra como afecta el reuso y la traducción automática a la estimación del esfuerzo nominal:

$$PM_{no\ min\ al} = A \times (KSLOC)^B + \frac{ASLOC \times \frac{AT}{100}}{\underbrace{ATPROD}_1}$$

$$KSLOC = KNSLOC + \left[ KASLOC \times \underbrace{\left( \frac{100 - AT}{100} \right)}_2 \times \frac{(AA + AAF \times (1 + 0.02 \times SU \times UNFM))}{100} \right], AAF \leq 0.5$$

$$KSLOC = KNSLOC + \left[ KASLOC \times \left( \frac{100 - AT}{100} \right) \times \frac{(AA + AAF + SU \times UNFM)}{100} \right], AAF > 0.5$$

Donde:

- PM<sub>Nominal</sub>* Esfuerzo expresado en meses personas
- B* Factor de Escala
- A* Constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, (*A=2.94*)

<b><i>KSLOC</i></b>	Tamaño del software a desarrollar
<b><i>KASLOC</i></b>	Tamaño del software a adaptar
<b><i>KNSLOC</i></b>	Tamaño del software a desarrollar desde cero
<b><i>AT</i></b>	Porcentaje de código que sufre un proceso de reingeniería mediante el uso de una herramienta de traducción automática.
<b><i>ATPROD</i></b>	Productividad de la herramienta utilizada en la traducción automática
<b><i>SU</i></b>	Porcentaje de comprensibilidad del software existente. Se determina en función a tres características: estructura, claridad y descriptividad. Ver Tabla 12
<b><i>AA</i></b>	Grado de Evaluación y Asimilación. Porcentaje de esfuerzo necesario para determinar si un módulo de software a adaptar es apropiado a la aplicación, como así también para integrar su descripción en la descripción total del producto. Ver Tabla 13
<b><i>UNFM</i></b>	Nivel de familiaridad del programador con el software. Ver Tabla 14
<b><i>1</i></b>	Término que representa el esfuerzo asociado a la traducción automática
<b><i>2</i></b>	Porcentaje de código que se adapta sin el uso de una herramienta automatizada

Considerando un módulo original de 5000 SLOC, y teniendo en cuenta los valores de los parámetros de la Figura 7, la ecuación anterior determina que el tamaño del nuevo módulo a considerar en la estimación de esfuerzo, es de 1731 SLOC.

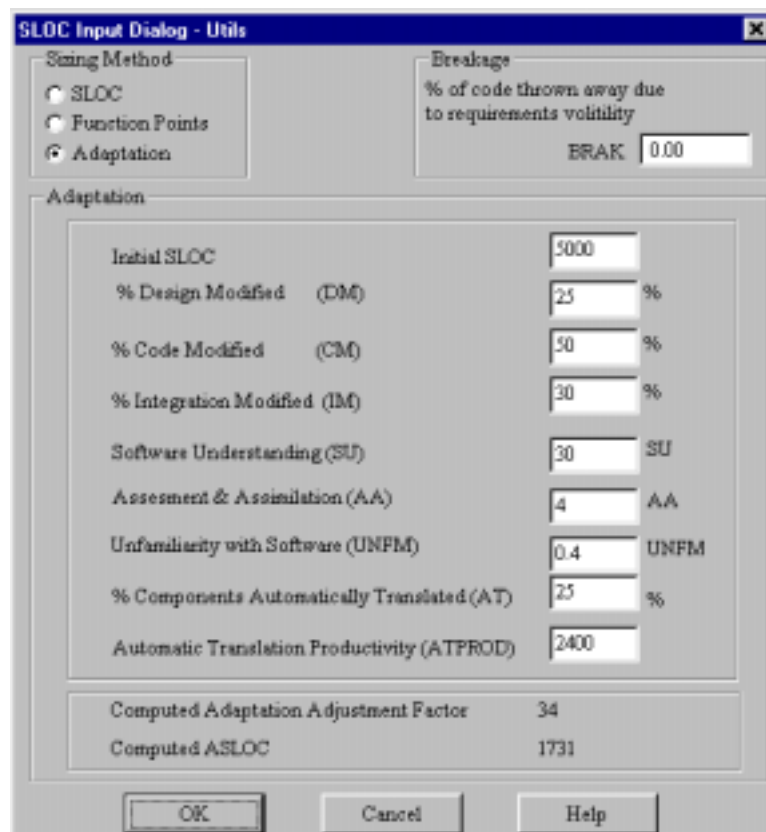


Figura 7: Cómputo de un módulo adaptado. [COCOMO II.0]

como AAF=0.34, se aplica la primera ecuación:

$$KSLOC = 0 + \left[ 5000 \times \left( \frac{100 - 25}{100} \right) \times \frac{(4 + 34 \times (1 + 0.02 \times 30 \times 0.4))}{100} \right] = 1731$$

#### 4.5 Factor Exponencial de Escala

Los modelos de estimación de costos analizan dos aspectos antagónicos que influyen notablemente en los procesos de estimación, la economía y deseconomía de escala. La economía de escala abarca factores que hacen más eficiente la producción de software en gran escala. Es frecuente lograr economía en proyectos de gran envergadura, gracias a la inversión en software de propósitos específicos que mejoran la productividad, tales como herramientas de testeo, librerías de programas, preprocesadores, postprocesadores. Ahora bien, estamos frente a una deseconomía de escala cuando al incrementarse el tamaño del producto se produce una considerable disminución de la productividad. El aumento de la cantidad de personas que conforman el equipo de desarrollo generalmente provoca problemas de integración, que sumados a los conflictos personales, las diferencias en la filosofía y hábitos de trabajos producen deseconomía de escala.

Los modelos de estimación de costos frecuentemente tienen un factor exponencial para considerar las economías y deseconomías de escala. En particular, COCOMO II captura esos efectos en el exponente **B**:

$$B = 1.01 + 0.01 \times \sum_{j=1}^5 W_j$$

Si **B** < 1.0, el proyecto exhibe economía de escala. Es decir si un producto aumenta el doble su tamaño el esfuerzo del proyecto es menos del doble. Esto significa que la productividad del proceso de desarrollo de software incrementa a medida que aumenta el tamaño del proyecto.

Si el **B** = 1.0 las economías y deseconomías de escala están en equilibrio. Este modelo lineal se usa siempre en la estimación de costos de proyectos pequeños.

Si el **B** > 1.0 el proyecto muestra deseconomía de escala. Esto generalmente se debe a dos factores principales: el crecimiento de las comunicaciones interpersonales y el de la integración de sistemas. Integrar un producto pequeño como parte de otro requiere no sólo el esfuerzo de desarrollar el producto sino también el esfuerzo de diseñar, mantener, integrar y testear interfaces con el resto del software. La productividad del proceso de desarrollo de software disminuye a medida que aumenta el tamaño del proyecto.

El cálculo del Factor Exponencial de Escala **B** está basado en factores que influyen exponencialmente en la productividad y esfuerzo de un proyecto de software. Estos factores toman valores dentro de un rango que va desde un nivel *Muy Bajo* hasta uno *Extra Alto*, tal como muestra la Tabla 15. Cada nivel tiene un peso asociado **W<sub>j</sub>**, y ese valor específico es el que se denomina factor de escala. En la Figura 8 se observan los pesos de cada factor según el nivel, considerados por el software COCOMO II.1999.0.

Factor de Escala Wj	Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
<b>Precedencia PREC</b>	Completamente sin precedentes	Ampliamente sin precedentes	Algún precedente	Generalmente Familiar	Ampliamente Familiar	Completamente Familiar
<b>Flexibilidad en el desarrollo FLEX</b>	Rigurosa	Relajación Ocasional	Alguna Relajación	Conformidad en general	Alguna Conformidad	Metas generales
<b>Arquitectura/ Resolución de riesgo RESL</b>	Poca (20%)	Alguna (40%)	Siempre (60%)	Generalmente 75%)	Principalmente (90%)	Completo (100%)
<b>Cohesión de equipo TEAM</b>	Interacciones difíciles	Interacciones con alguna dificultad	Interacciones básicamente cooperativas	Ampliamente Cooperativas	Altamente Cooperativas	Interacciones Sin Fisuras
<b>Madurez del proceso PMAT</b>	Desarrollado más adelante					

Tabla 15: Factores de Escala. [Boehm 1995/2]

	VLO	LO	NOM	HI	VHI	XHI
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Figura 8: Factores de Escala. [COCOMO II.0]

#### 4.5.1 Precedencia y Flexibilidad en el Desarrollo (PREC Y FLEX)

El factor de precedencia (**PREC**) toma en cuenta el grado de experiencia previa en relación al producto a desarrollar, tanto en aspectos organizacionales como en el conocimiento del software y hardware a utilizar.

El factor de flexibilidad (**FLEX**) considera el nivel de exigencia en el cumplimiento de los requerimientos preestablecidos, plazos de tiempos y especificaciones de interfase.

El modelo COCOMO II presenta la Tabla 16, en la cual se detallan las siete características a analizar para encontrar el peso de los factores **PREC** y **FLEX**.

Características	Muy Bajo Bajo	Nominal Alto	Extra Alto Muy Alto
<b>Precedencia</b>			
Entendimiento organizacional de los objetivos del producto	General	Considerable	Total
Experiencia en el trabajo con software relacionado	Moderada	Considerable	Amplia
Desarrollo concurrente de nuevo hardware y procedimientos operacionales	Abundante	Moderado	Escaso
Necesidad de innovación en el procesamiento de datos, arquitectura y algoritmos	Considerable	Alguna	Mínima
<b>Flexibilidad en el desarrollo</b>			
Necesidad de conformar requerimientos pre-establecidos	Total	Considerable	Básica
Necesidad de conformar especificaciones externas de interfase	Total	Considerable	Básica
Estímulo por terminación temprana	Elevado	Medio	Bajo

Tabla 16: Factores de Escala relacionados al modo de desarrollo de COCOMO. [COCOMO II.0]

#### 4.5.2 Arquitectura y Determinación del Riesgo (RESL)

Este factor involucra aspectos relacionados al conocimiento de los ítems de riesgo crítico y al modo de abordarlos dentro del proyecto.

El nivel del factor **RESL** es el resultado de un promedio de los niveles de las características listadas en la Tabla 17.

Características	Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Planificación de la administración de riesgo, identificando todos los ítems de riesgo y estableciendo hitos de control para su solución por medio de la revisión del diseño del producto (PDR)	Ninguna	Pequeña	Algo	General	En gran medida	Completa
Cronograma, presupuesto e hitos internos especificados en el PDR, compatibles con el plan de administración de riesgo	Ninguno	Pequeño	Algo	General	En gran medida	Completo
Porcentaje del cronograma dedicado a la definición de la arquitectura de acuerdo a los objetivos generales del producto	5	10	17	25	33	40
Porcentaje de arquitecturas de software disponibles para el proyecto	20	40	60	80	100	120
Herramientas disponibles para resolver ítems de riesgo, desarrollando y verificando las especulaciones de arquitecturas	Ninguna	Pocas	Algunas	Buenas	Muy Buenas	Completas

Nivel de incertidumbre en factores claves de la arquitectura: interfase de usuario, hardware, tecnología, performance	Extremo	Significativo	Considerable	Medio	Poco	Muy Poco
Cantidad y grado de criticidad de ítems de riesgo	> 10 Crítico	5 - 10 Crítico	2 - 4 Crítico	1 Crítico	> 5 No Crítico	< 5 No Crítico

Tabla 17: Componentes para calcular el factor de escala RESL. [COCOMO II.0]

#### 4.5.3 Cohesión del Equipo (TEAM)

El factor de escala denominado Cohesión del Equipo tiene en cuenta las dificultades de sincronización entre los participantes del proyecto: usuarios, clientes, desarrolladores, encargados de mantenimiento, etc. Estas dificultades pueden surgir por diferencias culturales, dificultad en la conciliación de objetivos, falta de experiencia y familiaridad con el trabajo en equipo. El valor del factor **TEAM** se calcula como un promedio ponderado de las características listadas en Tabla 18.

Características	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
Compatibilidad entre los objetivos y culturas de los integrantes del equipo	Poca	Alguna	Básica	Considerable	Fuerte	Total
Habilidad y predisposición para conciliar objetivos	Poca	Alguna	Básica	Considerable	Fuerte	Total
Experiencia en el trabajo en equipo	Ninguna	Poca	Poca	Básica	Considerable	Vasto
Visión compartida de objetivos y compromisos compartidos	Ninguna	Poca	Poca	Básica	Considerable	Amplia

Tabla 18: Componentes del factor TEAM. [COCOMO II.0]

#### 4.5.4 Madurez del Proceso (PMAT)

El procedimiento para determinar el factor **PMAT** se basa en el Modelo de CMM propuesto por el Software Engineering Institute. Existen dos formas de calcularlo:

- La primera captura el nivel de madurez de la organización, resultado de la evaluación según CMM y asignándole el valor correspondiente según Tabla 19:

Nivel de CMM	PMAT
1 – Mitad inferior	Muy Bajo
1 – Mitad superior	Bajo
2	Nominal
3	Alto
4	Muy Alto
5	Extra Alto

Tabla 19: Factor PMAT de acuerdo al nivel de CMM. [COCOMO II.0]

- La segunda está basada en las dieciocho Áreas de Procesos Claves (KPAs) del modelo del SEI. El procedimiento para determinar el **PMAT** es establecer el porcentaje de cumplimiento de cada una de las Áreas evaluando el grado de cumplimiento de las metas correspondientes. Para este procedimiento se emplea la Tabla 20.

Áreas de Procesos Claves	Casi Siempre (90%)	A menudo (60-90%)	La mitad de las veces (40-60%)	Ocasionalmente (10-40%)	Casi nunca (<10%)	No se aplica	No se conoce
Administración de Requerimientos							
Planificación del Proyecto de Software							
Seguimiento y supervisión del Proyecto de Software							
Administración de Subcontratos							
Aseguramiento de la Calidad							
Administración de la Configuración							
Objetivo del Proceso de Organización							
Definición del Proceso de Organización							
Programa de Entrenamiento							
Administración Integrada de Software							
Ingeniería del Producto							
Coordinación entre Grupos							
Revisión por Pares							
Administración Cuantitativa							
Administración de la Calidad							
Prevención de Defectos							
Administración de las Tecnologías de Cambio							
Administración de los Procesos de Cambio							

Tabla 20: Nivel de cumplimiento de los objetivos de cada KPA. [COCOMO II.0]

Después de determinar el nivel de cumplimiento de cada KPA el factor **PMAT** es calculado según la fórmula:

$$PMAT = 5 - \left[ \sum_{i=1}^{18} \left( \frac{KPA\%_i}{100} \right) \times \frac{5}{18} \right]$$

#### 4.6 Factores Multiplicadores de Esfuerzo ( Effort Multipliers EM )

El esfuerzo nominal de desarrollo de un proyecto de software se ajusta para una mejor estimación mediante factores que se clasifican en cuatro áreas: Producto, Plataforma, Personal y Proyecto. La Tabla 21 muestra los niveles correspondientes a cada factor según las características inherentes al área. Los valores asignados a cada factor según el nivel se pueden apreciar en la Figura 9, Figura 10, Figura 11 y Figura 12 (COCOMO II.1999.0).

	Factor	Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Producto	RELY	Inconvenientes insignificantes, que afectan solamente a los desarrolladores	Mínimas pérdidas al usuario, fácilmente recuperables	Pérdidas moderadas al usuario recuperables sin grandes inconvenientes	Pérdida financiera elevada o inconveniente humano masivo	Vida humana en riesgo	
	DATA		DB bytes/Pgm SLOC <10	10<=D/P<100	100<=D/P<1000	D/P >0 1000	
	CPLX	Ver Tabla 22					
	RUSE		Ningún componente reusable	Reusable dentro del mismo proyecto	Reusable dentro de un mismo programa	Reusable dentro de una misma línea de productos	Reusable dentro de múltiples líneas de producto
	DOCU	Muchas necesidades del ciclo de vida sin cubrir	Algunas necesidades del ciclo de vida sin cubrir	Necesidades del ciclo de vida cubiertas en su justa medida	Necesidades del ciclo de vida cubiertas ampliamente	Necesidades del ciclo de vida cubiertas excesivamente	
Plataforma	TIME			Uso de <= 50% del tiempo de ejecución disponible	70%	85%	95%
	STOR			Uso de <= 50% del porcentaje total de almacenamiento	70%	85%	95%
	PVOL		Un cambio principal cada 12 meses. Un cambio menor todos los meses	Cambio principal cada 6 meses. Cambio menor cada 2 semanas	Cambio principal cada 2 meses. Cambio menor uno por semana	Cambio principal cada 2 semanas. Cambio menor cada 2 días	
Personal	ACAP	15 percentil	35 percentil	55 percentil	75 percentil	90 percentil	
	PCAP	15 percentil	35 percentil	55 percentil	75 percentil	90 percentil	
	PCON	48 % por año	24 % por año	12 % por año	6% por año	3 % por año	
	AEXP	<= 2 meses	<= 6 meses	1 año	3 años	6 años	
	PEXP	<= 2 meses	<= 6 meses	1 año	3 años	6 años	
	LTEX	<= 2 meses	<= 6 meses	1 año	3 años	6 años	
Proyecto	TOOL	Herramientas que permiten editar, codificar, depurar	Herramientas simples con escasa integración al proceso de desarrollo	Herramientas básicas, integradas moderadamente	Herramientas robustas y maduras, integradas moderadamente	Herramientas altamente integradas a los procesos, métodos y reuso	
	SITE Ubicación Espacial	Internacional	Multi-ciudad y multi-compañía	Multi-ciudad o multi-compañía	Misma ciudad o área metropolitana	Mismo Edificio o complejo	Completamente Centralizado
	SITE Comunicación	Algún teléfono, mail	Teléfonos individuales, FAX	Email de banda angosta	Comunicaciones electrónicas de banda ancha	Comunicaciones electrónicas de banda ancha, ocasionalmente videoconferencia	Multimedia Interactiva
	SCED	75% del nominal	85% del nominal	100% del nominal	130% del nominal	160% del nominal	

Tabla 21: Factores de costo Modelo Post-Arquitectura. [Boehm 1995/1] [Boehm 1995/2]

#### 4.6.1 Factores del producto

Se refieren a las restricciones y requerimientos sobre el producto a desarrollar.

##### **RELY:** Confiabilidad requerida

Este factor mide la confiabilidad del producto de software a ser desarrollado, esto es, que el producto cumpla satisfactoriamente con la función que debe realizar y respete el tiempo de ejecución que se fijó para el mismo.

Los niveles de escala para este factor son Muy Bajo, Bajo, Nominal, Alto y Muy Alto. Si el efecto de la falla del software produce inconvenientes solamente al desarrollador, quien debe solucionarla, el valor de RELY es Bajo. Si por el contrario, la falla atenta contra la vida humana el valor que adopta es Muy Alto.

##### **DATA:** Tamaño de la base de datos

El esfuerzo requerido para desarrollar un producto de software está relacionado con el tamaño de la base de datos asociada. Un ejemplo que marca la importancia de esta influencia es el esfuerzo que insume la preparación de los lotes de prueba que se usan en el testeo del producto.

El valor de DATA se determina calculando la relación entre el tamaño de la base de datos y el tamaño del programa.

$$\frac{D}{P} = \frac{\text{Tamaño\_BasedeDatos(Bytes)}}{\text{Tamaño\_Programa(SLOC)}}$$

##### **CPLX:** Complejidad del producto

CPLX analiza la complejidad de las operaciones empleadas en el producto, clasificadas en operaciones: de control, computacionales, dependientes de los dispositivos, de administración de datos y de administración de interfaz de usuario. El nivel que adopta este factor es el promedio del nivel de cada una de las cinco áreas o tipo de operaciones involucradas, ver Tabla 22 .

	<b>Operaciones de Control</b>	<b>Operaciones computacionales</b>	<b>Operaciones dependientes de los dispositivos</b>	<b>Operaciones de administración de datos</b>	<b>Operaciones de administración de interfaces de usuario</b>
<b>Muy Bajo</b>	Pocas estructuras sin anidamiento: DO, CASE, IF_THEN_ELSE. Composición modular simple por medio de llamadas a procedimientos o simples script	Evaluación de una expresión simple  Por ejemplo: $A=B+C*(D-E)$	Sentencias de lectura / escritura con formatos simples	Arreglos simples en memoria principal.  Consultas, actualizaciones a COTS-DB	Generadores de reportes, Formularios de entrada simples.

<b>Bajo</b>	Estructuras anidadas sencillas	Evaluación de expresiones de complejidad moderada  Por ejemplo: $D = \sqrt{B^2 - 4 \cdot A \cdot C}$	Ninguna necesidad de dispositivos especiales para procesamiento de I/O	Archivo que subsiste sin cambios de estructuras de datos, ni ediciones ni archivos intermedios. Consultas y actualizaciones a COTS-DB moderadamente complejas	Uso de generadores de interfases de usuario gráficas simples
<b>Nominal</b>	Uso mayoritario de anidamientos sencillos .  Algunos controles entre módulos. Tablas de decisión. Pasaje de mensajes o llamadas a subrutinas. Soporte para procesamiento distribuido	Uso de rutinas estándares de matemática y estadística  Operaciones básicas con matrices y vectores	Procesamiento de Entradas/Salidas que incluye selección de dispositivo, procesamiento de errores y chequeo de estado	Varios archivos de entrada y solo un archivo de salida. Cambios estructurales sencillos y ediciones simples. Consultas y actualizaciones a COTS-DB complejas	Uso simple de algunos dispositivos
<b>Alto</b>	Programación estructurada con alto grado de anidamiento con predicados compuestos. Control de cola y pila. Procesamiento distribuido. Control en tiempo real con un procesador	Análisis numérico básico: Interpolación, ecuaciones diferenciales ordinarias, redondeos, truncamientos	Operaciones de Entrada/salida a nivel físico (traducciones a direcciones de almacenamiento físico, seeks, read, etc. )  Optimización de superposición Entradas/Salidas	Triggers simples activados por flujos de datos.  Reestructuración compleja de datos	Uso de un conjunto de dispositivo de Multimedia, Entrada/Salida de Procesamiento de voz
<b>Muy Alto</b>	Codificación recursiva. Manejo de interrupciones con prioridad fija. Sincronización de tareas, complejas llamadas a subrutinas Procesamiento distribuido heterogéneo. Control en tiempo real con un procesador	Análisis numérico estructurado: Matrices de ecuaciones. Ecuaciones diferenciales parciales	Rutinas para el control de interrupciones, enmascaramiento. Manejo de líneas de comunicación	Coordinación de base de datos distribuidas  Disparadores complejos.  Optimización de búsqueda	Multimedia, Gráficos dinámicos, Gráficos 2D y 3D de moderada complejidad
<b>Extra Alto</b>	Planificación múltiple de recursos con cambio dinámico de prioridades. Control al nivel de microcódigo. Control en tiempo real distribuido	Análisis numérico no estructurado: Datos estocásticos. Análisis de ruido con alto grado de precisión	Codificación de dispositivos dependientes del tiempo  Operaciones microprogramadas. Performance crítica con relación a I/O	Alto grado de acoplamiento, relaciones dinámicas, estructuras de objeto, administración de datos en lenguaje natural	Multimedia compleja Realidad virtual

Tabla 22. Factor Multiplicador CPLX. Complejidad del Producto. [COCOMO II.0]

**RUSE:** Requerimientos de reusabilidad

Este factor considera el esfuerzo adicional necesario para construir componentes que puedan ser reusadas dentro de un mismo proyecto o en futuros desarrollos. El incremento del esfuerzo se debe a que se incorporan tareas inherentes al reuso, tales como: creación de diseños genéricos de software, elaboración de mayor cantidad de documentación, testeo intensivo para asegurar que las componentes estén debidamente depuradas, etc.

**DOCU:** Documentación acorde a las diferentes etapas del ciclo de vida

Varios modelos de costo de software tienen un factor de costo para representar el nivel de documentación requerida. En COCOMO II este factor se evalúa en función de la adecuación de la documentación del proyecto a las necesidades particulares en cada etapa del ciclo de vida.

Los posibles valores de DOCU van desde Muy Bajo (documentación que no cubre varias necesidades) hasta Muy Alto (documentación excesiva de acuerdo a las necesidades).

	VLO	LO	NOM	HI	VHI	XHI
RELY	0.82	0.92	1.00	1.10	1.26	XXXX
DATA	XXXX	0.90	1.00	1.14	1.28	XXXX
DOCU	0.81	0.91	1.00	1.11	1.23	XXXX
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	XXXX	0.95	1.00	1.07	1.15	1.24

Figura 9: Factores del producto. Modelo Post-Arquitectura.[COCOMO II.0]

**4.6.2 Factores de la plataforma**

Estos factores analizan la complejidad de la plataforma subyacente.

La plataforma es la infraestructura base de hardware y software, lo que también recibe el nombre de máquina virtual. Si el software a desarrollar es un sistema operativo la plataforma es el hardware, si en cambio se trata del desarrollo de un administrador de base de datos se considerará como plataforma el hardware y el sistema operativo. Por ejemplo, la plataforma puede incluir cualquier compilador o ensamblador empleado en el desarrollo del software.

**PVOL:** Volatilidad de la plataforma

Este factor se usa para representar la frecuencia de los cambios en la plataforma subyacente.

**STOR:** Restricción del almacenamiento principal

Este factor es una función que representa el grado de restricción del almacenamiento principal impuesto sobre un sistema de software. Cuando se habla de almacenamiento principal se hace una referencia al almacenamiento de acceso directo, tales como circuitos integrados, memoria de núcleos magnéticos, excluyendo discos, cintas, etc.

EL valor de STOR está expresado en términos de porcentaje del almacenamiento principal que usará el sistema. El rango posible de valores va desde Nominal hasta Extra Alto.

**TIME:** Restricción del tiempo de ejecución

Este factor representa el grado de restricción de tiempo de ejecución impuesta sobre el sistema de software.

EL valor de TIME está expresado en términos de porcentaje de tiempo de ejecución disponible que usará el sistema. El rango posible de valores va desde Nominal hasta Extra Alto.

	VLO	LO	NOM	HI	VHI	XHI
TIME	XXXX	XXXX	1.00	1.11	1.29	1.63
STOR	XXXX	XXXX	1.00	1.05	1.17	1.46
PVOL	XXXX	0.87	1.00	1.15	1.30	XXXX

Figura 10: Factores de la plataforma. Modelo Post-Arquitectura.[COCOMO II.0]

#### 4.6.3 Factores del personal

Estos factores están referidos al nivel de habilidad que posee el equipo de desarrollo.

**ACAP:** Capacidad del analista

Se entiende por analista a la persona que trabaja con los requerimientos, en el diseño global y en el diseño detallado. Los principales atributos que deberían considerarse en un analista son la habilidad para el diseño, el análisis, la correcta comunicación y cooperación entre sus pares. En este análisis no se tiene en cuenta el nivel de experiencia.

**PCAP:** Capacidad del programador

Las tendencias actuales siguen enfatizando la importancia de la capacidad de los analistas. Sin embargo, debido a que la productividad se ve afectada notablemente por la habilidad del programador en el uso de las herramientas actuales, existe una tendencia a darle mayor importancia a la capacidad del programador. También se evalúa la capacidad de los programadores para el trabajo en equipo más que para el trabajo individual, resaltando las aptitudes para comunicarse y cooperar mutuamente.

**PCON:** Continuidad del personal

Este factor mide el grado de permanencia anual del personal afectado a un proyecto de software. Los posibles valores que puede adoptar PCON van desde 48% (muy bajo) al 3% (muy alto).

**AEXP:** Experiencia en la aplicación

Este factor mide el nivel de experiencia del equipo de desarrollo en aplicaciones similares. El rango de valores posibles de **AEXP** va desde Muy Bajo, representando una experiencia menor a 2 meses, hasta Muy Alto, experiencia de 6 o más años.

**PEXP:** Experiencia en la plataforma

COCOMO afirma que existe gran influencia de este factor en la productividad. Reconociendo así la importancia del conocimiento de nuevas y potentes plataformas, interfaces gráficas, base de datos, redes, etc.

El rango de valores posibles de **PEXP** va desde Muy Bajo, representando una experiencia menor a 2 meses, hasta Muy Alto, experiencia de 6 o más años.

**LTEX:** Experiencia en el lenguaje y las herramientas

Este factor mide el nivel de experiencia del equipo en el uso del lenguaje y herramientas a emplear. El desarrollo de software, hoy en día, incluye el uso de herramientas que soportan tareas tales como representación de análisis y diseño, administración de la configuración, extracción de documentación, administración de librerías, y chequeos de consistencia. Es por ello que, no sólo es importante la experiencia en el manejo del lenguaje de programación sino también en el uso de estas herramientas, ya que influye notablemente en el tiempo de desarrollo.

El rango de valores de posibles de **LTEX** va desde Bajo, representando una experiencia menor a 2 meses hasta Muy Alto representando una experiencia de 6 o más años.

	VLO	LO	NOM	HI	VHI	XHI
ACAP	1.42	1.19	1.00	0.85	0.71	XXXX
AEXP	1.22	1.10	1.00	0.88	0.81	XXXX
PCAP	1.34	1.15	1.00	0.88	0.76	XXXX
PEXP	1.19	1.09	1.00	0.91	0.85	XXXX
LTEX	1.20	1.09	1.00	0.91	0.84	XXXX
PCON	1.29	1.12	1.00	0.90	0.81	XXXX

Figura 11: Factores del personal. Modelo Post-Arquitectura.[COCOMO II.0]

#### 4.6.4 Factores del proyecto

Estos factores se refieren a las condiciones y restricciones bajo las cuales se lleva a cabo el proyecto.

**TOOL:** Uso de herramientas de software

Las herramientas de software se han incrementado significativamente desde la década del 70. El tipo de herramientas abarca desde las que permiten editar y codificar hasta las que posibilitan una administración integral del desarrollo en todas sus etapas.

El rango de valores posibles de **TOOL** va desde Muy Bajo, que corresponde al uso de herramientas sólo para codificación, edición y depuración, hasta Muy Alto, que incluye potentes herramientas integradas al proceso de desarrollo.

**SITE:** Desarrollo multisitio

La determinación de este factor de costo involucra la evaluación y promedio de dos factores, ubicación espacial (disposición del equipo de trabajo) y comunicación (soporte de comunicación).

**SCED:** Cronograma requerido para el desarrollo

Este factor mide la restricción en los plazos de tiempo impuesta al equipo de trabajo. Los valores se definen como un porcentaje de extensión o aceleración de plazos con respecto al valor nominal. Acelerar los plazos produce más esfuerzo en las últimas etapas del desarrollo, en las que

se acumulan más temas a determinar por la escasez de tiempo para resolverlos tempranamente. Por el contrario una relajación de los plazos produce mayor esfuerzo en las etapas tempranas donde se destina más tiempo para las tareas de planificación, especificación, validación cuidadosa y profunda. El rango de valores posibles de *SCED* va desde 75% al 160%.

	VLO	LO	NOH	HI	VHI	XHI
TOOL	1.17	1.09	1.00	0.90	0.78	XXXX
SCED	1.43	1.14	1.00	1.00	1.00	XXXX
SITE	1.22	1.09	1.00	0.93	0.86	0.80

Figura 12: Factores del proyecto. Modelo Post-Arquitectura.[COCOMO II.0]

#### 4.7 Consideraciones destacables del modelo

En el modelo COCOMO se pueden distinguir los siguientes aspectos relevantes:

1. Los factores de costo del modelo son, en orden de importancia:
  - Tamaño: Cantidad de líneas de código fuente.
  - Factor Exponencial de Escala: Representa el impacto de la economía y deseconomía de escala.
  - Factores Multiplicadores de Esfuerzo: Simbolizan características que influyen en el desarrollo del producto, clasificadas en 4 categorías: plataforma, personal, proyecto y producto.
2. COCOMO asume que la especificación de requerimientos no sufrirá cambios fundamentales después de que culmine la fase de planificación de requerimientos. Algunos refinamientos y reinterpretaciones pueden ser inevitables, por lo tanto, cualquier modificación importante implicará una revisión en la estimación de los costos.
3. El período de desarrollo cubierto por este modelo comienza después de la fase de revisión de requerimientos y finaliza con la aprobación de la fase de testeo.
4. El análisis de distribución de esfuerzo y tiempo de desarrollo por fase y actividad se hereda del modelo COCOMO' 81, donde se asume el uso de un modelo de desarrollo secuencial denominado comunmente "Cascada" (Waterfall). Si el proyecto bajo estudio se ejecuta usando otro modelo, los porcentajes de distribución deben reinterpretarse o directamente no deben ser tenidos en cuenta.
5. La estimación de COCOMO abarca todas las tareas en relación directa a las actividades del proyecto, quedando de esta manera excluidas las actividades ejecutadas por operadores, secretarias, administradores de alto nivel, etc.
6. COCOMO evita estimar costos en una unidad monetaria determinada puesto que la unidad mes-persona es más estable, al ser inmune a las fluctuaciones monetarias del mercado. Para convertir mes-persona a dólares se aplica un promedio del valor mes-persona diferente para cada fase del proyecto, lo que permite tener en cuenta los distintos niveles salariales.

## 5 Un Ejemplo Práctico

Se propone ilustrar la aplicación del modelo COCOMO II tomando como ejemplo el mismo proyecto de la Sección 3.4.1, el sistema STUJOB, Sistema de Administración de Trabajo para Estudiantes.

En el ejemplo presentado a continuación se usará un formulario similar a los introducidos en la Sección 3.4.1 denominado CLEF<sup>12</sup>, este formulario es de gran ayuda para la estimación manual. A efectos de clarificar el proceso, también se presentarán algunas pantallas de una herramienta automatizada que implementa el modelo, COCOMO II.1999.0. Vale la pena aclarar que este software está calibrado para módulos de más de 2000 SLOC, por lo tanto los resultados pueden ser no muy precisos.

Los pasos del proceso de estimación de esfuerzo y tiempo de desarrollo son:

1. Identificar los módulos que conforman el sistema, asignarles un número y un nombre e ingresarlos en las columnas **1** y **2**, respectivamente. Ej: **Módulo 2: Search**.
2. Determinar el tamaño de cada módulo expresado en SLOC, líneas de código fuentes liberadas, y registrarlos en la columna **3**.

Se debe tener en cuenta que el tamaño puede verse afectado por dos aspectos: el reuso y la traducción automática, como se analizó en las secciones 4.4.6 y 4.4.7.

Ej: Para el **Módulo Utilities**, adaptado a partir de un módulo de **5000 SLOC**, se considera un valor igual a **1731 SLOC**.

3. Determinar el tamaño en SLOC del Sistema, sumando el tamaño de los módulos que lo componen. Anotarlo en la celda **28**.

Ej: **Tamaño del Sistema:** 1800+700+1200+1700+900+1731 = **8031 SLOC**

4. Calcular el Factor Exponencial de Escala (B), considerando los 5 factores  $W_j$  (PREC, FLEX, RESL, TEAM y MAT) en un nivel nominal. Ver Figura 8.

$$B = 1.01 + 0.01 \times \sum_{j=1}^5 W_j$$

$$B = 1.01 + 0.01 \times (3.72 + 3.04 + 4.24 + 3.29 + 4.68) = 1.1997 \cong 1.20$$

5. Calcular el Esfuerzo Nominal requerido para desarrollar el sistema,  $PM_{\text{Nominal}}$ , en la celda **29** y la Productividad del Proyecto en la celda **30**.

$$PM_{\text{Nominal}} = (KSLOC/PM)_{\text{Nominal}} = 2.94 \times (8.031)^{1.20} = 35.81$$

$$\text{Productividad}_{\text{Nominal}} = (KSLOC/PM_{\text{Nominal}}) = \frac{8031}{35.81} = 224.27$$

---

<sup>12</sup> Component Level Estimating Form. Formulario de Estimación al nivel de componente.

6. Calcular y registrar en la columna **22** el Esfuerzo Nominal por Módulo ( $PM_{\text{Nominal, Módulo}}$ ), que se obtiene como el cociente entre el tamaño del módulo (columna **3**) y la Productividad del Proyecto (celda **30**).

Ej: Para el módulo **Modify**  $PM_{\text{Nominal, Módulo}} = 900 / 224.27 = 4.0130 \cong 4.0$

7. Analizar las características de cada módulo y determinar, con la ayuda de la Tabla 21, en que nivelse encuentra cada uno de los factores de costo. Según el nivel determinado (Muy Bajo, Bajo, Nominal, Alto, Muy Alto) asignar los valores de los multiplicadores de esfuerzo correspondientes, obteniéndolos de la Figura 9 a la Figura 12 y completar las columnas **4** a **20**.
8. Multiplicar los multiplicadores de esfuerzo de la columna **4** a la **20** para cada fila y así obtener el Factor de Ajuste del Esfuerzo **EAF** para cada módulo. Ingresar los resultados en la columna **21**.

Ej: Para el **Módulo Modify**, el cálculo es:

$$EAF_M = 0.87 \times 0.87 \times 0.85 \times 1.15 \times 0.81 \times 1.09 \times 1.09 \times 0.90 = 0.64$$

9. Calcular el Esfuerzo Estimado por Módulo,  $PM_{\text{Estimado, Módulo}}$ , en la columna **23**, multiplicando el valor de  $PM_{\text{Nominal, Módulo}}$ , columna **22**, por el correspondiente Factor de Ajuste  $EAF_M$  de la columna **21**.

Ej: Para el **Módulo Modify**, el cálculo es:

$$PM_{\text{Estimado, Módulo}} = PM_{\text{Nominal, Módulo}} \times EAF_M = 4.0 \times 0.64 = 2.56 \cong 2.6$$

10. Sumar los valores calculados en el ítem anterior para determinar el Esfuerzo Estimado del Sistema Total  $PM_{\text{Estimado}}$ , registrar este valor en la celda **31**.

Ej:  $PM_{\text{Estimado}} = 4.3 + 1.2 + 3.4 + 4.1 + 2.6 + 6.4 = 22$

11. Determinar el Tiempo de Desarrollo Estimado del proyecto **TDEV** y anotarlo en la celda **34**.

$$TDEV = \left[ 3.0 \times 22^{(0.33 + 0.2 \times (1.2 - 1.01))} \right] \times \frac{SCED\%}{100} = 9.36$$

12. Anotar en la columna **24** el Costo del Mes-Persona para cada módulo, expresado en miles de dólares. Posteriormente multiplicar estos costos por los  $PM_{\text{Estimado, Módulo}}$  correspondientes (columna **23**), encontrando así el Costo Estimado de cada módulo y registrarlo en la columna **25**.

Ej: Para el **Módulo Utils**, se asume un costo más bajo debido a la participación de un grupo de analistas y programadores novatos:

$$\text{Costo Mes-Persona} = 5250$$

$$\text{Costo}_{\text{Estimado, Módulo}} = \text{Costo Mes-Persona} \times PM_{\text{Estimado, Módulo}} = 5250 \times 6.4 = 33600$$

13. Calcular el Costo Total del Sistema sumando los valores obtenidos en el ítem anterior y registrarlo en la celda **32**.

Ej:  $\text{Costo}_{\text{Estimado}} = 23091 + 6444 + 18258 + 22071 + 13962 = 117490$

14. Para cada módulo determinar y registrar en la columna **26** el Costo por instrucción en US\$, el cual se calcula como el cociente entre el Costo de Desarrollo (columna **25**) y el Tamaño del Módulo (columna **3**).

Ej: Para el **Módulo Modify**

$$\text{Costo por instrucción en miles de US\$} = 13962 / 900 = 15.51 \text{ dólares}$$

15. Para cada módulo determinar y registrar en la columna **27** la Productividad, calculada como el cociente entre el Tamaño del Módulo (columna **3**) y el Esfuerzo Estimado por módulo  $PM_{Nominal, Módulo}$  (columna **23**).

Ej: Para el **Módulo Modify**

$$\text{Productividad} = 900/2.6 = 346.15 \text{ SLOC / mes-persona}$$

El software COCOMO II.1999.0 además de las estimaciones presentadas brinda otras posibilidades que vale la pena mencionar:

- Tanto el esfuerzo como el tiempo de desarrollo del proyecto completo se pueden distribuir por fase. Según se observa en la Figura 13, los porcentajes de distribución son similares a los usados en el modelo COCOMO' 81 correspondientes al Modo Semiacoplado y a un tamaño de 8 KSLOC, ver Tabla 2. La diferencia surge debido a que los porcentajes se han interpolado para considerar el tamaño real de software de 8031 SLOC.

Overall Phase Distribution					
PROJECT	STUBJOB				
SLOC	8031				
TOTAL EFFORT	22.065 Person Months				
	PCNT	EFFORT (PM)	PCNT	SCHEDULE	Staff
Plans And Requirements	7.000	1.545	18.003	1.767	0.874
Product Design	17.000	3.751	25.001	2.454	1.529
Programming	60.996	13.459	51.995	5.103	2.637
- Detailed Design	25.999	5.736	----	----	----
- Code and Unit Test	34.997	7.722	----	----	----
Integration and Test	22.004	4.855	23.004	2.258	2.150

Figura 13: Distribución del esfuerzo y tiempo de desarrollo del sistema total por fase

- Existe la posibilidad de analizar la distribución de esfuerzo y de tiempo de desarrollo de cada módulo en las distintas fases de desarrollo. La Figura 14 muestra los valores correspondientes al módulo Qedit.

Overall Phase Distribution					
MODULE	Qedit				
SLOC	1800				
TOTAL EFFORT	4.325 Person Months				
	PCNT	EFFORT (PM)	PCNT	SCHEDULE	Staff
Plans And Requirements	7.000	0.303	18.003	1.767	0.171
Product Design	17.000	0.735	25.001	2.454	0.300
Programming	60.996	2.638	51.995	5.103	0.517
- Detailed Design	25.999	1.124	----	----	----
- Code and Unit Test	34.997	1.514	----	----	----
Integration and Test	22.004	0.952	23.004	2.258	0.421

Figura 14: Distribución del esfuerzo y tiempo de desarrollo de un módulo (Qedit) por fase

- Al igual que COCOMO'81 el software también permite estudiar como se distribuye el esfuerzo y tiempo de desarrollo para cada actividad en cada fase. La Figura 15 muestra los valores considerando la fase de Integración y Testeo. La Figura 17 considera la misma fase sólo para el módulo Qedit.

Phase Distribution - Project Integration & Test				
-----				
Life Cycle Phase	Integration and Test			
Life Cycle Effort	4.855 Person Months			
Life Cycle Schedule	2.258 Months			
-----				
	PCNT	EFFORT (PM)	SCHEDULE	Staff
Requirements Analysis	2.500	0.121	2.258	0.054
Product Design	5.000	0.243	2.258	0.108
Programming	35.003	1.699	2.258	0.753
Test Planning	2.501	0.121	2.258	0.054
Verification and Validation	30.998	1.505	2.258	0.667
Project Office	7.999	0.388	2.258	0.172
CM/QA	8.000	0.388	2.258	0.172
Manuals	7.999	0.388	2.258	0.172

Figura 15: Distribución del esfuerzo y tiempo de desarrollo de la fase Integración y Testeo por subfases<sup>13</sup>

Phase Distribution - Module Integration & Test - Qedit				
-----				
Life Cycle Phase	Integration and Test			
Life Cycle Effort	0.952 Person Months			
Life Cycle Schedule	2.258 Months			
-----				
	PCNT	EFFORT (PM)	SCHEDULE	Staff
Requirements Analysis	2.500	0.024	2.258	0.011
Product Design	5.000	0.048	2.258	0.021
Programming	35.003	0.333	2.258	0.148
Test Planning	2.501	0.024	2.258	0.011
Verification and Validation	30.998	0.295	2.258	0.131
Project Office	7.999	0.076	2.258	0.034
CM/QA	8.000	0.076	2.258	0.034
Manuals	7.999	0.076	2.258	0.034

Figura 16: Distribución del esfuerzo y tiempo de desarrollo de la fase Integración y Testeo por subfases, considerando el módulo Qedit, solamente

<sup>13</sup> Se llaman subfases, a lo que COCOMO' 81 denomina actividades.

Número Módulo	Nombre Módulo	SLOC	Producto					Plataforma			Personal						Proyecto			EAF	PM Nominal Mes-Pers	PM Estimado Mes-Pers	Costo Mes-Pers Dólares	Costo	Costo x Instrucc Dólares	Productividad SLOC/Mes-Pers			
			RELY	DATA	CPLX	RUSE	DOCU	TIME	STOR	PVOL	ACAP	PCAP	PCON	AEXP	PEXP	LTEX	TOOL	SITE	SCED										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27			
1	Qedit	1800	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87	0.85	1.00	1.00	0.81	1.00	1.00	0.90	1.00	1.00	0.54	8	4.3	5370	23091	12.8	418.6			
2	Search	700	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87	0.85	0.88	1.00	0.81	0.91	0.91	0.90	1.00	1.00	0.39	3.1	1.2	5370	6444	9.2	583.3			
3	Output	1200	1.00	1.00	0.87	1.00	1.00	1.00	1.00	0.87	0.85	1.15	1.00	0.81	1.09	1.09	0.90	1.00	1.00	0.64	5.3	3.4	5370	18258	15.2	352.9			
4	UpEdit	1700	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87	0.85	1.00	1.00	0.81	1.00	1.00	0.90	1.00	1.00	0.54	7.6	4.1	5370	22071	12.9	414.6			
5	Modify	900	1.00	1.00	0.87	1.00	1.00	1.00	1.00	0.87	0.85	1.15	1.00	0.81	1.09	1.09	0.90	1.00	1.00	0.64	4	2.6	5370	13962	15.51	346.1			
6	Utils	1731*	1.00	1.00	0.73	1.00	1.00	1.00	1.00	0.87	1.00	1.15	1.00	1.00	1.09	1.09	0.90	1.00	1.00	0.78	8.2	6.4	5250	33600	19.4	270.4			
	<b>28</b>	<b>8031</b>	Total SLOC																					<b>31</b>		<b>32</b>		<b>33</b>	
																								Esfuerzo Estimado <i>PM<sub>Est</sub></i>	<b>22.0</b>		<b>117426</b>		<b>365</b>
	<b>29</b>	<b>35.81</b>	Esfuerzo $PM_{Nominal}$																					<b>34</b>	<b>9.36</b>		Costo Total Estimado		Productividad Estimada
																								Tiempo de Desarrollo <i>TDEV</i>					
	<b>30</b>	<b>224.27</b>	Productividad ( $KSLOC/PM$ ) <sub>Nominal</sub>																										

Figura 17: Formulario para la estimación de esfuerzo y tiempo de desarrollo utilizando COCOMO II.

\* Tamaño del módulo, considerando que es adaptado a partir de un módulo de 5000 SLOC. Ver Tema 4.4.6, pag. 37.

## 6 Conclusiones

Durante la última década, la evolución de las tecnologías de desarrollo de software impulsó un nuevo enfoque en la estimación de costos, que considerara conceptos tales como orientación a objetos, reingeniería, reusabilidad, utilización de paquetes comerciales, composición de aplicaciones. Además, surgió la necesidad de que estos nuevos modelos se adaptaran a la granularidad de la información disponible en las diferentes etapas de desarrollo.

La familia de modelos de COCOMO II, constituida por los modelos *Composición de Aplicación*, *Diseño Temprano* y *Post-Arquitectura*, conforma estas premisas ya que son parte de sus objetivos principales.

COCOMO II, al igual que el modelo original preserva su estado de dominio público en relación a los algoritmos, la herramienta de software, estructuras de datos, relaciones e interfases.

Para contar con información fidedigna y favorecer el continuo refinamiento y calibración del modelo, la USC implementó un *Programa de Recolección de Datos*<sup>14</sup>. Se espera que en el transcurso del año 2001 se libere una nueva versión de la herramienta calibrada y actualizada.

Otra de las ventajas de este modelo es que puede ser adaptado (calibrado) a un organismo en particular, si se cuenta con la experiencia de un número importante de proyectos ya culminados que puedan aportar los datos necesarios para la recalibración

Sin lugar a dudas, en la actualidad siguen existiendo inconvenientes y limitaciones para las estimaciones, pero más allá de esto COCOMO II ha recorrido un importante camino, logrando la madurez necesaria del modelo para conseguir estimaciones de gran precisión.

---

<sup>14</sup> Es importante destacar que cualquier entidad u organización que trabaje en proyectos de desarrollo de software puede participar en este esfuerzo de recolección de datos mediante la contestación de un cuestionario provisto por el USC o enviando los archivos generados por el software USC COCOMO II.1999.0.

## **7 Anexo I.**

Formularios para la Estimación Jerárquica de Software. Modelo COCOMO Detallado.

Proyecto: ..... Analista: ..... Fecha: .....

1 Nro SS	2 Subsistema SS	8 KSLOC	Producto		Atributos de la Computadora				Personal		Proyecto			32	20	33	34	35	36
			21 RELY	22 DATA	23 TIME	24 STOR	25 VIRT	26 TURN	27 ACAP	28 AEXP	29 MODP	30 TOOL	31 SCED	32 EAF SS	20 PM MOD	33 PM EST			
			PD DD CUT IT																

Modo: **9**

	Total KSLOC
<b>10</b>	Esfuerzo Nominal $PM_{Nominal}$
<b>11</b>	Productividad $(KSLOC/PM)_{Nominal}$

**12**

Fracción por Fase	
PD DD CUT IT	

PD  
DD  
CUT  
IT

Total


PM Schedule K



## 8 Acrónimos y Abreviaturas

3GL	Third Generation Language Lenguaje de 3era. Generación
AA	Percentage of reuse effort due to assessment and assimilation Porcentaje de esfuerzo de reuso debido a la evaluación y asimilación
ACAP	Analyst Capability Aptitud del Analista
ACT	Annual Change Traffic Tráfico Anual de Cambios
ASLOC	Adapted Source Lines of Code Líneas de Código Fuente Adaptadas
AEXP	Applications Experience Experiencia en las Aplicaciones
AT	Automated Translation Traducción automática
BRAK	Breakage Desperdicio de Código
CASE	Computer Aided Software Engineering Ingeniería de Software Asistida por Computadora
CLNT	Cantidad de tablas de datos en clientes usadas en SCREEN o REPORT
CM	Percentage of code modified during reuse Porcentaje de Código modificado durante el reuso
CMM	Capability Maturity Model Modelo de Madurez de Capacidades
COCOMO	Constructive Cost Model Modelo Constructivo de Costo
COTS	Commercial Off The Shelf Packages Paquetes, módulos, clases, librerías comerciales
CPLX	Product Complexity Complejidad del Producto
DATA	Database Size Tamaño de la Base de Datos
DBMS	Database Management System Sistema de Administración de Base de Datos

DI	Degree of Influence Grado de Influencia
DM	Percentage of design modified during reuse Porcentaje de Diseño modificado durante el reuso
DOCU	Documentation to match lifecycle needs Documentación acorde a las necesidades del ciclo de vida
EDS	Electronic Data Systems Sistemas Electrónicos de Datos
ESLOC	Equivalent Source Lines of Code Líneas de Código Fuente Equivalentes
FCIL	Facilities Facilidades
FP	Function Points Puntos Función
GFS	Government Furnished Software Software Provisto por el Gobierno
GUI	Graphical User Interfase Interfaz de Usuario Gráfica
ICASE	Integrated Computer Aided Software Environment Ambiente Integrado Asistido de Software
IM	Percentage of integration redone during reuse Porcentaje de Integración durante el reuso
KSLOC	Thousands of Source Lines of Code Miles de Líneas de Código Fuente
LEXP	Programming Language Experience Experiencia en el Lenguaje de Programación
LTEX	Language and Tool Experience Experiencia en lenguajes y Herramientas
MODP	Modern Programming Practices Prácticas Modernas de Programación
NIST	National Institute of Standards and Technology Instituto Nacional de Estándares y Tecnología
NOP	New Object Points Nuevos Puntos Objetos
OS	Operating Systems

	Sistemas Operativos
PCAP	Programmer Capability Aptitud del Programador
PCON	Personnel Continuity Continuidad del Personal
PDIF	Platform Difficulty Dificultad de la Plataforma
PERS	Personnel Capability Aptitud del Personal
PEXP	Platform Experience Experiencia en la Plataforma
PL	Product Line Línea de Productos
PM	Person Month Mes-Persona
PREX	Personnel Experience Experiencia del Personal
PROD	Productivity rate Tasa de Productividad
PVOL	Platform Volatility Volatilidad de la Plataforma
RCPX	Product Reliability and Complexity Confiabilidad y Complejidad del producto
RELY	Required Software Reliability Confiabilidad Requerida
RUSE	Required Reusability Reusabilidad Requerida
RVOL	Requirements Volatility Volatilidad de los Requerimientos
SCED	Required Development Cronograma Cronograma de Desarrollo Requerido
SECU	Classified Security Application Aplicación de Seguridad Clasificada
SEI	Software Engineering Institute Instituto de Ingeniería de Software

SITE	Multi-site operation Operación Multi-Sitio
SLOC	Source Lines of Code Líneas de Código Fuente
STOR	Main Storage Constraint Restricción de Almacenamiento Principal
SRVR	Cantidad tablas de datos en servidores (mainframe o equivalente) usadas en SCREEN o REPORT
TURN	Computer Turnaround Time Tiempo de Respuesta de la computadora expresado en horas
T&E	Test and Evaluation Test y Evaluación
SU	Percentage of reuse effort due to software understanding Porcentaje de esfuerzo de reuso debido al entendimiento del software
TIME	Execution Time Constraint Restricción del Tiempo de Ejecución
TOOL	Use of Software Tools Uso de Herramientas de Software
USAF/ESD U.S.	Air Force Electronic Systems Division División de Sistemas Electrónicos de la Fuerza Area
VEXP	Virtual Machine Experience Experiencia en la Máquina Virtual
VIRT	Virtual Machine Volatility Volatilidad de la Máquina Virtual
VMVH	Virtual Machine Volatility: Host Volatilidad de la Máquina Virtual: Principal
VMVT	Virtual Machine Volatility: Target Volatilidad de la Máquina Virtual
%reuse	El porcentaje de pantallas, reportes, y módulos de 3GL reusados de aplicaciones previas, clasificadas en grados de reuso

## 9 Referencias

[Albretch 1979] Albretch, A.J , “Measuring Application Development Productivity”, Proc. IBM Application Development Symposium, Monterrey, CA, Octubre 1979, Págs. 83-92.

- [Amadeus 1994] Amadeus, *Amadeus Measurement System User's Guide, Version 2.3a*, Amadeus Software Research, Inc., Irvine, California, July 1994.
- [Banker 1994] Banker, R., R. Kauffman and R. Kumar (1994), "An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment" *Journal of Management Information Systems*.
- [Behrens 1983] Behrens, C. (1983), "Measuring the Productivity of Computer Systems Development Activities with Function Points", *IEEE Transactions on Software Engineering*.
- [Boehm 1981] Barry W Boehm, *Software Engineering Economics*, Ed. Prentice Hall.
- [Boehm 1989] Boehm, B. and W. Royce, *Ada COCOMO and the Ada Process Model*, Proceedings, Fifth COCOMO Users' Group Meeting, Software Engineering Institute, Pittsburgh, PA, November 1989.
- [Boehm 1995/1] Boehm B.W., Clark B., Horowitz E., Westland C., Madachy R., Selby R., *Cost Models for Future Software Life Cycle Processes: COCOMO II*, Annals of Software Engineering Special Volume on Software Process and Product Measurement, 1995, Vol 1, pp. 45-60. Primer paper que trata COCOMO II.  
<http://sunset.usc.edu/COCOMOII/cocomo.html>
- [Boehm 1995/2] Boehm B.W., Clark B., Horowitz E., Westland C., Madachy R., Selby R., *The COCOMO 2.0 Software Cost Estimation Model*,  
<http://sunset.usc.edu/COCOMOII/cocomo.html>.
- [Chidamber and Kemerer 1994] Chidamber, S. and C. Kemerer, *A Metrics Suite for Object Oriented Design*, *IEEE Transactions on Software Engineering*, 1994.
- [COCOMO II.0] Documentos de la ayuda del software COCOMO II.0.
- [Fenton 1997] Fenton, N.E. Pfleeger, *Software Metrics. A Rigorous & Practical Approach*, PWS Publishing Company, Boston, 1997, S.L. Capitulo 7.
- [Ghezzi 1991] Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, "Fundamentals of Software Engineering".
- [Gerlich and Denskat 1994] Gerlich, R. and Denskat U., "A Cost Estimation Model for Maintenance and High Reuse", Proceedings, ESCOM 1994, Ivrea, Italy.
- [Goethert et al. 1992] Goethert, W., E. Bailey, M. Busby, *Software Effort and Cronograma Measurement: A Framework for Counting Staff Hours and Reporting Cronograma Information*, CMU/SEI-92-TR-21, Software Engineering Institute, Pittsburgh, PA.
- [Madachy 1997] Madachy, J. Raymond, *Heuristic Risk Assessment Using Cost Factors*, *IEEE Software*, May/June 1997, pp. 51-59.
- [Parikh and Zvegintzov 1983]
- [Park 1992] Park R., *Software Size Measurement: A Framework for Counting Source Statements*, CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA.

- [IFPUG 1994] IEPUG, *IFPUG Function Point Counting Practices: Manual Release 4.0*, International Function Point Users' Group, Westerville, OH.
- [Kunkler 1985] Kunkler, J., *A Cooperative Industry Study on Software Development/Maintenance Productivity*, Xerox Corporation, Xerox Square - -- XRX2 52A, Rochester, NY 14644, Third Report, March 1985.
- [Pressman 1997] Pressman Roger, "*Ingeniería de Software, Un Enfoque Práctico*".
- [Ruhl and Gunn 1991] Ruhl, M., and Gunn, M., "Software Reengineering: A Case Study and Lessons Learned" NIST Special Publication 500-193, Washington, DC, September 1991.
- [Selby 1988] Selby R., "Empirical Analyzing Software Reuse in a Production Environment", In *Software Reuse: Emerging Technology*, W. Tracz (Ed.), IEEE Computer Society Press, 1988, pp. 176-189.
- [Selby et al. 1991] Selby R., A. Porter, D. Schimidt and J. Berney , *Metric-Driven Analysis and Feedback systems for Enabling Empirically Guided Software Development*, Proceedings of the Thirteenth International Conference on Software Engineering (ICSE 13), Austin, TX, May 13-16, 1991, pp. 288-298.
- [USC 1989] Center for Software Engineering, *Modeling Software Defect Introduction*. Los Angeles, California 1989.  
<http://sunset.usc.edu/COCOMOII/cocomo.html>.