

ESTIMACIÓN DE COSTOS

• INTRODUCCION

Todo proyecto de ingeniería de software debe partir con un buen plan, pero lamentablemente, la planificación es una tarea nada trivial. Uno de los aspectos que dificultan la labor de administradores y jefes de proyecto en torno a la planificación es la difícil tarea de realizar una estimación de costos y plazos realista.

La estimación es más arte que ciencia; también es parte de la etapa de la planificación y algunas actividades de la ingeniería. La diferencia en la estimación de costos entre ingeniería de software y otras disciplinas es que en ingeniería de software lo principal para las personas es el costo; y en otras disciplinas el costo de las cosas materiales depende de la actividad.

Existen técnicas para la estimación de costos, pero para ello se requiere experiencia, acceso a una buena información histórica y coraje para confiar en medidas cuantitativas cuando todo lo que existe son datos cualitativos.

El manejador de costo principal para un proyecto de desarrollo de software es sin duda el tamaño del producto. La medida del tamaño debe ser tal que esté en relación directa con el esfuerzo de desarrollo, por lo que las métricas de tamaño tratan de considerar todos los aspectos que influyen en el costo, como tecnología, tipos de recursos y complejidad.

• MÉTRICAS PARA LA PRODUCTIVIDAD Y CALIDAD DEL SOFTWARE

La medición es esencial para cualquier disciplina de ingeniería y la ingeniería de software no es una excepción.

Las métricas de software se refieren aun amplio rango de medidas para el software de computadoras dentro del contexto de la planificación del proyecto de software, las métricas de calidad pueden ser aplicadas a organizaciones, procesos y productos los cuales directamente afectan a la estimación de costos.

Las mediciones en el mundo físico pueden ser catalogadas en dos campos: medidas directas (por ej. La longitud de un tornillo), y medidas indirectas (por ej. Calidad de tornillos producidos, medida por la cuenta de los tornillos rechazados). Las métricas de software pueden ser catalogadas de forma parecida.

Se puede clasificar en:

Métricas de productividad, se centran en el rendimiento del proceso de la ingeniería de software.

Métricas de Calidad, proporcionan una indicación de cómo se ajusta el software, a los requerimientos implícitos y explícitos del cliente.

Métricas Técnicas, se centran en el carácter del software mas que en el proceso, a través del cual el software a sido desarrollado.

Métricas Orientadas al tamaño, son utilizadas para obtener medidas directas del resultado y la calidad de la ingeniería del software.

Métricas Orientadas a la Función, son medidas indirectas del software y del proceso por el cual se desarrollará; se centran en la funcionalidad o utilidad del programa (Puntos de Función)

Métricas Orientadas a la persona, consiguen información sobre la forma en que la gente desarrolla software de computadora y sobre el punto de vista humano de la efectividad de las herramientas y métodos.

• ESTIMACION DEL PROYECTO DE SOFTWARE

Para realizar estimaciones seguras de coste y esfuerzo surge un numero de posible de opciones como:

Retrasar la estimación mas adelante en el proyecto (obviamente se puede hacer una estimación cien por ciento fiable después de completar el proyecto)

Utilizar "técnicas de descomposición " relativamente simples para generar las estimaciones del proyecto de software (por ej. Estimación LDC y PF)

Desarrollar un modelo empírico para el coste y el esfuerzo de software.

Adquirir una o más herramientas automáticas de estimación.

Desdichadamente la primera opción, aunque atractiva no es practica, porque las estimaciones del coste deben ser proporcionadas de *antemano*. Las tres opciones restantes son aproximaciones viables para la estimación del proyecto de software. Las técnicas de descomposición utilizan una aproximación de "divide y vencerás " para la estimación del proyecto de software. Los modelos de estimación empíricos pueden utilizarse para completar las técnicas de descomposición y ofrecer una aproximación de la estimación potencialmente evaluable por ella misma. Las herramientas automáticas de estimación implementan una o mas técnicas de descomposición o modelos empíricos.

• MODELOS DE ESTIMACIÓN EMPÍRICA

Un modelo de estimación para el software por computadora utiliza formulas derivadas empíricamente para predecir los datos.

Los datos empíricos que soportan la mayoría de los modelos se obtienen de una muestra limitada de proyectos. Tomando en cuenta los recursos se tienen los modelos recursos y consisten en una o más ecuaciones obtenidas empíricamente que predicen el esfuerzo (personas-mes), la duración del proyecto (meses cronológicos) o otros datos pertinentes al proyecto. Según Basili describe cuatro modelos de recurso:

modelos simple-variable estáticos (ej. COCOMO), modelos multi-variables estáticos, modelos multi-dinámicos variables y modelos teóricos.

3.1.1 MODELO COCOMO

Barry Boehm en su libro "economía de la ingeniería de software" detalla un modelo amplio de estimación de costos llamado COCOMO (Constructive Cost Model). La palabra "constructive" se refiere a el hecho que el modelo ayuda a un estimador a comprender mejor la complejidad del software; este modelo es un ejemplo de variable simple estático y es usado por miles de administradores de proyecto de software .

COCOMO ayuda a estimar el esfuerzo, tiempo, gente y costos (ya sea estos de desarrollo, equipamiento y mantenimiento).

El modelo provee tres "niveles" de aplicación: básico, intermedio y avanzado, basados en los factores considerados por el modelo.

Básico, es un modelo estático simplemente evaluado que calcula el esfuerzo (y costo) del desarrollo del software como función del programa expresado en líneas de código (LDC estimados).

Intermedio, calcula el esfuerzo del desarrollo del software como función del tamaño del programa y un conjunto de "guías de costo" que incluye una evaluación subjetiva del producto, hardware, personal y de los atributos del proyecto.

Avanzado, incorpora todas las características de la versión intermedia con una evaluación del impacto de las vías de costo en cada fase (análisis, diseño, etc) del proceso de la ingeniería de software.

El modelo básico se extiende para considerar un conjunto de atributos de guías de costo que pueden agruparse en cuatro categorías principales:

Producto (por ej. Requerimientos de software, confiabilidad, tamaño de la base de datos, y complejidad del producto).

Computadora (por ej. Restricciones en el tiempo de ejecución y almacenamiento).

Personal (por ej. Capacidad de análisis, experiencia en aplicaciones tanto en lenguajes de programación y capacidad del programador)

Proyecto (por ej. Uso de practicas modernas de programación, uso de herramientas de software y requerimiento de un plan de desarrollo).

En cada nivel de aplicación están definidos para tres tipos de proyectos de software:

Modo orgánico, proyectos de software relativamente pequeños y sencillos en los que pequeños equipos con buena experiencia en la aplicación trabajan en un conjunto de requerimiento poco rígidos.

Modo semi-acoplado(semi-detached), un proyecto de software intermedio en tamaño y complejidad en el cual equipos con distintos niveles de experiencia debe satisfacer requerimientos poco y medio rígidos

Modo acoplado(detached), un proyecto de software que debe ser desarrollado dentro un conjunto estricto de hardware, software y de restricciones operativas.

Modos que están basados en la complejidad de la aplicación y el desarrollo del ambiente. El modelo de esfuerzo general aplicable a todos los niveles de aplicación y modos está dado por :

Donde:

E = es el esfuerzo estimado expresado en hombres-mes

EDSI es el número estimado de líneas de código distribuidas en miles para el proyecto

a, h son constantes determinadas por el modo del desarrollo, ambos incrementados por la complejidad de la aplicación.

EAF es el factor de ajuste de esfuerzo, es igual a 1 para la modelo básica e igual al producto de 15 factores de costo para la modelo intermedia y avanzada. Cada factor de costo multiplicativo es reflexivo de un incremento proporcional (> 1) o decremento (<1) en costo.

A continuación veremos los coeficientes para el modelo intermedio que depende de modo de desarrollo:

MODO DE DESARROLLO	A	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Modo básico utiliza el tamaño y el modo intermedio 15 manejadores de costo que son los siguientes:

Manejadores de Costo	Very Low	Low	Nominal	High	Very High	Extra High
ACAP Analyst Capability	1.46	1.19	1.00	0.86	0.71	-
AEXP Applications Experience	1.29	1.13	1.00	0.91	0.82	-
CPLX Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
DATA Database Size	-	0.94	1.00	1.08	1.16	-
LEXP Language Experience	1.14	1.07	1.00	0.95	-	-
MODP Modern Programming Practices	1.24	1.10	1.00	0.91	0.82	-
PCAP Programmer Capability	1.42	1.17	1.00	0.86	0.70	-
RELY Required Software	0.75	0.88	1.00	1.15	1.40	-

Reliability						
SCED Required Development Schedule	1.23	1.08	1.00	1.04	1.10	-
STOR Main Storage Constraint	-	-	1.00	1.06	1.21	1.56
TIME Execution Time Constraint	-	-	1.00	1.11	1.30	1.66
TOOL Use of Software Tools	1.24	1.10	1.00	0.91	0.83	-
TURN Computer Turnaround Time	-	0.87	1.00	1.07	1.15	-
VEXP Virtual Machine Experience	1.21	1.10	1.00	0.90	-	-
VIRT Virtual Machine Volatility	-	0.87	1.00	1.15	1.30	-

El tiempo de desarrollo es igual a :

Donde:

E, es el esfuerzo

c,d son coeficiente, cuyos valores se indicaron anteriormente en una tabla.

El número de programadores es igual a:

Representando un enfoque monolítico para la estimación de costos, a continuación presentaremos un ejemplo:

Ejemplo

1. Usando COCOMO básico para estimar el esfuerzo requerido en el desarrollo de un programa de 850 líneas en modo orgánico se tiene los siguiente:

$$E = 3.2 (8.5)1.05 * 1 = 30 \text{ Mes-hombre}$$

Boehn también adopta el modelo COCOMO Intermedio para repartir costos a componentes individuales, considerando las 8500 líneas proyectadas, realizando la lista de componentes:

COMPONENTES	EDSI	% TOTAL	CMMNOM
PERSONAL	2000	23.4%	7.06
FACTURA	3000	35.3%	10.60
POR COBRAR	3500	41.2%	12.36

Nivel de Componente de COCOMO Intermedio

Basado sobre 30hombre-mes para el esfuerzo (E), el numero de EDSI para hombre-mes es dado por.

$$(EDSI/mes-hombre)NOM = 8500/30 = 283 \text{ EDSI mes-hombre}$$

Usando el EDSI/ mes-hombre, cada componente aporta una proporción al total de valor por ejemplo el componente *nominal* mes-hombre(CMMNOM) para el componente de personal es dado por:

$$(CMMNOM) = EDSI \text{ por componente} / (EDSI/MM)NOM = 200/283 = 7.06 \text{ CMMNOM}$$

Después de calcular el CMMNOM para cada componente, el factor de ajuste de esfuerzo (EAF) es calculado individualmente para cada componente. El factor EAF es aplicado a CMMNOM llegando a un nuevo ajuste en mes-hombre, estimando (CMMADJ) para cada componente. Esto es como un modelo monolítico, el cual es aplicado a un simple EAF para el sistema. Por lo tanto, es posible aclarar las variaciones entre los factores de costo y las diversidades de componentes. Por ejemplo: el CMMADJ para el componente factura es calculado por :

$$CMMADJ = (CMMNOM) * (EAF) = 10.60 * 1.13 = 11.98 \text{ CMMADJ}$$

2. usando COCOMO Intermedio

Modo es orgánico

Tamaño = 200 KDSI

Manejadores de costo >

- Baja Confiabilidad => 0.88
- Alta Complejidad del producto => 1.15
- Baja experiencia en la aplicación => 1.13
- Alta experiencia en los lenguajes de programación => 0.95
- Otros manejadores de costo asumen a ser nominales => 1.00

$$EAF = .88 * 1.15 * 1.13 * .95 = 1.086$$

$$E = 3.2 * (2001.05) * 1.086 = 906 \text{ mes-hombre}$$

$$TDEV = 2.5 * 9060.38 = 33.24$$

$$PG = 906/33.24 = 27 \text{ programadores}$$

3. Utilizando la herramienta Modelo de Implementación Intermedio COCOMO81, para la estimación de costos de un Sistema Colaborativo. (**ver anexos**)

Ventajas

- COCOMO es transparente, se puede ver como trabaja con otros modelos tal como SLIM (Software Life Cycle Management).
- Manejadores de costo ayudan particularmente a el estimador a comprender el impacto de diferentes factores que afectan en el costo del proyecto.

Desventajas

- Triunfo depende ampliamente de la adaptación de el modelo a las necesidades de la organización, usando datos históricos; los cuales no siempre están disponibles.
- Extremadamente vulnerable para la mis-clasificación de el modo de desarrollo.
- Es difícil estimar KDSI con precisión sobre el antiguo proyecto, cuando la mayoría de las estimaciones de esfuerzo son requeridas.
- KDSI, realmente, no es una medida del tamaño, sino una medida de longitud.

Como mejora de COCOMO surgieron varias versiones de COCOMO y podemos mencionar una de ellas que es: COCOMO II, Ada COCOMO y COCOMO Incremental.

COCOMO II

El nuevo modelo incorporado en el año 1990, tiene características de los modelos COCOMO 81 y Ada COCOMO. COCOMO II tiene también tres submodelos ; El modelo de composición de la aplicación es usada para estimar el esfuerzo y planificación de proyectos que usa las herramientas integradas CASE (Computer Aided Software Engineering) para un desarrollo rápido de la aplicación.

Realizando una comparación entre COCOMO 81 y COCOMO II; a este último se le añadió nuevos manejadores de costos para la aplicaciones precedentes, flexibilidad en el desarrollo, necesita documentación para el ciclo de vida, múltiples sitios de desarrollo y requiere software reusable.

Modelo COCOMO II post-arquitectura cubre el actual desarrollo y mantenimiento de un producto de software. Esta etapa del ciclo de vida procede mas a un costo efectivo, si el ciclo de vida de una arquitectura de software ha sido desarrollada, validada con respecto a la misión del sistema y establecida como un marco de trabajo para el producto.

El modelo de post-arquitectura predice el esfuerzo de desarrollo del software, personas-mes (PM), utiliza un conjunto de 17 multiplicadores de manejadores de costo (EM) y un conjunto de 5 escalas de manejadores de costo para determinar la escala del exponente del proyecto (SF). Esta escalas de los manejadores de costo remplazan los modos de aplicación (orgánico, sem.-acoplado y acoplado); el modelo tiene la siguiente forma:

$$PM = A * (Size)^{1.01} * \prod_{i=1}^{17} EM_i^{E_i} * SF^E$$

Los manejadores de costo tiene para elegir una de las seis posibilidades que son: Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH), y Extra High (XH); no todos los rangos son válidos para todos los manejadores de costo.

ADA COCOMO

Barrí Bohema & Walter Rocíe, 1987, 1988 definen el nuevo modelo COCOMO, llamado "Ada COCOMO".

Este modelo al igual que el COCOMO estándar utiliza los manejadores de costo y ecuaciones anteriormente definidas.

COCOMO INCREMENTAL

Fue definido casi al mismo tiempo que Ada COCOMO. EL modelo COCOMO Incremental es una moderna alternativa para el tradicional modelo cascada de el desarrollo de procesos de software.

El modelo de desarrollo Incremental COCOMO permite una variedad de desarrollo de procesos. En vez de modelar el software como a esfuerzo simple para obtener un producto simple; el modelo incremental COCOMO permite desarrollar una serie de proyectos de software concurrente y producir un producto intermedio.

Esta estrategia reduce risk y permite entregar un producto inicial más fácilmente al cliente.

También existen algunas derivaciones de COCOMO como ser:

- Cocots, (Constructive Cost)
- Cossemo, (Constructive Staged Schedule & Effort Model).
- Copromo, (Constructive Productivity Improvement Model).
- Coqualmo
- Coradmo
- **CONCLUSIONES**

Al realizar el trabajo de investigación acerca de COCOMO llegamos a las siguientes conclusiones:

- COCOMO es una herramienta basada en la líneas de código la cual le hace muy poderoso para la estimación de costos y no como otros que solamente miden el esfuerzo en base al tamaño.
- Hoy en día es necesario para un administrador de proyectos posser una herramienta de estimación de costos; y esta herramienta puede ser COCOMO.

- COCOMO representa el más extenso modelo empírico para la estimación de software publicado hasta la fecha.
- Existen herramientas automáticas que estiman costos basados en COCOMO como ser: Costar, COCOMO 81.
- Con la realización de este trabajo ampliamos nuestro conocimientos acerca de la estimación de costos que es fundamental para un analista, administrador de proyecto.
- **BIBLIOGRAFIA**
- Ian Somerville, "Software Engineering", cuarta Edición.
- R. S. Pressman, "Ingeniería de Software"

URL'S

http://www.reifer.com/cocomo_eff.htm

www.softstarsystems.com

http://sunset.usc.edu/COCOMOII/cocomo81_pgm/cocomo81.html

<http://sunset.usc.edu/COCOMOII/cocomo.html>

<http://yunus.hun.edu.tr/~sencer/cocomo.html>

<http://www.stsc.hill.af.mil/crosstalk/1995/jun/metrics.asp>

<http://zaurak.cis.ksu.edu/~radhika/hw2.html>

PAPER'S

- [Randy K. Smith, Allen Parrish, Joanne Hale] Cost Estimation for Component Based Software Development, 1988
- [Bradford Clark, Sunita Devnani-Chulani, Barry Boehm] Calibrating the COCOMO II Post-Architecture Model

EVALUACIÓN DE MODELOS DE ESTIMACIÓN

DE COSTOS DE SOFTWARE

(SLIM,COCOMO,PUNTOS DE FUNCIÓN)

A continuación se presenta un relato de una evaluación de modelos que estiman costos para el desarrollo software. Para esta evaluación se precisó los datos (información) de 15 empresas (proyectos), información que ayudaron a evaluar cuatro de los más populares

modelos algorítmicos usados para estimar costos. Estos son: SLIM, COCOMO, PUNTOS DE FUNCIÓN y ESTIMACS.

• INTRODUCCIÓN.

Los investigadores han expresado su interés en la estimación de costos para el desarrollo de software, esta preocupación se ha hecho presente a medida que avanza el desarrollo de proyectos de software. Principalmente, el interés de comparar resultados usando diferentes métricas para cálculos de productividad.

Para ello se presentan tres preguntas a responder:

- ¿Son los modelos de estimación de costo del software realmente generalizables para ambientes diferentes, para los cuales fueron desarrollados?. Si no, ¿pueden ser fácilmente moldeables para el ambiente típico de procesamiento de datos para los negocios?
- ¿Los modelos que no usan líneas de código(SLOC) son tan exactos como las que sí? Si es así, ¿Se podría eliminar la necesidad de estimar las líneas de código del proyecto?

Para un mejor entendimiento, primeramente se presenta una explicación de cómo se seleccionó los cuatro modelos para la evaluación, seguidamente se procederá a con una breve descripción de los modelos, para continuar con la descripción del ambiente en el cual los datos se originaron. Luego se describirá los métodos de recolección-datos y análisis-datos, para entrar directamente a explicar los resultados y finalmente dar conclusiones.

• ACERCA DE LOS MODELOS.

2.1 Selección de los Modelos.

Un repaso de la literatura revela una interesante diferencia entre modelos de estimación, esta se refiere a modelos que usan SLOC(líneas de código) como principal entrada y aquellos que no las utilizan. SLOC fue seleccionado rápidamente, como métrica por los investigadores sin ninguna duda, por su cuantificabilidad y objetividad.

El siguiente paso para escoger modelos fue el de buscar en la literatura actual, modelos que se centren en SLOC, de esta búsqueda se selecciono dos: modelo COCOMO y Algoritmo SLIM. Igualmente se procedió a la búsqueda de modelos no-SLOC, solo se encontraron dos: Método de Puntos de Función, desarrollado por Alan Albrecht y el modelo ESTIMACS, desarrollado por Howard Rubin.

2.2 Descripción de los MODELOS

Continuando, se presenta una breve descripción de los cuatro modelos mencionados anteriormente.

- **Algoritmo SLIM.**

Software Life Cycle Management, SLIM fue desarrollado en el año 1970 por Larry Putman. Depende de una estimación SLOC para el tamaño general del proyecto, se modifica a través del uso de la curva de Raleigh para producir la estimación del esfuerzo.

El usuario puede influenciar la forma de la curva a través de dos parámetros, la pendiente inicial de la curva y el factor de productividad. Puesto que estos son números dimensionales el usuario tiene dos maneras de ingresar los datos, en *la primera* el usuario puede calibrar el modelo de ingreso de datos o bien responde a las 22 preguntas que el SLIM puede proporcionar recomendando un PF y MBL, el segundo método fue el escogido para esta investigación debido a la ausencia de un banco de datos que pueda calibrarse, lo que reflejaría con precisión la experiencia de los usuarios.

- **Modelo COCOMO.**

Constructive Cost Model, fué desarrollado por Barry Boehm de TRW, publicado en 1981. Basado en el análisis de 63 proyectos de software desarrollados. Boehm desarrollo un modelo (de tres niveles: básico, intermedio y avanzado) fácil de entender que predice la duración del proyecto, así como el esfuerzo para la realización de este.

El modelo detallado COCOMO(utilizado en esta evaluación) es muy similar al modelo intermedio excepto que el proyecto es dividido en cuatro fases: diseño del producto, diseño detallado, codificación de pruebas, integración-pruebas.

- **Modelo PUNTOS de FUNCIÓN.**

El modelo Puntos de función fue realizado por Alan Albrecht y fue publicado en 1979. Albrecht estaba interesado en el problema general de medición de productividad en sistemas y creo un método como alternativa a la estimación SLOC. Este método captura los el numero de transacciones de entrada y el numero de reportes.

El modelo Puntos de Función tiene ventajas sobre la cuantificación de las líneas de código (SLOC) como: es posible estimarlas en el ciclo de vida, alrededor del tiempo de definición de requerimientos para el documento y pueden ser estimadas por un miembro del proyecto relativamente no técnico.

Existen dos pasos que involucrados para el conteo de puntos de función:

- el primero contar las funciones de usuario (tipos de entradas externas e internas, archivos lógicos internos, de interface externos y tipos externos de encuesta)
- el segundo, ajustar la complejidad de los procesos.

- **Modelo ESTIMACS.**

El modelo Estimacs fue desarrollado por Howard Rubin. Este modelo no requiere como entrada el numero de SLOC, tiene 25 preguntas, cuyas respuestas le sirven de datos de entrada.

3. ACERCA DE LOS DATOS.

Base de Datos del Proyecto, los proyectos seleccionados para la evaluación poseían dos atributos en común: eran de un tamaño mediano y el estaban disponibles para completar la recolección de datos.

- **METODOLOGÍA.**

- **Formas para recolectar los datos.** Se presentaron tres formas para recolectar los datos: Se diseñó una Forma para capturar la información a fondo a cerca del proyecto (por ej. Fabricante del hardware, modelo), después del análisis de requerimientos de cada modelo, se consolidaron dos Formas adicionales para recolectar los datos. Aunque cada uno de los modelos tiene una cantidad de requerimientos, se tomó en cuenta aquellos que tenían en común, y que fueran relacionados a la productividad.

- **Procedimiento para Recolectar los datos.** Para cada proyecto hubo una junta con el manejador de proyectos que tenía que llenar las formas específicas. Los propósitos eran:

Discutir cada pregunta para asegurarse que se entendiera el concepto y se respondiera consistentemente

Resaltar la importancia de la participación del manejador del proyecto en el trabajo.

- **Procedimiento para Analizar los datos,** Una vez llenadas las Formas se revisó la consistencia del contenido. Se presentó redundancia de muchas preguntas en los modelos, a excepción de las preguntas del modelo Punto de Función con relación a las del modelo ESTIMACS. Lo que significó que era posible asegurar una respuesta acerca del conocimiento del personal del proyecto para el modelo COCOMO, ya que presentó consistencia con preguntas similares del modelo SLIM.

- **Análisis del error,** Para tener un margen de error entre una estimación, se calculó el error relativo relacionando el esfuerzo estimado con el esfuerzo verdadero realizado en el proyecto. Usando dos exámenes para evaluar el proyecto:

- El de cálculo de error relativo(MRE).
- El método de regresión.

- **RESULTADOS.**

Los resultados que se presentan, son los encontrados después de que cada modelo fue corrido en los 15 proyectos seleccionados.

- **Resultados de SLIM,** fue corrido usando los parámetros por defecto del modelo, los usuarios respondieron a las 22 preguntas del SLIM.

Este modelo no tiene una vía buena para el cálculo del error relativo(MRE), obteniendo un porcentaje de error promedio de 772%, con el pequeño error de 21%.

- **Resultados de COCOMO,** los resultados para las tres versiones COCOMO en el cálculo del MRE fueron pobres, obteniéndose un promedio de 601% con el error más bajo de 83%.

- **Resultados de Puntos de Función**, el promedio MRE es de 102,74%.
- **Resultados ESTIMACS**, presento un promedio del 85%.
- **CONCLUSIONES.**

Después de examinar cada modelo independientemente, se procedió a responder las tres preguntas formuladas al comenzar con esta evaluación:

¿Son los modelos de estimación de costo del software realmente generalizables para ambientes diferentes, para los cuales fueron desarrollados?. Si no, ¿pueden ser fácilmente moldeables para el ambiente típico de procesamiento de datos para los negocios?

Los modelos desarrollados en ambientes diferentes no trabajan muy bien, cuando se desea acoplarlos. Los porcentajes de error calculados usando la formula de error relativo están en el rango de 85 a 775%. Esta variación se debe la diferencia de ambientes.

¿Los modelos que no usan líneas de código(SLOC) son tan exactos como las que si? Si es así, ¿Se podría eliminar las necesidad de estimar las líneas de código del proyecto?

En términos de los resultados de error relativo los modelos que no usan líneas de código como entrada fueron mejores. Pero en términos de los resultados de regresión están altamente correlacionados.

En conclusión: los modelos, a pesar de su perfeccionamiento sobre diferentes entradas para la estimación de esfuerzo, no modelan de manera adecuada los factores que afectan la productividad. Es necesario hacer mas investigaciones acerca de cómo medir todos los factores que afectan los sistemas de productividad profesional, si la profesión es encontrarse con los cambios del futuro.

Resolución del ejemplo 3

Sistema Colaborativo, para calcular el esfuerzo requerido se tiene:

$$E = 3.6 * (EDSI) ^ 1.20$$

Donde 3.6 y 1.20 son constantes para lo que son programas de sistema y EDSI son las líneas de código

De donde en el caso del sistema colaborativo el esfuerzo será:

$$E = 3.6 * (11.038) ^ 1.20$$

$$E = 64$$

Donde (11.038) significa 11038 líneas de código fuente, y esto se conoce por la notación KDSI

Nota. - Los estimadores de esfuerzo excluyen los costos de planeación, instalación y entrenamiento, así como los costos de secretarías, personal de limpieza y operadores del equipo de cómputo.

El tiempo de desarrollo para un programa, según lo sugiere Boehm, es de:

$$TDEV = 2.5 * (E) ^ 0.32$$

En el caso del sistema colaborativo el tiempo necesario se calculará de la siguiente manera:

$$TDEV = 2.5 * (64) ^ 0.32$$

$$TDEV = 9.5$$

Dado el número total de meses programador de un proyecto y el tiempo nominal de desarrollo requeridos, el nivel promedio de contratación puede obtenerse mediante una simple división:

$$PG = E / TDEV$$

Donde PG son el número de programadores requeridos.

De donde para el sistema colaborativo el número de programadores requeridos será:

$$PG = 64 / 9.5$$

$$PG = 6.73$$

$$PG = 7$$

Ahora para la distribución del tiempo total de desarrollo del sistema colaborativo, para cada fase se tiene que:

40 % Análisis y Diseño

20% Codificación

40% Pruebas

De donde se tendrá que el tiempo para cada fase será:

4 Meses Análisis y Diseño

5 Meses Pruebas

3 Meses Codificación

Ahora se verá lo que son los costos, estos serán:

- Costos de Desarrollo,
- Costos de Equipamiento,
- Costos de Mantenimiento.
- **Costo de Desarrollo**
- Personal: Analistas y Programadores 20900\$

Analistas (400\$ / mes, durante 9.5 meses): 2 analistas

Programadores (200\$/mes, durante 9.5 meses): 7 programadores

- Pruebas (500 \$ / mes, durante 5 meses) 2500 \$
- Alquiler Equipo (100 \$ / mes, durante 9.5 meses) 6650 \$
- Material: 200\$
- Suministros de Computadora
- Material de Escritorio

TOTAL COSTO DESARROLLO: 30250 \$

- **Costo Equipamiento**
- 25 Maquinas Terminales 23750 \$

Características:

- Tarjeta madre QDI Brillant I BX AGP ATX
- Procesador Intel Pentium II 350 Mhz (Original)
- Cache en tarjeta de 512 Kb
- 32 Mb. DIMM de memoria RAM PC-100
- Disco duro de 6.4 Gb. (Western Digital)
- Monitor de 14" SVGA (Samsung SyncMaster 400b)
- Tarjeta de Video de alta velocidad de 4 Mb.
- Tarjeta de Red FAST 100Mb. PCI PnP

Precio Unitario de 950 \$

- 1 Maquina Servidora 1300 \$

Características:

- Tarjeta madre marca SOYO - 5BT
- Procesador Intel Pentium II 400 Mhz (Original)
- Cache en tarjeta de 512 Kb
- 64 Mb. DIMM de memoria RAM PC-100
- Disco duro de 8.4 Gb. (Western Digital)
- Monitor de 14" SVGA (Samsung SyncMaster 400b)
- Tarjeta de Video de alta velocidad de 4 Mb.
- Tarjeta de Red FAST 100Mb. PCI PnP
- Conexión Red: 2600 \$
- Topología Estrella
- Cable UTP (200 mts., 5\$/metro)
- 2 HUB's (450 \$ precio unitario)
- Accesorios de Red

TOTAL COSTO EQUIPAMIENTO 27650 \$

- **Costo de Mantenimiento** (Anual) 500 \$

TOTAL COSTO MANTENIMIENTO 500 \$

TOTAL COSTO SISTEMA COLABORATIVO 58400 \$

RESULTADOS DEL MODELO DE IMPLEMENTACION INTERMEDIO COCOMO 81

Los resultados obtenidos con el calculador MODELO INTERMEDIO COCOMO 81 son los siguientes:

Esfuerzo = 64 personas / mes

Tiempo = 9.5 meses

Cantidad de líneas de código en ambos casos es:

11038 líneas

Este número de líneas fueron determinadas en el anterior trabajo.

COCOMO

Haciendo comparaciones con el COCOMO realizado manualmente se tiene las siguientes observaciones:

- En cuanto al esfuerzo se tiene una variación aceptable de 2.16 personas/mes
- Tiempo se tiene una discrepancia de 0.98 meses

Nuestra opinión sería que los resultados obtenidos no están muy lejos; y que son aceptables y también si se alteraría algún parámetro del Calculador entonces se llegaría a una aproximación todavía mas aceptable; pero esto sería ya forzando el resultado.

En su libro: Model and Metrics for Software Management and Engineering