Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

Tipo de artículo: Artículo original
Temática: Ingeniería y gestión de software
Recibido: 28/04/2015 | Aceptado: 02/11/2015

Estimación de costos de desarrollo, caso de estudio: Sistema de Gestión de Calidad del Reactor TRIGA Mark III

Estimation development cost, study case: Quality Managment System Reactor TRIGA Mark III

Tereso Antonio Antúnez Barbosa [1*], Rosa María Valdovinos Rosas [1], José Raymundo Marcial Romero [1], Marco Antonio Ramos Corchado [1], Edgar Herrera Arriaga [2]

[1] Universidad Autónoma del Estado de México, Facultad de Ingeniería, Ciudad Universitaria, Cerro de Coatepec s/n, Toluca, México. CP 50110. antonioantunez7@outlook.com, li_rmvr@hotmail.com, jrmarcialr@gmail.com, marco.corchado@gmail.com

[2] Instituto Nacional de Investigación Nuclear ININ, Departamento del Reactor, Carretera México-Toluca s/n, 52750, La Marquesa, Ocoyoacac, México. edgar.herrera@inin.gob.mx

[*] Autor para correspondencia: antonioantunez7@outlook.com

Resumen

El proceso de estimación de costos en Ingeniería del *software* no es una tarea sencilla, más que eso es un proceso que debe tratarse cuidadosamente para obtener una estrategia que permita resolver problemas asociadas al esfuerzo, costo y tiempo de las actividades que se realizan en un proyecto de desarrollo de sistemas de información. En este contexto, lo principal tanto para desarrolladores como para los clientes es el costo, los primeros para tener una remuneración adecuada por su trabajo y los segundos para sentir que están pagando lo justo por lo solicitado. Sin embargo, en otras disciplinas los costos dependen de la actividad o proceso que se realiza, con lo que se puede deducir que el costo principal del producto final de un proyecto de desarrollo de *software* es sin duda su tamaño. En este artículo se realiza un estudio comparativo de los modelos de estimación de costos más comunes y utilizados en la actualidad con la finalidad de crear un análisis estructurado que proporcione la información necesaria acerca de costo, tiempo y esfuerzo para la toma de decisiones en un proyecto de desarrollo de *software*. Posteriormente se muestra la aplicación a un caso de estudio, el cual se denomina Sistema de Monitorización Automática del Sistema de Gestión de Calidad del Reactor TRIGA Mark III.

Palabras clave: Ingeniería del *Software*, métrica, estimación de costos, SLOC

Abstract

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

The process of estimating costs in software engineering is not a simple task, it must be addressed carefully to obtain an efficient strategy to solve problems associated with the effort, cost and time of activities that are performed in the development of an information system project. In this context the main goal for both developers and customers is the cost, since developers are worry about the effort pay-load and customers are worry about the product pay-load. However, in other fields the cost of goods depends on the activity or process that is performed, thereby deduce that the main cost of the final product of a development project software project is undoubtedly its size. In this paper a comparative study of common models for estimating costs are developed. These models are used today in order to create a structured analysis to provide the necessary information about cost, time and effort for making decisions in a software development project. Finally the models are applied to a case study, which is a system called Monitorización Automática del Sistema de Gestión de Calidad del Reactor TRIGA Mark III.

Keywords: *Software Engineering, metrics, estimation of costs, SLOC*

Introducción

No importa que tan grande o tan pequeño sea un proyecto de desarrollo de *software*, una buena estimación de su costo permitirá resolver problemas asociados al esfuerzo y tiempo invertido en su realizacion. En la bibliografía se proponen métricas para calcular el costo del desarrollo de *software*, con la finalidad de tener una mejor planeación en el desarrollo de sistemas. En esta investigación se analizan tres modelos de estimación de costos: Un modelo basado en líneas de código fuente (mejor conocido como SLOC) (Nussbaum, 2015), un modelo no-SLOC (Nussbaum, 2015), y el modelo basado en puntos de casos de uso (Tuya, 2007), con la intención de determinar sus beneficios de acuerdo a la cuatificabilidad y objetividad en el diseño de *software* a la medida.

Los aspectos evaluados en el análisis de un sistema de información determinan que método de estimación es el más adecuado aplicar para obtener el mejor resultado, es decir tomando en cuenta el lenguaje de programación utilizado, documentación UML y el sueldo remunerado por experiencia y lenguaje utilizado.

Para el análisis de este artículo, resultó como mejor práctica que para la determinación del costo de un proyecto se deba estimar en la etapa de levantamiento de requerimientos como es el caso del modelo basado en puntos de casos de uso o también en la etapa previa a liberación del sistema cuando ya se cuenta con el código casi terminado (modelo basado en SLOC) y las interfaces de usuario (modelo basado en puntos de función), según en el estado en que se encuentre el proyecto de *software*.

Modelos de estimación de costos

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

En esta sección se analizan los modelos más utilizados para estimar costos en el desarrollo de sistemas, uno basado en SLOCS, otro no basado en SLOCS y el modelo basado en puntos de casos de uso.

1. Modelos basados en SLOCS

En estos modelos las líneas de código fuente se utilizan como métrica para contar el tamaño de un producto de *software* (Álvarez, 2007). El modelo aquí estudiado es el modelo COCOMO nombrado así por sus siglas *Constructive Cost Model*, cuya traducción al español es Modelo Constructivo de Costos (Farr, 2011). COCOMO permite predecir la duración de un proyecto, así como el esfuerzo necesario para su realización medido en personas-mes. Para ello COCOMO divide los proyectos de *software* en tres tipos dependiendo de su tamaño (Campos, 2012): modo orgánico, semi-acoplado y acoplado. La ecuación (1) permite calcular el esfuerzo y la (2) el tiempo de desarrollo del proyecto:

$$E_i = a * (KLOCS)^b \quad (1)$$

$$T_d = c * (E_i)^d \quad (2)$$

Donde:

E_i es el esfuerzo, medido en meses-hombre

a, b, c y d: son valores que dependen del tipo de proyecto (Tabla 1)

T_d es el tiempo de desarrollo requerido en meses

KLOCS: es el valor en miles de líneas de código

Tabla 1. Valores de las constantes de acuerdo al modelo COCOMO (Garzón, 2003)

Modo de desarrollo	***a***	***b***	***c***	***d***	**Mes-Hombre (nominal)**	**Tiempo de desarrollo (nominal)**
Orgánico	3.2	1.05	2.5	0.38	$E_i = 3.2 * KLOCS^{1.05}$	$T_d = 2.5 * E_i^{0.38}$
Semi-acoplado	3.0	1.12	2.5	0.35	$E_i = 3.0 * KLOCS^{1.12}$	$T_d = 2.5 * E_i^{0.35}$
Acoplado	2.8	1.20	2.5	0.32	$E_i = 3.2 * KLOCS^{1.05}$	$T_d = 2.5 * E_i^{0.32}$

Una de las principales ventajas de este modelo es que se puede aplicar en diferentes fases del ciclo de vida y puede aplicarse a cualquier organización (Brewer, 2013), además utiliza manejadores de costos que ayudan principalmente al estimador a comprender el impacto de otros factores que afectan en el costo del proyecto, tales como presiones de tiempo, tamaño y requisitos de desarrollo (Garzón, 2003). Por otro lado, la principal desventaja del modelo es que utiliza datos históricos, como son archivos de código fuente que ya no se utiliza, que no siempre están disponibles y es extremadamente vulnerable a la clasificación del modelo de desarrollo, ya sea programación estructurada o programación orientada a objetos (Pressman, 2002).

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

2. Modelos no basados en SLOCS

Estos modelos son una alternativa a los modelos basados en SLOCS y utilizan principalmente preguntas o transacciones de entrada (Álvarez, 2007). Uno de los métodos más estudiados es el método de puntos de función nombrado así por sus siglas en inglés *Function Point Analysis* (FPA), está definido como un método estándar para medir el desarrollo de *software* desde el punto de vista del usuario (Durán, 2003). En su funcionamiento, mediante la asignación de "puntos" identifica los componentes del sistema en términos de transacciones y grupos de datos lógicos que son relevantes para el usuario en su negocio. Los puntos de función miden el tamaño de una aplicación plafinificada (lógico) o existente (funcional), también puede ser usado para medir el tamaño de los cambios de una aplicación existente. Si los cambios están en los requerimientos funcionales del usuario o el diseño está compleado (Garmus, 2011). El proceso general es el siguiente (Durán, 2003):

1. Determinar el tipo de conteo. Existen tres tipos de conteo de puntos de función: para proyectos en desarrollo, para proyectos en mantenimiento y para una aplicación desarrollada.
2. Identificar los alcances de la medición y los límites de la aplicación. Identificar el alcance es identificar los sistemas, aplicaciones o subconjuntos de una aplicación que será medida. La frontera de la aplicación es el límite entre la aplicación que está siendo medida y las aplicaciones externas al dominio del usuario.
3. Conteo de las funciones de datos. Identificar y contar la capacidad de almacenamiento de los datos que representan la funcionalidad que satisfacen requerimientos de datos internos y externos. Se distinguen dos tipos de funciones de datos: ILF: Archivo Lógico Interno, es un grupo de datos internos relacionados que el usuario identifica, cuyo propósito principal es almacenar datos mantenidos a través de alguna transacción y EIF: Archivo Lógico Externo, es un grupo de datos externos relacionados y referenciados pero no mantenido por alguna transacción dentro del conteo. El procedimiento a seguir es:
    a) Identificar archivos.
    b) Asignar a cada uno un tipo de ILF o EIF.
    c) Identificar la cantidad de DET (*Data Element Type*) que es un campo único no repetitivo reconocible por el usuario y RET (*Record Element Type*) que es un conjunto de campos en un archivo, reconocible por el usuario.
    d) Asignar a cada uno un valor de complejidad (alta, media, baja) en función de la cantidad de DET y RET como se muestra en la Tabla 4.

Tabla 4. Nivel de complejidad para los ILF y EIF (Salazar, 2009)

	Número de campos			
	Para ILF/EIF	1 a 19 DET	20 a 50 DET	51 o más DET
Número de registros	1 RET	Baja	Baja	Media
	2 a 5 RET	Baja	Media	Alta
	6 o más RET	Media	Alta	Alta

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

4. Contar las funciones transaccionales. Identificar y contar la capacidad de realizar operaciones. Se distinguen tres tipos de funciones transaccionales: Entrada Externa (EI): Su propósito principal es mantener uno o más archivos lógicos internos, Salida Externa (EO): Su propósito principal es presentar información al usuario mediante un proceso lógico diferente al de sólo recuperar datos y Consulta externa (EQ): Su propósito principal es presentar información al usuario leída de uno o más grupos de datos.

   El procedimiento para contar las funciones transaccionales es:

   - Identificar las transacciones.
   - Asignar a cada una un tipo EI, EO o EQ.
   - Identificar la cantidad de DET y FTR (*File Type Referenced*) que es un tipo de archivo al que se hace referencia en una transacción, tiene que ser un ILF o EIF.
   - Asignar a cada una un valor de complejidad (alta, media, baja) en función de la cantidad de DET y FTR como se muestra en la Tabla 5.

Tabla 5. Nivel de complejidad para EI, EO y EQ (Salazar, 2009)

EI/EO/EQ	**1 a 4 DET**	**5 a 15 DET**	**16 o más DET**
0 a 1 FTR	Baja	Baja	Media
2 FRT	Baja	Media	Alta
3 o más FTR	Media	Alta	Alta

5. Determinar los puntos de función sin ajustar (PFSA): Sumar el número de componentes de cada tipo conforme a la complejidad y utilizar la Tabla 6 para obtener el total.

Tabla 6. Cuenta total de puntos de función (Salazar, 2009)

	Baja	**Media**	**Alta**	**Total**
EI	$EI * 3$	$EI * 4$	$EI * 6$	$\sum EI$
EO	$EO * 4$	$EO * 5$	$EO * 7$	$\sum EO$
EQ	$EQ * 3$	$EQ * 4$	$EQ * 6$	$\sum EQ$
ILF	$ILF * 7$	$ILF * 10$	$ILF * 15$	$\sum ILF$
EIF	$EIF * 5$	$EIF * 7$	$EIF * 10$	$\sum EIF$
Total (PFSA)				$\sum PFSA$

6. Determinar el valor del factor de ajuste: A través de una ponderación de 0 a 5 de catorce factores que completan la visión externa de la aplicación, se obtiene el GTI (Grado Total de Influencia), (Tabla 7).

Tabla 7. Factores de complejidad (Salazar, 2009).

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

#	Factor de Complejidad	Valor
1	Comunicación de datos	
2	Proceso distribuido	
3	Objetivos de rendimiento	
4	Configuración de explotación usada por otros sistemas	
5	Tasa de transacciones	
6	Entrada de datos en línea	
7	Eficiencia con el usuario final	
8	Actualizaciones en línea	
9	Lógica del Proceso Interno Compleja	
10	Reusabilidad del código	
11	Contempla la conversión e instalación	
12	Facilidad de operación	
13	Instalaciones múltiples	
14	Facilidad de cambios	
Factor de Complejidad Total (GTI)		$\sum valor\ de\ los\ factores$

7. Determinar los puntos de función ajustados. Una vez evaluadas las 14 características descritas anteriormente se suman para obtener el GTI. Posteriormente el GTI se aplica en la ecuación 3, y se obtiene el FAV (Factor de Ajuste de Valor).

$$FAV = (GTI * 0.01) + 0.65 \quad (3)$$

Del total de puntos de función ajustados se utiliza la siguiente formula.

$$PF = FAV * PFSA \quad (4)$$

El cálculo del esfuerzo en horas-hombre se calcula a través de la ecuación:

$$Esfuerzo = \frac{PF}{\frac{1}{\#\ personas}} \text{ horas-hombre} \quad (5)$$

Para calcular la duración en horas o en meses se debe aplicar la ecuación 6 y 7.

$$Duración\ en\ horas = \frac{esfuerzo}{0.125} \quad (6)$$

$$Duración\ en\ meses = \frac{Duración\ en\ horas}{100\ hrs/mes} \quad (6)$$

$$Costo\_total = sueldode1participante * \#personas * duraciónenmeses + otroscostos \quad (7)$$

Algunas ventajas de este modelo son (Campos, 2012) que estima los puntos de función en el ciclo de vida alrededor del tiempo de definición de requerimientos, análisis y diseño, son independientes del lenguaje, herramientas o tecnologías utilizadas y están basados en la vista de un usuario externo al sistema lo que permite al personal no técnico tener una mejor comprensión de lo que se está midiendo. Por otro lado, entre sus desventajas destaca conteo subjetivo, diferentes personas pueden llegar a estimaciones diferentes

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

para el mismo problema, dificil de automatizar y de calcular, y es orientado a las aplicaciones de procesamiento de datos tradicionales (Agarwal, 2010).

3. Modelo basado en puntos de casos de uso

La estimación mediante el análisis de puntos de casos de uso, consiste en la medición del tiempo de desarrollo de un proyecto a través del proceso de asignación de "pesos" a un cierto número de factores que lo afectan (Thomas, 2011). En específico, el método obtiene como entrada los requisitos del sistema en términos de actores y casos de uso, proporcionando uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico (Cuadrado, 2008).

Un caso de uso documenta una interacción entre el *software* y un actor o más. Dicha interacción tiene que ser, en principio, una función autónoma dentro del sistema (Yuya, 2007) que permite estimar el tamaño (cuantificar) del *software* en términos de horas necesarias para la operación de los casos de uso y el número de personas que se requieren para realizarlo, cuantificando la complejidad del sistema.

Algunas de las ventajas de este modelo son que trabaja bien con diferentes tipos de *software*, muestra buen rendimiento en proyectos pequeños, medianos y grandes. En tanto que los principales inconvenientes son que a pesar de que existe el estándar UML para escribir casos de uso, cada ingeniero de *software* escribe el caso de uso según comprenda los requerimientos del sistema. A continuación se describe el proceso que se sigue.

1. Cálculo de los puntos de casos de uso no ajustados (UUCP, *Unajusted Use Case Points*). Consiste en calcular el peso tanto para actores (UAW, *Unajusted Actor Weights*) como para casos de uso (UUCW, *Unajusted Use Case Weights*) y sumar el resultado de UAW y UUCW. Los criterios para la asignación de pesos de los actores se muestran en la Tabla 8.

Tabla 8. Clasificación y peso de los actores (Cuadrado, 2008)(Thomas, 2011).

Tipo de actor	Descripción	Peso
Simple	Otro sistema externo, interactúa con el sistema a desarrollar mediante una interfaz de programación definida y conocida, (API, *Application Programming Interface*)	1
Medio	Otro sistema externo que interactúa a través de protocolo (conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de la red)	2
Complejo	Un usuario físico que interactúa a través de una interfaz gráfica de usuario.	3

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

Se debe contar el número de actores que hay en el sistema, multiplicar cada tipo por su factor de peso y sumar esos productos para obtener el UAW, como se muestra en la ecuación (8).

$$UAW = \sum_{i=1}^{n-actores}(cantidadtipoactor_i * peso) \quad (8)$$

Por otro lado, los criterios para la asignación de pesos de los actores se muestran en la tabla 9 (Cuadrado, 2008)(Thomas, 2011).

Tabla 9. Clasificación y peso de los casos de uso (Colomo, 2014).

Tipo de caso de uso	Descripción	Peso
Simple	El caso de uso contiene menos de 3 transacciones	5
Medio	El caso de uso contiene de 4 a 7 transacciones	10
Complejo	El caso de uso tiene más de 7 transacciones	15

De la misma forma que con los actores, es necesario contar el número de casos de uso que hay en el sistema, multiplicar cada tipo por su factor de peso y sumar esos productos para obtener el UUCW (ecuación (9)).

$$UUCW = \sum(tipocasodeuso_i * peso) \quad (9)$$

Por último solo resta aplicar la fórmula de puntos de casos de uso no ajustados UUCP (ecuación (10)).

$$UUCP = AUW + UUCW \quad (10)$$

2. Calcular los puntos de casos de uso (UCP, *Use Case Points*). Consiste en realizar el producto de los puntos de casos de uso no ajustados, el peso de los factores técnicos (TCF, *Technical Factors*) mostrados en la Tabla 10 y el peso de los factores ambientales (EF, *Enviroment Factors*) mostrados en la Tabla 11, los cuales se ponderan respecto a las habilidades y experiencias del grupo o equipo de trabajo. Para calcular los TCF, se deben considerar valores entre 0 y 5, donde: Irrelevante de 0 a 2, Medio de 3 a 4, Esencial 5.

Tabla 10. Factores técnicos ((Colomo, 2014)

Factor	Descripción	Peso
T1	Sistema distribuido	2
T2	Objetivos de performance o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Objetivos especiales de seguridad	1
T12	Acceso directo a terceras partes	1
T13	Facilidades especiales de entrenamiento a usuarios	1

*Irrelevante de 0 a 2, Medio de 3 a 4, Esencial 5

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

Posteriormente, se realiza la multiplicación del peso de cada factor por el nivel asignado por los valores de la Tabla 10, y sumar esos productos para obtener el TFactor. La fórmula resultante es la siguiente:

$$TFactor = \sum(peso * nivel) \quad (11)$$

Por último solo resta aplicar la fórmula de peso de factores técnicos (TCF).

$$TCF = 0.6 + (0.01 * TFactor) \quad (12)$$

De igual forma que los TCF hay que considerar valores para estimar cada factor entre 0 y 5.

Tabla 11. Factores ambientales (Colomo, 2014).

Factor	Descripción	Peso
E1	Familiaridad con el modelo del proyecto utilizado	1.5
E2	Experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1
E4	Capacidad del análisis líder	0.5
E5	Motivación	1
E6	Estabilidad en los requerimientos	2
E7	Personal de medio tiempo	-1
E8	Dificultad en el lenguaje de programación	-1

* Sin experiencia, motivación o estabilidad de 0 a 2, Promedio 3, Amplia experiencia, motivación o estabilidad de 4 a 5.

Posteriormente, se realiza la multiplicación del peso de cada factor por el nivel asignado por la Tabla 12, y se suman los productos para obtener el EFactor. La fórmula resultante es la siguiente:

$$EFactor = \sum(peso * nivel) \quad (13)$$

Por último solo resta aplicar la fórmula de peso de factores técnicos (TCF).

$$EF = 1.4 + (-0.03 * EFactor) \quad (14)$$

Finalmente, para concluir con el paso 2, solo resta aplicar la fórmula de puntos de casos de uso (UCP).

$$UCP = UUCP * TCF * EF \quad (15)$$

3. Después de obtener el valor de los puntos de casos de uso (UCP), se procede a calcular las horas-hombre de acuerdo a la siguiente fórmula.

$$Horas - Hombre = UCP * 20 \quad (16)$$

El autor de la técnica sugiere usar 20 horas-hombre por UCP (*Use Case Points*). Por ejemplo, para un sistema de 60 UCP*20 hrs/hombre el resultado es un total de 1200 hrs/hombre. Lo que equivale a 30 semanas, de esta forma, un equipo de 5 personas desarrollarían el sistema en 6 semanas. Es decir, 75 semanas a 40 hrs por semana para una sola persona o 15 semanas para un equipo de 5 personas de tiempo completo.

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

Los tiempos estimados por etapa (análisis, diseño, programación, documentación, etc.), así como los costos por hora son criterio del equipo de desarrollo y dependen en gran medida de su experiencia.

Caso de estudio

Para fines de análisis de los métodos descritos en la sección anterior, se utilizó el Sistema de Monitorización Automática del Sistema de Gestión de la Calidad del Reactor TRIGA Mark III. El ININ (Instituto Nacional de Investigaciones Nucleares) del cual se desarrolló un *software* cuyo objetivo es regular las actividades técnico-administrativas en el Reactor TRIGA Mark III que permite la organización operativa del personal, con el fin de evitar actividades extemporáneas y así favorecer el cumplimiento de criterios del sistema de gestión de calidad.

a. Descripción del sistema

Para el reactor TRIGA Mark III los procesos involucrados en el sistema de gestión de calidad, se realizan de forma manual, siendo ésta una actividad delicada obliga a postergar actividades que requieren realizarse en tiempo y forma para el funcionamiento adecuado del reactor y la seguridad del personal que se encuentra en contacto con él. Por esto, resultó de especial interés dotar al sistema de gestión de calidad, con un sistema informático que automatizara los procesos involucrados a fin de mantener la certificación con las mejoras continuas y, en consecuencia, garantizar la integridad del personal, ya que, de no ser así, podría producirse un sobrecalentamiento por la falta de supervisión en éste y si fallara el extinguidor que tiene integrado, podría ocasionar una explosión en las instalaciones (Hernández, 2013).

El Sistema de Monitorización Automática del Sistema de Gestión de la Calidad del Reactor TRIGA Mark III, es un *software* desarrollado en lenguaje HTML (*Hypertext Markup Language*) y PHP (*Hipertext Preprocessor*) bajo la metodología Iweb, la cual se enfoca en crear, implementar y mantener las aplicaciones de un sistema Web. Así mismo en Iweb se deben establecer y utilizar principios científicos, con un enfoque sistemático y disciplinado al desarrollar, manejar y dar mantenimiento a los sistemas y aplicaciones que se basan en Web (Rossi, 2007).

b. Estimación de Costos

La dinámica de aplicación de los métodos de estimación de costos consistió en seguir los pasos descritos en cada uno de los modelos propuestos. Los parámetros utilizados se describen a continuación.

- En el modelo COCOMO se contabilizó un total de 6,080 líneas de código por lo que las fórmulas aplicadas corresponden a un proyecto orgánico, asimismo se obtuvo un total de tiempo de desarrollo de 8 meses-hombre correspondientes a un esfuerzo de 1,360 horas.

- En el método de puntos de función se determinó el tiempo de conteo para una aplicación desarrollada, se identificaron los alcances de la medición y los límites de la aplicación, además se realizó el conteo de las funciones de datos donde se analizaron los tipos existentes, en este caso la base de datos con un total de 24 ILF, el conteo de las funciones transaccionales dio como resultado 120 PFSA (puntos de función sin ajustar) en este paso se contabilizaron DETs y RETs del tipo EI, EO, EQ, por consiguiente el grado total de influencia (GTI) fue de 7. Finalmente el valor de puntos de función ajustados fue de 197.1 con dos personas de desarrollo, para dar un resultado de 1,382.4 horas-hombre de desarrollo.
- En el modelo basado en puntos de casos de uso se contabilizaron 6 puntos de actores y 115 puntos de casos de uso de acuerdo a su complejidad, el TFactor de factores técnicos fue de 0.715 y el EFactor de factores ambientales fue de 0.785. Tomando en cuenta los valores obtenidos y aplicando las fórmulas correspondientes resultó un esfuerzo de 1,358 horas-hombre.

Resultados y discusión

Los resultados obtenidos después de la aplicación de los tres métodos de estimación de costos son:

En el modelo COCOMO como entrada de información para su aplicación se contabilizaron 6,080 líneas de código en lenguaje PHP y HTML dando como resultado un esfuerzo de 1,360 horas trabajadas. En la aplicación del método de puntos de función se tomó en cuenta el número de DET's y RET's como entrada de información, así como EI, EO, EQ, ILF, EIF. Otro dato importante es que trabajaron dos personas en el desarrollo del sistema de información dando como resultado un total de 1,382.4 horas-hombre trabajadas como esfuerzo en total por ambos desarrolladores. Lo que equivale a 172.8 días con una jornada de trabajo de 8 horas cada uno. En el modelo de puntos de casos de uso se utilizaron como datos de entrada los casos de uso pertenecientes a UML y los factores técnicos y ambientales, dando como resultado un esfuerzo de 1,358 horas trabajadas.

Tras la investigación por conocer el sueldo de un programador en lenguaje PHP se encontró que la revista "SG Software Guru" publicó un artículo llamado "Estudio de salarios 2014" donde da a conocer los lenguajes de programación mejor pagados en México con la siguiente información, Tabla 12.

Tabla 12. Salario por lenguaje de programación (Galván, 2014).

Lenguaje	Muestra	Mediana	Media	Desviación estándar
Objetive C	21	$33,000	$40,141	$29,611
C	52	$25,725	$24,989	$13,762
Java	321	$25,000	$26,227	$16,342
VB	103	$25,000	$23,922	$11,809
C#	293	$24,000	$24,637	$13,549
Node JS	30	$23,970	$28,952	$20,037

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 1, Enero-Marzo, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 215- 228

PL/SQL	226	$23,450	$25,379	$16,436
Ruby	58	$23,250	$26,796	$19,870
Javascript	429	$22,700	$24,463	$14,372
Bash	47	$21,500	$24,959	$11,466
Python	40	$20,000	$23,910	$19,481
PHP	167	$18,000	$20,074	$14,016
Delphi	18	$18,000	$20,028	$9,681

Tomando en cuenta el valor de la mediana del pago por mes del programador en lenguaje PHP en México y aplicando esta cantidad al resultado obtenido por cada uno de los métodos. Se tiene que el pago por hora trabajada es de $102, entonces el costo total del proyecto según los métodos de estimación de costos aplicados sería.

- Modelo COCOMO: $138,720 pesos
- Método de puntos de función: $141,004.8 pesos
- Modelo de puntos de casos de uso: $138,516 pesos

Obtenidos los resultados se deduce que la estimación es correcta ya que en los tres métodos son muy aproximados.

Conclusiones

Una vez realizada la estimación al caso de estudio se observó que el modelo COCOMO, el cual representa el más extenso modelo empírico para la estimación de *software* publicado hasta la fecha, resulta como buena práctica ya que se ajusta fácilmente al tamaño del proyecto y sólo puede ser aplicado a un *software* terminado. El método basado en puntos de función, el cual se centra en contar funciones del usuario por medio de las interfaces gráficas y archivos de almacenamiento como la base de datos, a pesar de haberse contabilizado 120 puntos de función, que por regla menor a 100 puntos de función es poco fiable, el conteo resultó poco preciso, puesto que sobrepasó el resultado a comparación de los otros métodos aplicados. En la aplicación del modelo de puntos de casos de uso, el cual esta bien documentado y permite fácilmente conocer el costo de un *software* basado en la documentación de su desarrollo, debido a que el sistema es pequeño se tuvo que hacer un ajuste en la evaluación de factores ambientales y técnicos para lograr un resultado adecuado, a pesar de eso resulta una buena práctica para la estimación de costos de *software* para proyectos medianos y grandes en la etapa de análisis de requerimientos. En conclusión el modelo COCOMO y modelo basado en puntos de casos de uso resultan como buena práctica para la estimación de costos de *software*, el modelo COCOMO para proyectos terminados y modelos de puntos de casos de uso para proyectos en la etapa de análisis de requerimientos.

Como trabajo futuro se explorarán herramientas automáticas que estiman costos basados en COCOMO como son: Costar y COCOMO 81 mismas que forman parte de las líneas abordadas en este estudio.

Agradecimientos

Este trabajo fue realizado gracias al apoyo recibido de los proyectos: 3673/2014CE de la UAEM, PEI4-212752 del CONACYT, 3834/2014/CIA de la UAEM y "Operación y uso del Reactor, UR-001 del ININ".

Referencias

AGARWAL, B. B., *et al. Software Engineering and Testing. Sudbury,* Massachusetts: Jones & Bartlett, 2010, p. 155-156.

ÁLVAREZ, Jesús. *Revisión de los modelos de estimación de costos.* [en línea]. Modelos de estimación en proyectos de software. 2007, [Consultado el 10 de diciembre de 2014] Disponible en: [http://sinbad2.ujaen.es/cod/archivosPublicos/pfc/pfc_jesus_alvarez.pdf].

BREWER, Jeffrey L., et al. *Methods of IT Project Management*. West Lafayette, Indiana: Purdue University Press, 2013, p. 184-185.

CAMPOS, César. *Método de estimación de costos de construcción*. [en línea]. La Ingeniería de *Software*. 2012, [Consultado el: 30 de octubre de 2014] Disponible en: [http://laingeneriade*software*.blogspot.mx/2012_01_19_archive.html]

COLOMO-PALACIOS, Ricardo (ed). *Agile Estimation Techniques and Innovate Approaches to Software Process Improvement*. IGI Global, 2014, p. 54-56.

CUADRADO, Juan J., *et al. Estudio experimental de la conversión entre las unidades de medición funcional del software puntos de caso de uso e IFPUG.* Revista Española de Innovación, Calidad e Ingeniería del Sofware, 2008, 4 (2): p. 3-5.

DURÁN, Sergio E. *Puntos de función. Una métrica estándar para establecer el tamaño del software*. INEGI Boletín de política informática, México, 2003, 1 (6): p 41-52.

FARR, John V. *Systems Life Cycle Costing: Economic Analysis, Estimation, and Management*. CRC Press, 2011.

GALVÁN, Pedro. *Estudio de salarios 2014*. SG Software Guru, 2014, 1 (46): p. 16-19.

GARMUS, David., et al. Certified Function Point Specialist Examitation Guide. USA: CRC Press, 2011, 244 p.

GARZÓN, Ma. Luisa. *Informática. Temario A. Volumen IV. Profesores de Educación Secundaria Ebook.* Madrid, España, Mad, S. L., 2003, p 45-46.

GRAEF, Carlos. *Instalaciones del Reactor TRIGA Mark III*. [en línea]. 2008, [Consultado el: 16 de abril de 2015]. Disponible en: [www.inin.gob.mx/publicaciones/documentospdf/51%20INSTALACIONES.pdf]

HERNÁNDEZ, Teresita de Jesús, *et al*. *Monitorización Automática del Sistema Gestión de Calidad del Reactor TRIGA Mark III*. Tesis de licenciatura, Centro Universitario UAEM Valle de Chalco, México, 2013.

NUSSBAUM, Daniel A., *et al.* Cost Estimation: Methods and Tools. John Wiley & Sons, 2015, p. 261-262.

ORANTES, Sandra, *et al*. *Prototipo de una herramienta de apoyo para la estimación de costos, en la etapa de desarrollo de un proyecto de software*. Revista Digital Universitaria, UNAM, 2011, 12 (6): p. 4-5.

PRESSMAN, Roger S. *Ingeniería del Software: Un enfoque práctico*. España, McGrawHill, 2002. Quinta Edición.

ROSSI, Gustavo, *et al*. (ed.). *Web engineering: modelling and implementing web applications*. Springer Science & Business Media, 2007.

SALAZAR, Gabriela. *Estimación de proyectos de software: Un caso práctico.* Ingeniería y Ciencia, 2009, 5 (9), p. 123-143.

THOMAS, Pablo Javier, *et al*. *Análisis comparativo de estimación de esfuerzo en el desarrollo de software*. En XVII Congreso Argentino de Ciencias de la Computación. 2011.

TUYA, Javier, *et al*. *Técnicas cualitativas para la gestión en la ingeniería del software*. Netbiblo, 2007, p. 223-230.