Revista Digital Universitaria
1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

ARTICULO

PROTOTIPO DE UNA HERRAMIENTA DE APOYO PARA LA ESTIMACIÓN DE COSTOS, EN LA ETAPA DE DESARROLLO DE UN PROYECTO DE SOFTWARE

Sandra Dinora Orantes Jiménez y Antonio Ramírez Ramírez

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

Prototipo de una herramienta de apoyo para la estimación de costos, en la etapa de desarrollo de un proyecto de software

Resumen

En el presente artículo se analiza y se hace un enfoque sobre un tema de importancia para la Ingeniería de Software, que es la estimación de los costos de un proyecto de software. La intención es proponer y presentar el planteamiento y el desarrollo de una herramienta de software, que sirva de apoyo en esta tarea, tomando como base los modelos existentes y en sus versiones más recientes, en cuanto a métricas de la Ingeniería de Sistemas Computacionales, como lo son el COCOMO II (COnstructive COst MOdel, Modelo Constructivo de Costos) y el IFPUG (International Function Point Users Group, Grupo Internacional de Usuarios de Puntos Función), conocido este último como Métrica por Puntos de Función, siempre que el proyecto en cuestión se apegue a los lineamientos que fija la propia Ingeniería de software, que es mediada por organismos como la IEC/ISO (In-ternational Electro technical Commission/International Organization for Standardization) para garantizar, tanto al desarrollador como al usuario, que el proyecto se realiza bajo ciertas medidas de eficiencia y calidad.

Palabras clave

COCOMO II, IFPUG, estimación, métricas, ingeniería de software.

Prototype of a support tool to estimate costs in the development stage of a software project.

Abstract

This document, makes an analysis and get focus on very important theme from Software Engineering as is Costs Estimation of a software project's time, on intent to presents the proposal and development of a software tool that provides support in the evaluation and costs estimation as far as time and resources talk about, of a software project, with base in the most recent models of measurement on the Engineering of Computing Systems; such as COCOMO (constructive cost model) and IFPUG (international function point users group) known this last one like function points software measurement; whenever the project at issue, is become attached under rules that the own Software Engineering fixes, that is restrained by organisms like the ICE/ISO (International Elec-trotechnical Commission/International Organization for Standardization) to guarantee so much to the developer as to the user, which the project is

made keeping certain guarantees of efficiency and quality.

Keywords

COCOMO II, IFPUG, estimation, measurement, engineering of computing systems.

Introducción

En México y en algunos países de América Latina, por lo general, la creación y el desarrollo de software son realizados por ingenieros en sistemas, programación, electrónica o computación, pero realmente no dan a la Ingeniería de Software la importancia que merece. Por lo tanto, la industria del software ni siquiera se interesa en las cualidades y los alcances de los proyectos. En ocasiones ni siquiera se tiene idea de los mecanismos de evaluación que existen en otras partes del mundo.

Sin embargo los desarrolladores y empresas de software, que sí conocen y se ajustan a los estándares y los lineamientos de la administración de proyectos de software, entienden la importancia de aplicar los modelos de evaluación que existen o al menos tratan de seguir rutas adecuadas en el desarrollo de software. Cuando estas empresas son demasiado grandes o se atreven a contratarse para desarrollos importantes, deben saber que lo más seguro, comercialmente hablando, es comprometerse primero internamente, en la correcta aplicación de un ciclo de evaluación constante, que les permita establecer niveles de eficiencia que les ratifique el cumplimiento de contratos.

Todo ello sólo tiene importancia si el interés legítimo de cualquier empresa desarrolladora de software, es cumplir con lo que ofrece y subsistir económicamente, pues se ha demostrado que uno de los principales problemas de las empresas de servicios en casi toda América Latina, es el incumplimiento de los contratos, la mayoría de ellos en tiempo más que en contenido. Con esto se deduce que los problemas internos que padecen las empresas, no tienen que ver con la capacidad o el desempeño, sino más bien con la planeación.

Tal como en su momento la Ingeniería, que es una ciencia aplicada, reconoció a la administración como parte fundamental de la estructura empresarial, toca ahora a la Ingeniería de Software entender que la Administración de Proyectos sólo será completa si es capaz de implementar las métricas de evaluación en sus procesos de mejora continua.

Con el presente trabajo se pretende participar directamente en dicha mejora, pues se propone una alternativa confeccionada con una herramienta de software, capaz de auxiliar al Ingeniero de Software y con ello hacer posible, en alguna medida, que se abrevie y, quizá, se facilite su trabajo.

Esta herramienta de software es planteada desde el punto de vista de la Ingeniería de Software,

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

por lo que el término, en sí, representa una manera ordenada, probada y establecida para generar aplicaciones de cómputo.

Planteamiento

Existen diferentes metodologías o estándares de medición. Una de las más populares es la del International Function Point Users Group (IFPUG). La métrica del punto función es un método utilizado en Ingeniería del Software para medir el tamaño del software. Fue definida por Allan Albrecht, [1] de IBM, en 1979, para medir la funcionalidad entregada al usuario, independientemente de la tecnología utilizada para la construcción y la explotación del software. También pretende ser útil en cualquiera de las fases del proceso de vida del software.

Otra de las más utilizadas es el Modelo COCOMO II, [4] que nace como la adaptación del exitoso modelo COCOMO a los nuevos sistemas, herramientas y técnicas de desarrollo. Utilizando la base de su predecesor, COCOMO II implementa nuevos conceptos para ajustar más la estimación a las características de los proyectos. Por esto es que se definen tres sub modelos de estimación:

- Composición de aplicaciones. Utilizado para estimaciones en las primeras etapas del ciclo de vida de un proyecto software.
- Diseño anticipado. Preparado para sistemas en diseño, produce estimaciones tan fiables como el grado de evolución se haya alcanzado en el diseño de la aplicación.
- Post-Arquitectura. Empleado para sistemas completamente diseñados y como paso previo al comienzo de la etapa de desarrollo, así como para el seguimiento del esfuerzo invertido en el desarrollo del producto.

Con respecto al anterior modelo, también se han incluido nuevos “drivers de coste” y eliminado algunos otros que empleaba. También y debido a la capacidad de proceso de las nuevas computadoras, las herramientas de calibración permiten obtener resultados más ajustados y manejar bases de datos de proyectos más grandes. De esta manera las fórmulas que propone el Modelo COCOMO II son mucho más confiables. Además, la clasificación del modo de desarrollo en orgánico, semilibre o rígido, se realiza ahora de forma más compleja, con base en algunos factores de escala:

- Precedencia. Experiencia en aplicaciones del mismo tipo.
- Flexibilidad de desarrollo. Grado de sujeción del desarrollo a tiempos y requisitos.
- Resolución de arquitectura y riesgos. Identificación de riesgos en la aplicación.

- Cohesión del equipo. Nivel de integración del equipo de desarrollo.
- Madurez del proceso. Experiencia en el modelo de desarrollo.

De acuerdo con el primer modelo de COCOMO, esta evolución maneja también los rangos de constantes, que se utilizarán para las evaluaciones con el resto de las variables según el tipo de desarrollo que se está pronosticando, considerando los valores de la tabla 1 para el modelo básico.

MODO	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	2.80	1.20	2.50	0.32

Tabla 1. Coeficientes para los modos de desarrollo según COCOMO II

Éstas como algunas otras metodologías, han reportado cierta "exitosa aplicación" en la industria del software; sin embargo, no es posible establecer un parámetro de medición de resultados de dichas métricas en empresas de habla hispana en México y en la mayor parte de América Latina, por la simple razón de que no son ampliamente conocidas y, en ocasiones, aunque se conozca algo de ellas, no es tan fácil implementar una nueva norma, cuando se carece de la habilidad empresarial y la cultura informática necesarias.

Apoyar la aplicación de una de estas metodologías, con la creación de una herramienta sencilla y de fácil acceso, podría ayudar a las empresas de desarrollo de software en la implementación de mecanismos de autoevaluación y apoyo a la toma de decisiones dentro de sus organismos, para hacer eficiente la contratación de un nuevo proyecto.

La razón fundamental para darle importancia a la estimación de costos del software, es que en México se carece enormemente de metodologías confiables para la correcta administración de proyectos de software. Por un lado existe la necedad de los ingenieros en computación que insisten en diseñar programas sin llevar a cabo la realización de un contrato o cuando éste existe, la mayoría de las veces se da de manera verbal. Emprenden el desarrollo de software sin ninguna planeación o metodología a seguir, provocando con ello que el mercado de software local en muchos países de América Latina, incluyendo a México, sea prácticamente nulo, en comparación con el desarrollo y la distribución de software en países industrializados.

Por otro lado, la unión entre los conceptos generales del análisis económico y el mundo de la ingeniería del software, no es cosa sencilla. Por lo tanto, es necesario precisar que no hay

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

forma de hacer un análisis costo-beneficio sobre el desarrollo de software, si no se cuenta con un método de estimación de costos del software, así como el análisis de la influencia de los demás factores que intervienen en la creación del producto, el proyecto en sí o el entorno sobre la producción del mismo software.

Así, las técnicas de estimación son importantes porque proporcionan la parte esencial de una buena gestión de proyectos. Sin una aceptable capacidad de estimación de software, los proyectos podrían sufrir los siguientes problemas:

- Los programadores no tienen una base firme sobre la cual apoyarse para afirmar, a su cliente o patrón, que el tiempo y los recursos que le han sido otorgados para finalizar el producto son suficientes o, en su caso, poco realistas.
- Los analistas de software en México, comúnmente no tienen forma de realizar un análisis fiable de intercambio de piezas hardware—software durante la fase de diseño del sistema. Esto puede provocar un diseño en el que el desempeño del hardware quede por debajo de lo esperado, a causa de un software que ha costado mucho más de lo estimado.
- Los gestores del proyecto no saben cómo estimar el tiempo y el esfuerzo que conlleva cada fase y actividad, durante el desarrollo de un determinado proyecto.

Objetivo

Tradicionalmente según la forma en que es llevada a cabo la ingeniería del software, es como se determina el coste y la calidad del software producido. Esto hace importante, precisamente, a la ingeniería del software, a causa de los siguientes aspectos:

- El software debe ser cada vez más específico.
- El software es cada vez más costoso.
- El software precisa de ser liberado en menos tiempo, so pena de caer en obsolescencia.
- El software tiene cada vez más un impacto importante en el bienestar del ser humano.

Actualmente puede considerarse al hardware como la “caja negra” que contiene al software, siendo el mismo hardware el que determina el costo de un sistema de información. Por otro lado, el incremento en la demanda de programas cuyo uso sea cada vez más fácil, es un gran desafío para la Ingeniería del Software; primero, porque aumenta significativamente la productividad en el desarrollo de software y, segundo, porque incrementa la necesidad y la eficacia del software de mantenimiento.

Es importante hacer una distinción entre una metodología de Administración de Proyectos y una de Desarrollo de Software, ya que debe establecerse que ambos conceptos, aunque pertenecen a un mismo tema y se encuentran relacionados dentro de una organización, deben diferenciarse en forma clara y, además, deberá contarse con una metodología de administración de proyectos que sea consistente con cualquiera que sea la naturaleza del proyecto que se desarrolla. Por otra parte, debe contarse también con una metodología relacionada específicamente con el desarrollo de un proyecto de software, ya que en la práctica esto casi no sucede, ni tratándose, en ocasiones, de una organización dedicada a la industria del software. México no es la excepción.

Uno de los puntos principales de la Metodología de Desarrollo de Aplicaciones, es aquél que dice que los proyectos deben ser divididos en fases y antes de iniciar cada etapa debe existir un plan para cada una de ellas, que defina las actividades involucradas, así como los roles y las responsabilidades de cada elemento involucrado, para después pasar a la fase de elaboración del presupuesto de cada nuevo proyecto.

En este sentido, la herramienta software propuesta deberá proporcionar la suficiente utilidad al Ingeniero de Software, sin importar el giro o ámbito en el que se desenvuelva. Deberá ofrecerle una ventaja significativa, sobre cualquier método que emplee, para que realice las actividades que le permitan conocer, de manera anticipada y tal vez sólo aproximada, el nivel de gastos que ejercerá al emprender un proyecto de software. Al tener conocimiento de dicha información, podrá anticiparse a cualquier eventualidad o, inclusive, tomar la decisión de llevar a cabo o no un proyecto.

La utilidad adicional o ventaja que el Ingeniero de Software adquirirá al utilizar la herramienta propuesta, será la de contar, en un mismo instrumento, con la evaluación y el apoyo que brindan dos métricas por separado, que son el COCOMO II y los Puntos de Función. Como se ha comentado, estas métricas que fueron propuestas y demostradas en los ochenta, se han utilizado con éxito desde entonces en el ámbito comercial.

Desarrollo

Se propone una herramienta capaz de combinar y aplicar, a elección del usuario, los dos principales métodos de medición que existen en el campo de la ingeniería de software: COCOMO II e IFPUG, para auxiliarlo en la planeación de un nuevo proyecto de Desarrollo de Software. En dicha herramienta se permitirá al usuario establecer el ámbito del proyecto y establecer sus objetivos y metas. Asimismo, debe sugerir o permitir que el usuario ingrese estrategias, políticas y procedimientos, para alcanzarlos. Haciendo una clasificación, se tiene lo

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

siguiente:

1. Se podrá establecer un ambiente amigable y claro.
2. Se ofrecerá la elección entre las dos metodologías.
3. Se podrán delimitar las tareas o actividades del software.
4. Se permitirá ingresar los recursos a estimar (hardware, software, personal, etcétera).
5. Se podrá elegir el valor para las diferentes variables, en rangos preestablecidos.
6. Se proporcionará al usuario una predicción basada en un modelo probabilístico (no determinístico).
7. El usuario podrá proponer distintos valores, considerando que el objetivo de la estimación es reducir los costos e incrementar los niveles en aspectos de servicio y calidad.

Como resultado de estas opciones, deberá obtenerse información que permita establecer:

- El esfuerzo humano requerido para completar el proyecto.
- El tiempo que habrá de tomar (incluso cambiando la cantidad de personal).
- El Costo total del proyecto (desglosando rubros importantes).

Debe considerarse que el hecho de que exista determinado software, no significa que haya sido resuelto un problema en su totalidad, pues se estaría vanamente suponiendo que por que se cuenta con una determinada herramienta de software, se estaría agotando la problemática. Ésta no puede ser tan general como para abarcar todos los aspectos de la Ingeniería del Software. Basta observar cómo para cada tarea cotidiana, como suelen ser las actividades de una oficina, existe más de una versión proveída por más de un fabricante de aplicaciones de computadora, que de alguna forma coadyuvan en la realización de trabajos de oficina. Sin embargo, un software está lejos de resolver el trabajo del oficinista y de sustituir el elemento humano. Incluso, si un usuario no está medianamente capacitado en el manejo de dichas aplicaciones, éstas están lejos de ser una ayuda o una facilidad para desarrollar con mayor eficiencia su trabajo. Se vuelve un verdadero obstáculo, pues en lugar de permitirle avanzar en su trabajo, puede perder éste por no hacer ni siquiera lo que hacía antes de contar con la nueva herramienta.

Así, entre más general se pretenda una herramienta de software, ésta deberá abarcar muchas más funciones, pero sería extensa y difícil de aprender en su totalidad. Por lo tanto, no debe ser

un asistente tipo CASE, que resuelva el trabajo del ingeniero de sistemas. Tampoco sustituirá alguna de sus tareas. Sólo será un auxiliar en la etapa de planeación.

Al contrario, tal como sucede con el oficinista, que previamente debe aprender a realizar cada una de sus actividades de la manera tradicional y además conocer al menos de manera superficial el manejo de una computadora, para después hacer dichas tareas con ayuda de una herramienta de software, el Ingeniero de Sistemas podrá valerse de esta aplicación sólo si sabe cómo desarrollar sus actividades aún sin valerse de ninguna. En la figura 1 se muestra, en resumen, una abstracción de los elementos que componen la aplicación, en cuanto a la interacción con el usuario.

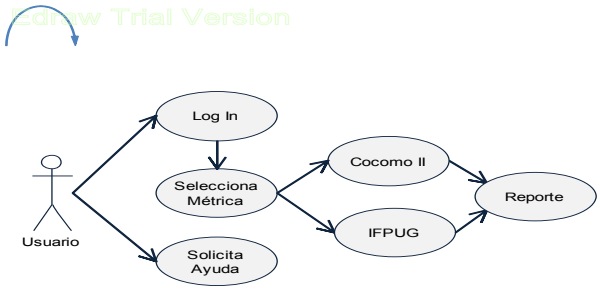

Figura 1. Representación UML de la primera vista de la Aplicación

Estos elementos son el inicio del sistema o aplicación, a partir del cual es posible finalizarlo o seleccionar una métrica para efectuar un llenado de ciertos datos, para obtener una aproximación o una estimación. La elección de una de esas "opciones" se representa en la figura 2, mediante un diagrama de estados o "momentos", en los que podría estar la aplicación, en caso de elegir "una métrica" en lugar de "finalizarla".

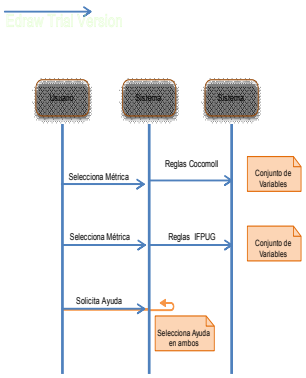

Figura 2. Diagrama de estados de las fases de la Aplicación.

El conjunto de las variables de la primera metodología empleada, así como la métrica en sí, está establecido desde principios de la década de los ochenta y, aunque se ha ido actualizando,

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

básicamente se refiere a las mismas áreas de evaluación, que son las siguientes:

- Atributos del producto
  - Fiabilidad requerida del software
  - Tamaño de la base de datos
  - Complejidad del producto
- Atributos del ordenador
  - Limitaciones en el tiempo de ejecución
  - Limitaciones de memoria principal
  - Volatilidad de la máquina virtual
  - Tiempo de respuesta
- Atributos de personal
  - Capacitación de los analistas
  - Experiencia en aplicaciones
  - Capacitación de los programadores
  - Experiencia en la máquina virtual
  - Experiencia en el lenguaje de programación
- Atributos de proyecto
  - Prácticas modernas de programación
  - Uso de herramientas
  - Limitaciones en la planificación

Una vez estimado el valor de cada uno de estos parámetros, se multiplican entre sí y el factor total obtenido se multiplica por el esfuerzo calculado mediante la fórmula nominal correspondiente (1).

$$E = a(Kl)b * m(x) E = a(Kl)b * m(x) \quad (1)$$

Esta fórmula nominal se refiere a las ponderaciones obtenidas en cada una de sus variables,

con base en las mediciones de diversos trabajos similares, desde finales de los sesenta y hasta la publicación del modelo en 1980, [4] donde "E" es Salario/mes y corresponde a una Media. "a" y "b" son constantes según el modo, ya sea orgánico, semilibre o rígido. "Kl" es la cantidad de líneas de código (K por miles). "m(X)" es el multiplicador que depende de los 17 atributos constantes.

La clasificación del modo de desarrollo en orgánico, semilibre o rígido, se realiza ahora en forma más compleja, con base en algunos factores de escala:

- Precedencia: experiencia en aplicaciones del mismo tipo.
- Flexibilidad de desarrollo: grado de sujeción del desarrollo a tiempos y requisitos.
- Resolución de arquitectura y riesgos: identificación de riesgos en la aplicación.
- Cohesión del equipo: nivel de integración del equipo de desarrollo.
- Madurez del proceso: experiencia en el modelo de desarrollo.

Dichos factores de integración, que visiblemente tienen más que ver con el factor humano que con el práctico, se han ponderado según la técnica derivada del propio COCOMO II, conocido como COQUALMO, tal como se muestra en la tabla 2.

Scale Factor	
(PREC) Precedentness	1.55
(FLEX) Development Flexibility	-
(RESL) Risk Resolution	2.00
(TEAM) Team Cohesion	1.40
(PMAT) Process Maturity	2.50

Tabla 2. Ponderación CoQualMo para los Factores de Escala

La técnica o modelo COQUALMO (COcomo QUAlity MOdels) está conformado por un modelo de inyección de defectos y un modelo de remoción de defectos. La salida de ese modelo de inyección es el número de defectos no triviales, que incluyen:

- Defectos críticos. Son los que causan caídas del sistema o pérdida irrecuperable de datos; o que en casos extremos podrían poner en riesgo a la gente;
- Defectos graves y moderados. Son los que impiden que ejecuten correcta o apropiadamente

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

funciones críticas del sistema.

Estos modelos están bajo desarrollo. Según una primera aproximación, las tasas de inyección base de defectos no triviales, son:

- 10 defectos de requerimientos por KSLOC;
- 20 defectos de diseño por KSLOC;
- 30 defectos de codificación por KSLOC.

Nótese que COQUALMO considera la generación de una tasa de defectos por tamaño de código, a diferencia de TSP, que considera la generación por tiempo de desarrollo. Las tasas base de COQUALMO pueden ajustarse con casi todos los multiplicadores y factores de ajuste, salvo FLEX.

Aplicación

Es un producto de software que podría considerarse una aplicación de escritorio ya que puede ser parte de la administración de proyectos, sirve para la programación de actividades, efectúa cálculos y determina mejores alternativas. Surge como desarrollo interno. Es mono usuario, pero puede distribuirse y posiblemente podría algún día ser parte de un integrado como Project. Incluye en un típico modelo MDI una sola ventana de bienvenida y un diálogo del tipo “Log-in” para validar al usuario en cuestión. Una vez en el programa se visualiza una ventana con opciones de selección, para elegir una de las métricas bajo las cuales se quiere realizar la estimación, dependiendo de las necesidades de cada usuario o las del desarrollo en sí. En dicha vista (figura 3) se ofrece además ayuda suficiente para saber qué deberá elegirse, pero de forma oculta. Visible solamente a petición del operador.

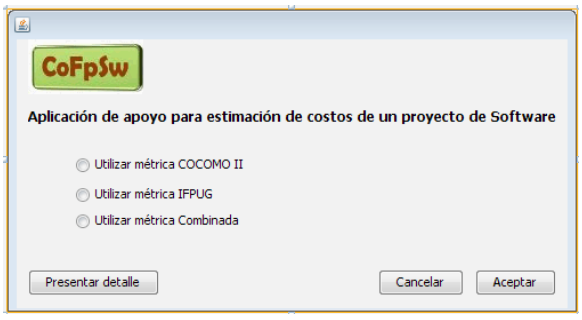

Figura 3. Vista principal de la aplicación

Como puede intuirse, en esta ficha puede hacerse la elección entre las métricas comentadas, así como en una tercera opción, que incluirá la combinación de las características de ambas técnicas. Estas vistas se desarrollaron con Swing de Java. En la figura 4 puede observarse cómo ocurre la selección de los parámetros, para cada uno de los factores involucrados.

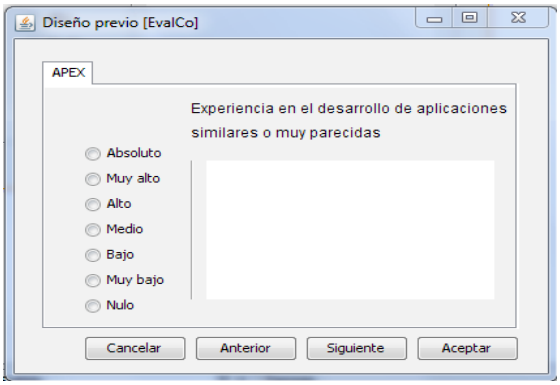

Figura 4. Vista del elemento que evalúa Cocomo II

Los cálculos que realiza el software tienen que ver con cada una de las metodologías utilizadas en el programa, siendo la finalidad obtener una "Estimación", empleando los algoritmos que dan solución a las tres tareas principales, que son:

- Estimar el tamaño
- Estimar el esfuerzo
- Determinar su Duración

A partir de estas evaluaciones se entrega al usuario un reporte medianamente completo, acerca de los costos que derivarán del proyecto a realizar. En una etapa posterior se podrá obtener en ese mismo reporte, una aproximación de los recursos necesarios para concluirlo en un tiempo determinado.

Los objetivos principales de la aplicación, son:

- Estimar tanto el tamaño como el esfuerzo
- Si es posible determinar la duración o costo del mismo. En tal caso se emplean indistintamente las metodologías COCOMO II e IFPUG.

Resultados

Con lo que se ha presentado en este trabajo, podría intentarse sugerir al director o ingeniero de desarrollo de un software, que debe tener siempre presentes los siguientes conceptos:

- Existe un tiempo mínimo de desarrollo. Su valor descansa en un sistema de ecuaciones, basado en unas constantes que permiten predecirlo para cada caso de forma fiable.
- Intentar dirigir un proyecto, más rápido de lo que su límite permite, es imposible o al menos nadie lo ha conseguido.

1 de junio 2011 • Volumen 12 Número 6 • ISSN: 1067-6079

- Incluso intentar dirigir un proyecto para cumplir con el tiempo mínimo de desarrollo implica un esfuerzo sobrehumano, un sustancial incremento de los costes y un aumento de los defectos en la aplicación desarrollada.

- Emplear elementos humanos en cualquier tarea y pretender ajustar su comportamiento a una “variable” del problema, conlleva su grado de riesgo.

Conclusión

Al proponer una herramienta de software de índole administrativa, debido al nivel con el que puede involucrarse en la fase de planeación, se plantea la opción de propiciar que la manera de hacer software en México se ajuste más a los lineamientos y objetivos de la organización, cuando ésta pertenece al giro de la industria del software.

Las políticas y procedimientos de la organización, pueden ser tan puntuales y al mismo tiempo tan flexibles, a grado tal que rijan y controlen los tiempos y los movimientos de quienes participan en ella, sin intervenir en su autonomía o la restricción de sus libertades. Del mismo modo, sujetar cualquier actividad a este nivel de control, puede ser benéfico y se reflejará en los resultados. Es benéfico para la tarea en sí, para quien se sirve de ella y para la organización entera.

Referencias

[1] Albrecht, A. J. Measuring Application Development Productivity, IBM Applications Development Symposium, Monterey, CA, USA, 1979.

[2] Symons, Charles R. Function Point Analysis: Difficulties and Improvements IEEE Transactions on Software Engineering, vol. 14, No.1, January 1988

[3] Software Engineering Standards Committee. IEEE Standard for Software Maintenance, IEEE-SA Standards Board June 1998

[4] Boehm, B., COCOMO II Model Definition Manual, http://sunset.usc.edu/research/COCOMOII, USA, 1999.

[5] Pressman,. Software Engineering. A Practitioners Approach. McGraw Hill. USA, 1997

[6] Sommerville, R., Software Engineering. Addison Wesley. USA, 1996.