Revista Cubana de Ciencias Informáticas
Vol. 7, No. 3, Julio - Septiembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 23-36
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Ingeniería y Gestión de software
Recibido: 11/03/2013 | Aceptado: 10/07/2013

Estimación en proyectos de software integrando los métodos de Boehm y Humphrey

Software projects estimation integrating Bohem and Humphrey method's

Yadira Ruíz Constanten[1*], Dasiel Cordero Morales[2]

[1] Facultad 2. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½ Torrens, Boyeros, La Habana, Cuba. C.P.: 19370.
[2] ISEC. Facultad 2. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½ Torrens, Boyeros, La Habana, Cuba. C.P.: 19370.

[*]Autor para la correspondencia: yadirar@uci.cu

Resumen

La necesidad de obtener datos objetivos que permitan evaluar, predecir y mejorar la calidad del software así como el tiempo y coste de desarrollo del mismo es imprescindible para garantizar resultados satisfactorios; evidenciándose la importancia de la medición del software. El valor de las mediciones aumenta cuando se realiza sobre modelos construidos en las primeras fases del proyecto ya que los resultados obtenidos permiten tenerlo bajo control en todo momento y corregir a tiempo posibles desviaciones. La continua proliferación de métricas y el gran volumen de datos que se maneja han puesto de manifiesto que las técnicas clásicas de análisis de datos son insuficientes para lograr los objetivos perseguidos. Específicamente los métodos de medición de tamaño tienen ciertas limitaciones, debido a que muchos de los resultados no son lo necesariamente satisfactorios y adecuados para algunos tipos de software. Algunos de los problemas más significativos están vinculados con la objetividad y la fiabilidad; la utilización del factor de ajuste; la precisión y la medición en distintas fases de desarrollo. En este documento se da una visión general del proceso de estimación del software. Se indican algunos fundamentos y se le ubica dentro de la gestión de proyectos de software. Se divide la estimación, en predicción del tamaño, del esfuerzo y del tiempo empleado para realizar el proyecto. Se hace especial énfasis, en los modelos propuestos por Barry Boehm y Watts Humphrey para estimar la duración de proyectos de software tratando de buscar puntos en común para una posible integración entre ellos.
Palabras clave: Estimación, medición, planificación, proyectos de software.

Abstract

The need for objectives data to asses, predict, improve software quality, and the development time and cost of it, is essential for success; showing the importance of software measurement. The value of the measurements is increased when it is realize based in models built in the early stages of the project because the results obtained allows to bring it under control in every moment and to correct in time possible deviations. The continued proliferation of metrics and the large volume of data being handled have shown that the classical techniques of data analysis are insufficient to achieve the objectives. Specifically the size measure methods have some limitations, because many of the results are not necessarily satisfactory and suitable for some types of software. Some of the most significant problems are

related to the objectivity and reliability, the use of the adjustment factor, the precision and the measure in different development stages. This document gives an overview of software estimation process. Also shown some basics and it is located within the project management software. It divides the estimate in the prediction of the size, effort and time spent for the project. The emphasis in the models proposed by Barry Boehm and Watts Humphrey to estimate the duration of software projects trying to find common ground for a possible integration between them.
Keywords: *Estimation, measurement, planning, software projects.*

Introducción

El trabajo a través de proyectos es la forma frecuente de actuación en el desarrollo de software. Diversas han sido las conceptualizaciones que se han hecho de un proyecto, manteniéndose como factor común en todas la presencia de un conjunto de etapas compuestas por actividades para alcanzar un objetivo en un tiempo determinado.

Los proyectos de desarrollo de software, al igual que el resto, deben iniciarse con un buen plan, pero lamentablemente, la planificación no es una tarea trivial. Uno de los aspectos que dificulta la labor de administradores y jefes de proyecto en torno a la planificación radica en la complejidad de realizar una estimación de costos y plazos realista. El proceso para crear una planificación de desarrollo exacta consta de tres pasos: estimar el tamaño del producto, estimar el esfuerzo necesario para su desarrollo y estimar la duración.

De la integración de estos tres pasos resulta la obtención de un intervalo de estimación que debe refinarse periódicamente, para aumentar la precisión a medida que avanza el proyecto. Para realizar dicha estimación se han desarrollado disímiles modelos. Los modelos matemáticos paramétricos que fueron de los primeros creados, están basados en un conjunto de variables de entrada significativas o parámetros (International Society of Parametric Analysts, 2008).

El funcionamiento de los modelos matemáticos de estimación para proyectos de software se fundamenta en la utilización de ecuaciones matemáticas mediante las cuales se obtiene el valor de un conjunto de variables dependientes de salida (esfuerzo, tiempo de desarrollo, entre otras) en función de los valores numéricos dados a otro conjunto de variables independientes de entrada (tamaño de la aplicación en líneas de código, fiabilidad requerida de la misma o complejidad). La mayor parte de esos modelos suelen operar mediante un proceso de dos pasos, en el primero se realiza una primera aproximación o estimación en función del valor de un conjunto reducido de variables independientes y en el segundo se precisa el resultado mediante la utilización de otro conjunto de variables que permite refinar los cálculos (Marbán, 2003).

Estimar consiste en determinar el valor de una variable desconocida a partir de otras conocidas, o de una pequeña cantidad de valores conocidos de esa misma variable. Es importante pensar en una predicción como en un rango más que como un simple número. Una estimación o predicción no es un objetivo, sino una valoración probabilística. El valor que se obtiene de una estimación es el centro del rango (Cerrillo, 2000).

La estimación en proyectos de software es una tarea extremadamente compleja, que requiere, entre otras cosas, disponer de información detallada del proyecto o de los proyectos a estimar, realizar una primera planificación del proyecto y conocer los recursos disponibles. Aun disponiendo de todos los medios y de la información necesaria, las estimaciones de los proyectos de software suelen errar, normalmente, pronosticando resultados menores de los que finalmente se producen.

Hay cuatro factores que influyen significativamente en las estimaciones: la complejidad y el tamaño del proyecto, el grado de incertidumbre estructural y la disponibilidad de información histórica [Navarro, 2012]. Una mala planeación y/o ejecución de un proyecto causa pérdidas relacionadas principalmente con los factores tiempo y costo, razones por las cuales éstos deben planearse y ejecutarse tomando en cuenta la premisa de que los proyectos se desarrollan para obtener una mejora significativa en la empresa, cumpliendo con las expectativas de calidad, costo y tiempo. La correcta definición y gestión de proyectos, tomando en cuenta dicha premisa determina su éxito o fracaso.

Materiales y métodos

Métodos de estimación

Los métodos de estimación de manera general, han sido diseñados para medir un determinado tipo de software. Por tanto, la aplicación de cada método depende particularmente del dominio del software o del tipo de desarrollo. Su evolución ha permitido obtener otros beneficios, tales como el perfeccionamiento del análisis de los riesgos de los proyectos o la posibilidad de realizar análisis cuantitativo sobre la eficacia de las disímiles propuestas de cambio de los procesos de desarrollo de software.

Los modelos de estimación que se han publicado desde los años 60 hasta la actualidad, fundamentan el progreso de los métodos de producción de software y el propio software en sí. Varios de ellos han quedado obsoletos, tales como el modelo Aaron (1969), el modelo de Wolverton (1974), el modelo de Walston-Feliz (1977), el modelo Doty (1977), el modelo Putnam (1978) y otros como el modelo SLIM (1979), el modelo COCOMO 81 (1981) y su actualización COCOMO II (1997) han ido evolucionando, siendo la base de las herramientas de estimación existentes en la actualidad (Gómez, 2008).

Jorgensen (2007) destaca que casi un tercio de los artículos de investigación escritos sobre estimación de costo en los años 90 tratan sobre Puntos de Función, registrándose en este período el pico más alto de interés en esta métrica. Por otra parte, hoy día los usuarios no están completamente satisfechos con los resultados obtenidos con los métodos de estimación; los errores en la estimación de esfuerzo son demasiado grandes, lo que lleva a reflexionar sobre si las métricas actuales son adecuadas a las necesidades de los líderes de proyectos o dueños de las aplicaciones (Robiolo, 2010).

Paralelamente al desarrollo y evolución de los modelos y métodos de estimación se han propuesto distintas calificaciones para éstos basándose en múltiples criterios en la medida que se ha desarrollado en la ingeniería de software la disciplina de estimación. En 1981 Bohem (1981) pretendió ordenar el conjunto de modelos de estimación existentes hasta ese momento proponiendo una clasificación que atendiera la disparidad de rutas existentes y que cada modelo pudiera ser incluido en alguna de las categorías definidas, logrando tener 7: Modelos algorítmicos, Estimación basada en expertos, Analogía, Parkinson, Price to win, Estimación de arriba abajo y Estimación de abajo a arriba.

En 1998 Capers Jones propone otra clasificación de los métodos de estimación (Jones, 2007) en la que aparecen seis metodologías agrupadas en dos grandes grupos: metodologías manuales, que no tienen herramientas que automaticen el cálculo y los métodos automáticos, que sí las tienen.

Entre 1999 el 2000 se propone otra clasificación, esta vez en seis métodos: modelos estáticos, experiencia de expertos, orientada al aprendizaje, modelos dinámicos, modelos estadísticos y modelos compuestos (Jones, 2007).

En el 2001 se resumen en dos todas las clasificaciones anteriores: Metodologías de estimación basada en Modelos Operacionales (del inglés, *Operational Models*, OM) y las Metodologías de estimación basada en modelos experimentales (del inglés, *Experimental Models*, EM). Estas a su vez se derivan en varias subcategorías: modelos

matemáticos, modelos basados en técnicas de Inteligencia Artificial, modelos comparativos, modelos dinámicos (Boehm, 2012).

En el 2012, P.K. Suri y Pallavi Ranjan (P.K., 2012) hacen un análisis de los métodos de estimación desde la década del 1970 concluyendo que en los últimos 5 años se han introducido diversos métodos que han tenido como objetivo incrementar la precisión de los resultados tras la identificación de algunos problemas teóricos (Kitchenham, 2009). Para ello la tendencia ha sido combinar diferentes métodos de estimación y utilizar a su vez, técnicas de inteligencia artificial, entre las que se destacan lógica difusa, sistemas basados en conocimiento y redes neuronales.

Generalmente en la actualidad cuando se hace referencia a la estimación de un proyecto se puede hacer referencia a tres grandes áreas: estimación de tiempo, de recursos y de costo. A pesar de tratar de distinguirlas a cada una de ellas de manera individual, su relación es muy estrecha. La estimación de recursos, costo y tiempo es un esfuerzo que requiere experiencia, acceso a buena información histórica (métricas) y valor para comprometerse con predicciones cuantitativas cuando la información cualitativa es todo lo que existe (Pressman, 2010).

Dificultades en el uso de los métodos de estimación

Tanto los métodos o modelos que quedaron obsoletos como los que constituyen las bases de las estimaciones de estos tiempos, tienen asociados un conjunto de inconvenientes que han dado al traste con unos y establecen limitantes para otros.

El modelo SLIM, cuya denominación procede de Software LIfecycle Management (Gestión del ciclo de vida del software) se fundamenta en el análisis de Putnam del ciclo de vida del software. Este modelo de estimación tiene como objetivo encontrar las variables estratégicas del proceso de desarrollo de software y derivar, de su comportamiento, un sistema de ecuaciones para luego introducirlas en una computadora para su estudio y análisis.

A pesar de estar considerado como uno de los más utilizados por las organizaciones, su uso tiene algunos inconvenientes al ser un método propietario. Desde que el modelo fue patentado, las ecuaciones no han sido editadas para el dominio público, aunque los algoritmos de funcionamiento pueden deducirse a partir de las publicaciones que se han realizado del modelo (Marbán, 2003).

Específicamente los métodos de medición de tamaño tienen ciertas reticencias, debido a que muchos de los resultados no son lo necesariamente satisfactorios y adecuados para ciertos tipos de software. Algunos de los problemas más significativos están vinculados con la objetividad y la fiabilidad; la utilización del factor de ajuste; la precisión y la medición en distintas fases de desarrollo.

Para algunos desarrolladores de aplicaciones informáticas considerar que el tamaño de un sistema está relacionado con la funcionalidad es algo natural. En ese sentido, en los inicios de la década de los años 80, se extendió el uso del método puntos de función, presentado en 1979 (primera publicación) por Allan Albrecht, quien lo refinó en 1984, para medir la funcionalidad. Este método mide tamaño independiente de la tecnología. En él, una entrada de complejidad media tiene 4 puntos de función y a una consulta media también le corresponden 4 puntos de función. ¿Qué equivalencia existe entre la entrada de datos y las consultas? Albretch estudió 24 proyectos de aplicaciones de negocios con un rango de tamaño desde 3000 a 318.000 líneas de código desarrolladas en DMS, PL/1 y COBOL. Hallando una relación entre entrada y consulta que le permitió decir que la cantidad de líneas de código utilizadas para una entrada de complejidad media es igual a la cantidad de líneas de código de una consulta media. Por otra parte, debe tenerse en cuenta además que esa igualdad de líneas de código se da para los lenguajes analizados y no necesariamente para cualquiera. En la actualidad queda demostrada la invalidez de esta relación para algunos lenguajes, con el uso de herramientas que automatizan las consultas y el acceso a los datos. De esta manera se

considera que queda descartado el principio de que el tamaño, es "independiente de la tecnología". Por lo tanto no sería tamaño según la definición clásica, sino esfuerzo normalizado para una o más tecnologías. Por otra parte, a los efectos de una estimación temprana, constituye otra debilidad de este método, asumir que se conocen los grupos de datos que utilizará el sistema. Asumiendo también que ese conocimiento es lo suficientemente amplio como para calificar esos grupos en complejidades baja, media y alta.

Tras la generalización del uso de metodologías orientadas a objeto, para el análisis, diseño y desarrollo de sistemas, se extendió el uso de la técnica de los Casos de Uso, desarrollado por Gustav Karner en 1993, que si bien no es privativa de la orientación a objetos, es claramente preponderante. Su utilización desde su surgimiento demostró que resulta más eficiente utilizar para la estimación, una técnica que tenga coincidencias con la metodología utilizada. Puntos por Casos de Uso clasifica, de manera similar a puntos de función, los casos de uso, en simple, medio y complejo según la cantidad de transacciones. Se define como transacción a un evento que ocurre entre un actor y el sistema a ser modelado. Boehm y otros (Boehm, 2000) consideran que esta clasificación es totalmente inadecuada a los efectos de realizar una estimación. El principal motivo es que un caso de uso no tiene un tamaño determinado, citando el ensayo realizado para *Rational Software* por John Smith en 1999 (Smith, 1999), quien considera como normal un promedio de 30 escenarios por caso de uso. Un caso de uso de este tipo podría llegar a tener 30 transacciones. Si a cada escenario le correspondiera una transacción, este caso de uso tendría igual factor de peso que uno de 8 transacciones. Es decir el rango superior, 8 a infinito, da igual valor para 8 transacciones que para infinitas, lo cual es un inconveniente. Por otra parte la agrupación de los requisitos del sistema en casos de uso puede ser totalmente arbitraria haciendo que el método dé valores muy distintos para la misma solución. En ese sentido un sistema en el que se desglosen las acciones vinculadas a gestionar (entiéndase la gestión definida por el patrón de casos de uso CRUD: insertar, mostrar, modificar y eliminar (del inglés, *Create, Read, Update y Delete*)) cierta información pueden considerarse uno, dos o cuatro casos de uso, teniendo en cuenta la complejidad del mismo, haciendo que la estimación llegue casi hasta cuadruplicarse según el desglose que se haga. Lo que indica que no debe tenerse en cuenta solo la cantidad de transacciones, sino además su complejidad.

Otra de las dificultades que se le imputan consiste en la dependencia que tiene este método de la persona encargada de efectuar la estimación así como de las que realizan la descripción detallada de los casos de uso. Estas pueden tener más o menos experiencia en esta actividad, más o menos facilidad de expresión y redacción, lo que implica que las descripciones sean más o menos escuetas y en función de ello podría existir un caso de uso con marcada implicación en la arquitectura del software, que tras clasificarlo, puede resultar menos importante que otro, solo por la cantidad de transacciones existentes. También puede ocurrir que se deba incluir algún caso de uso que en el primer instante no se consideraba como crítico y que después se decida modificar su clasificación o viceversa, es decir, que se decida eliminar tras dicha modificación.

Ha de tenerse en cuenta además, que si se estima con respecto a proyectos pasados, según se va evolucionando, esas estimaciones corren el riesgo de ser cada vez menos confiables mientras no se vayan adecuando paulatinamente los métodos utilizados. Asimismo, al no poder contar en la mayoría de las ocasiones con las condiciones idóneas, la productividad del nuevo grupo de trabajo no puede medirse en su totalidad por un trabajo previo.

Por otra parte, si se revisan las tendencias del desarrollo de software es evidente que su complejidad no disminuirá mientras tenga la posibilidad de progresar. Cada vez surgen nuevos y potentes lenguajes de programación, con aplicaciones que explotan las nuevas potencialidades de dichos lenguajes. La cantidad de líneas de código a desarrollar en el período en que se están estimando el tiempo y el esfuerzo, es una estimación en sí. Su implementación difiere en gran medida a la de hace algunos años tras el cambio en algunos de los paradigmas de

programación y la utilización de nuevas herramientas (Frameworks, IDE de desarrollo) y las facilidades que brindan para programar.

Pero el lenguaje es solo uno de los elementos en el desarrollo de software. Actualmente pueden existir aplicaciones tan complejas que el documento con las especificaciones de los requisitos puede tener cientos de páginas, teniendo entre los riesgos fundamentales, la volatilidad de dichos requisitos, que pueden variar según cambie el problema, los implicados, la interpretación y comprensión que puedan tener ellos del problema, así como la comunicación existente entre dichas personas y el equipo de desarrollo.

Dada la poca experiencia en el uso de métodos de estimación de las personas encargadas de planificar, así como el tiempo disponible para realizar el análisis previo, en la mayoría de las organizaciones, empresas y equipos de desarrollo de software cubanos, es una generalidad el hecho de que normalmente se estime la duración de un proyecto en un único momento, y no solo esto, sino también empleando un solo método, quedando imposibilitada así la realización de comparaciones en la búsqueda de alguna concordancia razonable para asegurar que las estimaciones realizadas son confiables.

Además se utiliza el mismo método para estimar proyectos con características totalmente distintas, en los que la presencia y el impacto de los indicadores definidos en cada método pueden tener comportamientos contradictorios y no estar en correspondencia con los resultados esperados. Esto influye también en que la recolección de datos para la confección de los históricos necesarios como base para futuras estimaciones no sea específica de un mismo tipo de proyecto informático.

Los riesgos a los que se exponen los que tienen en la estimación utilizando métodos cimentados exclusivamente en la experiencia, la única fuente de información para planificar, son innumerables. Por una parte pueden resultar útiles a pesar de la ausencia de datos históricos, se pueden aplicar en las fases tempranas del desarrollo brindando información necesaria al jefe de proyecto. Pero dependen de la experticia y requiere el involucramiento de múltiples expertos, exigiendo muchas horas de trabajo. En el caso de que se tenga solo una persona con estas características el riesgo es mayor, pues con ella se va todo el conocimiento adquirido.

No puede decirse de manera general, que la joven industria de software cubana tenga una vasta experiencia en la realización de aplicaciones informáticas para todos y cada uno de los sectores en los que puede incursionar, a pesar del gran número de ellas que existen en algunas áreas de desarrollo. En estas últimas, a pesar de tener un trabajo más extenso en este sentido, no se aplica a cabalidad la teoría defendida por Humphrey para emplear, en el momento de la estimación, la experiencia en la que puede basarse el equipo de desarrollo en la realización de aplicaciones similares.

Resultados y discusión

Boehm y Humphrey. Sus puntos de vista

Dos de las teorías con más reconocimiento internacional en la estimación de proyectos de desarrollo de software son las definidas por los investigadores Barry Bohem y Watt Humphrey. Cada una de ellas se basa en técnicas diametralmente opuestas pero muestran elementos mundialmente acertados y aceptados por las empresas desarrolladoras de software, que tienen en la calidad de sus productos y servicios, su meta fundamental.

El modelo original COCOMO se publicó por primera vez en 1981 por Barry Boehm y reflejaba las prácticas en el desarrollo de software de aquel momento. Durante los años 80, el modelo se continuó perfeccionando y consolidando, siendo actualmente el modelo de estimación de costos más ampliamente utilizado en el mundo, es el preferido para la estimación del esfuerzo cuando no se tiene información histórica a la cual recurrir. Además es el más documentado de todos los modelos de estimación de esfuerzo de las actividades de diseño, codificación, pruebas y mantenimiento.

Este método ha tenido varias versiones: COCOMO 81, ADA COCOMO, COCOMO II. Este último que es el más actualizado y al que cada año se le agregan mejoras está formado por tres modelos: el modelo de Composición de Aplicación, el modelo de Diseño Temprano y el modelo Post-Arquitectura, adaptándose al tipo y cantidad de información disponible en cada etapa del ciclo de vida de desarrollo. Estos modelos utilizan:

- Puntos Función y/o Líneas de Código Fuente para estimar tamaño.
- Un conjunto de 17 atributos, denominados factores de costo, que permiten considerar características del proyecto referentes al personal, plataforma de desarrollo, entre otros, que tienen injerencia en los costos.
- Cinco factores que determinan un exponente, que incorpora al modelo el concepto de deseconomía y economía de escala. Reemplazando los modos Orgánico, Semiacoplado y Empotrado del modelo COCOMO 81.

Por su parte Watts S. Humphrey fundó el Proceso de Software Personal (PSP) ampliando el proceso de mejora a las personas que realizan el trabajo de desarrollo de software, pues PSP se concentra en las prácticas de trabajo de los ingenieros en una forma individual, teniendo como principio servir para producir software de calidad, basados en la utilización constante de prácticas sanas de ingeniería de software. De la misma forma que enseña a cómo planear y darle un seguimiento al trabajo, utilizando un proceso bien definido y medido, estableciendo metas mesurables, y finalmente utilizando el rastreo constante para alcanzar dichas metas.

Como el objetivo es realizar mejores estimaciones, cada vez más eficientes, precisas y que sirvan para tener un modelo de comparación con datos reales para que al final se generen los mejores resultados finales, PSP comienza a estimar los tamaños de los productos que los ingenieros desarrollan personalmente basándose en el tamaño y en los datos de la productividad de cada ingeniero y con ellos estima el tiempo requerido para hacer el trabajo. Teniendo en cuenta que el tamaño del programa también será expresado en cantidad de Líneas de Código.
Para gestionar el tiempo empleando los principios de PSP, se deben hacer planes realistas, intentar seguir dichos planes, controlar el uso del tiempo y determinar los errores y cómo corregirlos. PSP realiza las estimaciones, tanto del tamaño del programa como de los recursos del mismo, con un método que se creó para estos fines que tiene por nombre PROBE (del inglés, PROxy Based Estimating) y que se aplica a todos los objetos que se encuentran en el diseño conceptual.
En aras de facilitar la recogida de estos datos, PSP provee varias plantillas con un formato específico para ir obteniendo todos y cada uno de los cambios en el sistema. De manera tal que se pueda llegar a entender cómo se aprovecha el tiempo.

Puntos de contacto entre las teorías de Boehm y Humphrey

Tras el estudio y análisis realizado al tratamiento que dan Barry Boehm y Watts Humphrey en sus teorías a la determinación de la duración de un proyecto de software, se puede concluir que sí existen varios puntos de contacto o posibles elementos que los integren. Cada uno de ellos da la posibilidad de establecer un nexo entre las dos teorías, lo que quizás tras su aplicación, puede conllevar a la obtención de mejores resultados en estos procesos de estimación.

Líneas de Código

El primero de los contactos está dado por el vínculo que tienen ambos puntos de vista para estimar tamaño en aplicaciones de software. En los dos casos se evidencia la plena dependencia del tiempo de duración con el tamaño de

los proyectos. COCOMO II realiza la estimación de tamaño según el modelo que utilice: para Composición de Aplicaciones usa Puntos Objetos, para Diseño Temprano y Post-Arquitectura usa Puntos de Función. Las ecuaciones formuladas para determinar el esfuerzo dependen del tamaño, por ende para determinar el esfuerzo nominal en dichos modelos, los puntos de función no ajustados tienen que ser convertidos a líneas de código fuente, considerando el lenguaje de implementación (ensamblador, lenguajes de alto nivel, lenguajes de cuarta generación, entre otros). Al aplicar la teoría de Humphrey no se puede, a menos que se esté trabajando con una persona experta en la realización de una actividad determinada, hacer una estimación temprana del tiempo de duración, pues se basa en la experiencia acumulada y ésta se obtiene tras la contabilización de las líneas de código. La combinación de ambas teorías podría estar en que un experto en la realización de aplicaciones específicas (multimedia, de gestión, realidad virtual, drivers, entre otros), puede estimar teniendo en cuenta su vasta experiencia, la cantidad de líneas de código que tendría el sistema a implementar sin tener que aplicar ninguna otra técnica para estimar tamaño y tener de esta manera otro valor con el cual comparar el resultado. De la misma forma al analizar el comportamiento del método PROBE planteado por Humphrey, se le pueden introducir el cálculo de la cantidad de líneas de código estimadas tras la aplicación de algún otro método y comparar entonces los resultados.

Por otra parte en el método COCOMO II se utilizan distintos factores de escala y multiplicadores de esfuerzo al ponderar los valores de las variables vinculadas en aras de lograr que en sus valores estén implícitos la incidencia de todos los factores que puedan tener alguna relación directa e indirecta con el proyecto. Dos de ellos están directamente vinculados a los procesos de calidad de software que de manera tan exhaustiva recoge Humphrey en su propuesta: factor de escala Madurez del Proceso (PMAT) y el multiplicador de esfuerzo Experiencia Personal (PREX).

Afiliación Madurez del Proceso (PMAT)

La determinación del PMAT está basada en el uso del Modelo de Madurez de Capacidad (CMM) del Instituto de Ingeniería del Software (SEI). El período de tiempo para medir la madurez del proceso es el momento en el que el proyecto comienza, hay dos formas de hacerlo. La primera toma el resultado de una evaluación organizada basada en el nivel alcanzado, según se muestra en la tabla 1 en la que se asocia el valor del nivel de CMM obtenido con su clasificación correspondiente (desde Muy Bajo a Extra Alto) y este a su vez se vincula con el valor de factor de escala para PMAT. La segunda está en correspondencia con las 18 Áreas de Proceso Principales (KPA´s por sus siglas en inglés) en el CMM.

Tabla 1. PMAT según niveles de CMM.

Nivel de CMM		**PMAT**	**Valor PMAT**
1- Mitad Inferior	Inicial	Muy bajo	7.80
1- Mitad superior		Bajo	6.24
2	Repetible	Nominal	4.68
3	Definido	Alto	3.12
4	Gestionado	Muy Alto	1.56
5	Optimizado	Extra Alto	0.00

Teniendo en cuenta las 18 Áreas de Proceso Principales (Administración de requerimientos, Planificación del Proyecto de Software, Seguimiento y supervisión del Proyecto de Software, Administración de subcontratos, Aseguramiento de la calidad, Administración de la configuración, Objetivo del Proceso de

Organización, Definición del Proceso de Organización, Programa de entrenamiento, Administración Integrada de Software, Ingeniería del Producto, Coordinación entre grupos, Revisión por Pares, Administración Cuantitativa, Administración de la calidad, Prevención de defectos, Administración de las Tecnologías de Cambio y Administración de los Procesos de Cambio) el procedimiento para determinar PMAT es establecer el porcentaje de cumplimiento de cada una de las áreas evaluando el grado de cumplimiento de las metas correspondientes. Para definir el nivel de conformidad se utiliza la escala porcentual propuesta por Likert.

Cuando se aplica el modelo evaluativo pueden ocurrir fundamentalmente dos situaciones que atenten contra las estimaciones y el proceso de desarrollo respectivamente, la primera de ellas es que la(s) persona(s) encargadas de determinar el porcentaje de conformidad para cada una de las KPA's no tengan precisión en la definición de los valores, debido principalmente al desconocimiento del estado de las mismas y la segunda es que se pueden obtener valores de la variable PMAT que demuestren un esfuerzo que se puede disminuir si se aumentan los porcentajes de cumplimiento en las diferentes KPA's.

La integración entre ambas teorías es natural teniendo en cuenta que PSP es el proceso de aplicación de los principios de CMM para ayudar y guiar a los ingenieros a realizar el trabajo con calidad acercando CMM al individuo. PSP cubre parcialmente 12 de las 18 KPA's que define CMM. Como puede apreciarse en la figura, éstas son las que están orientadas a los procesos a los que puede dársele un enfoque personal. Se propone integrar las prácticas de PSP al trabajo de los miembros del equipo de desarrollo, en aras de lograr que exista precisión y seguridad en los valores ofrecidos de los porcentajes de conformidad para cada una de las KPA's, lo que representaría realmente el estado de madurez de la organización. Logrando aumentar luego los porcentajes de cumplimiento de cada una de las KPA's, tras el mejoramiento notable en el proceso de desarrollo, que pudiera significar una disminución sustancial del esfuerzo necesario para el desarrollo de proyectos y por consiguiente en el tiempo de duración de estos.

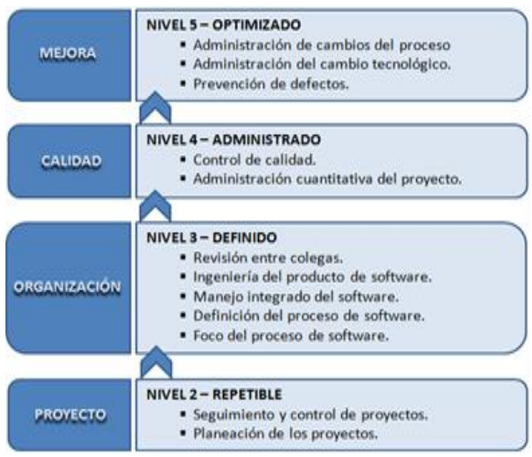

Figura. Elementos de PSP en CMM.

Experiencia Personal (PREX)

El multiplicador de esfuerzo PREX es uno de los parámetros de costo del Diseño Temprano que combina los tres parámetros de costo del modelo Post-Arquitectura (experiencia en la aplicación (AEXP), experiencia en la plataforma (PEXP), y experiencia en las herramientas y lenguajes (LTEX)). Cada uno de estos indicadores recibe influencia directa de la aplicación de los puntos de vista defendidos por Humphrey pues de forma general capturan la experiencia del personal involucrado en el proceso de desarrollo.

PSP se concentra en las prácticas de trabajo de cada especialista de manera individual para desarrollar software de calidad, a través de la utilización de un proceso planeado, medido y bien definido, dándole mucho peso a la experiencia del individuo y el análisis de los datos históricos registrados de proyectos anteriores que sirven de guía y ayuda en el proceso de desarrollo. Se propone integrar las prácticas propuestas por PSP para controlar el trabajo, dominio y experiencia de los ingenieros sobre las herramientas, lenguaje, plataforma y aplicaciones específicas en que este trabaja o ha trabajado, teniendo así una medida de la madurez y las potencialidades que se pudieran aprovechar ante un proyecto con características específicas, pudiéndose distribuir las tareas y responsabilidades de manera eficiente, lo que podría aumentar la productividad y disminuir el esfuerzo en el proceso de desarrollo y a su vez el tiempo de terminación de los proyectos. Además a partir del control que proporciona PSP sobre el trabajo de los individuos, se tienen criterios y datos históricos para realizar estimaciones más fáciles y confiables.

Reutilización de código

La reutilización de código se refiere al comportamiento y a las técnicas que garantizan que una parte o la totalidad de un programa informático existente se puedan emplear en la construcción de otro programa. De esta forma se aprovecha el trabajo anterior, se economiza tiempo y se reduce la redundancia. Tanto COCOMO como PROBE dan tratamiento al código reutilizado y lo tienen en cuenta para estimar el tamaño del software en Líneas de Código (LOC, por sus siglas en inglés).

El tratamiento que hace COCOMO del software reutilizado utiliza un modelo de estimación no lineal para calcular las LOC equivalentes al nuevo desarrollo (*ESLOC*) a través de las fórmulas (1) y (2):

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

$$ESLOC = ASLOC * \frac{[AA + AAF + SU * UNFM)]}{100}, \quad si\ AAF > 0.5 \quad (2)$$

donde:

ASLOC: es la cantidad de LOC que se va a adaptar,

AA (Assesment and Assimilation): es el grado de valoración y asimilación necesario para decidir cuándo un módulo de software reutilizado por completo es apropiado para la aplicación,

SU (Software Understanding): es el porciento de esfuerzo de reutilización debido a la comprensión del software,

UNFM (Programmer Unfamiliarity): es el indicador de la familiaridad del programador con el software y

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 3, Julio - Septiembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 23-36
http://rcci.uci.cu

AAF (Adaptation Adjustment Factor): es el factor de ajuste de la adaptación, cuyo valor se calcula por la fórmula (3) teniendo en cuenta tres parámetros del grado de modificación

$$AAF = 0.4 * DM + 0.3 * CM + 0.3 * IM \quad (3)$$

donde:

DM: es el porcentaje de diseño de software que es modificado para adaptarlo a los nuevos objetivos y al entorno,

CM: s el porcentaje de código software adaptado que es modificado para adaptarlo a los nuevos objetivos y al entorno e

IM: es el porcentaje de esfuerzo requerido para integrar el software adaptado dentro de la totalidad del producto y comprobar el producto resultante comparado con la cantidad de esfuerzo normal de integración y pruebas para software de un tamaño similar.

En las fórmulas (1) y (2) los valores obtenidos por PSP tendrían una incidencia directa sobre las variables ASLOC, AA y UNFM.

Según PROBE la categoría de reutilizado se utiliza sólo para partes del código sin modificaciones. Cuando se modifican los programas existentes, el programa sin modificar es la base (tamaño total del programa sin modificar antes el desarrollo), y se estiman las adiciones (tamaño del código añadido para mejorar el programa base), modificaciones (LOC que se añaden al código existente como parte de una modificación) y supresiones (tamaño del código que es eliminado del programa base por no ser necesario). Incluso si el programa no se modifica, no se considera reutilizado a menos que se destine específicamente para su reutilización.

En ambas teorías, tras aplicar el método particular de cada una para la reutilización del código, el valor obtenido de estimar la cantidad de LOC reutilizadas es usado directamente para estimar el tamaño total del software y este a su vez es usado en los cálculos necesarios para estimar el tiempo de duración de los proyectos. Se propone utilizar en COCOMO la forma en que PROBE estima el código reutilizado y viceversa.

Como elemento distintivo entre estas dos teorías está el hecho de que ambas tienen su base fundamental en la calibración de las variables que se utilizan para ajustar los valores obtenidos según los distintos factores que inciden en el resultado. Otras vías de integración pueden existir si se redefinen los parámetros a calibrar en función de los intereses de una organización específica, ya que ha de tenerse en cuenta que las medidas obtenidas, fundamentalmente al aplicar COCOMO II son el resultado de haber trabajado con factores inherentes al campo de acción de Boehm.

Validación de la propuesta

Como parte de la validación de la propuesta, se propone realizar, antes de aplicar el experimento, un diagnóstico como ejercicio inicial, con la aplicación de técnicas de recopilación de información. El objetivo es obtener, de los encuestados, la mayor cantidad de información posible sobre el nivel de conocimiento,

experticia en el tema y pertinencia de la propuesta. Se propone realizar este diagnóstico base a 30 especialistas, entre los que estarían líderes de proyectos, planificadores, personas con experiencia en temas de gestión de proyectos y calidad de software. Lo que hará que la muestra sea seleccionada con carácter intencional. Este diagnóstico permitirá además identificar si los miembros de la población seleccionada tienen criterios similares.

Se tomará como población los proyectos informáticos de los dos centros de desarrollo de software de Facultad 2 de la Universidad de las Ciencias Informáticas: Telemática (TLM), dedicado al desarrollo de productos y servicios en el área de las telecomunicaciones y la seguridad informática e Informatización de la Seguridad Ciudadana (ISEC), dedicado a desarrollar aplicaciones que apoyen la gestión de información en los diversos órganos que brindan seguridad a los ciudadanos. Se tomará como muestra ocho de estos proyectos de software teniendo en cuenta los que han realizado estimaciones más precisas, teniendo poca desviación con respecto a los resultados reales y los que han tenido que realizar varias estimaciones y a su vez, continuos ajustes en los cronogramas. Esta selección se realizará utilizando la técnica de muestreo no probabilística, muestreo intencional y la muestra seleccionada corresponde a un 40% de la población, por lo que se considera que se logrará la representatividad de los elementos heterogéneos que se analizan, así como la significativa representación de los mismos, pues por las propias consideraciones de la técnica, se seleccionan elementos representativos con alta posibilidad de brindar mayor información (Hernández, 2002).

Aplicación del experimento a la muestra seleccionada

El proceso de aplicación de la propuesta se realizará a través de tres actividades: Preparación del personal, Implantación y Recopilación de Resultados. Como parte de la preparación del personal se iniciará mediante la presentación de la visión de la propuesta a los Directores de los Centros de Desarrollo TLM e ISEC exponiendo las ideas generales y los presuntos resultados esperados. De este encuentro se recogerán y analizarán las opiniones emitidas. Luego se realizará un taller con los líderes, planificadores, gestores de la calidad y de control y configuración de cambios de los proyectos seleccionados, para explicar detalladamente la propuesta. Los criterios expresados también serán sometidos a revisión. Una vez realizados los dos primeros momentos de la preparación del personal se propondrá un cronograma de ejecución para la aplicación de los elementos contenidos en la propuesta como parte de la Implantación. Para la Recopilación de Resultados se propone aplicar a la muestra antes seleccionada un instrumento, cuyas características aparecen detalladas en la Tabla 2 y que tendrá como objetivo obtener información sobre los resultados de la implantación de la propuesta.

Tabla 2. Caracterización del instrumento de captación de datos.

Tipo de pregunta	Descripción	Cantidad
A	Preguntas de afirmación o negación (Sí/No)	8
B	Preguntas donde se evalúa, categoriza o califican los indicadores	14
C	Preguntas donde se recogen criterios libres de los encuestados	5

Las preguntas de tipo A tienen por objetivo comprobar la factibilidad de aplicación de la propuesta, así como la pertinencia de la misma. Las preguntas de tipo B tienen por objetivo evaluar el nivel en que se encuentra un indicador específico de los que forman parte de la propuesta. Las preguntas de tipo B serán cuantificadas en una escala de valores de uno a cinco, siendo este último el máximo valor posible, que estará en correspondencia con los elementos de cada indicador que lo lleven a ser el indicador con impacto más positivo en la estimación de la duración de proyectos de desarrollo de software. Las preguntas de tipo C tienen por objetivo identificar determinados factores con incidencia directa en los proyectos o algún criterio específico que pueda ser tenido en cuenta como parte del proceso de estimación del proyecto. Esta descripción de las categorías es tomada de referencia de un diagnóstico de este tipo [Piñero, 2007]. Luego se hará un análisis comparativo de la aplicación del último instrumento con el diagnóstico inicial y se valorarán los resultados.

Conclusiones

Tras el análisis realizado a los procesos de estimación de software como parte de la planificación de la gestión de proyectos, a varios métodos y diferentes enfoques, se puede concluir que, la mayoría de los especialistas se rigen y sugieren el proceso ideado por Barry Boehm, aunque se reconoce la importancia de aplicar la teoría de Humphrey en aras de obtener sistemas con calidad e ir formando mejores especialistas. A pesar de ello existen elementos que garantizan la integración entre las dos teorías analizadas: la de Barry Boehm y la de Watts Humphrey, y que además puede establecerse dicha integración, pues existen algunos factores de escala y multiplicadores de esfuerzo utilizados en el método de Boehm sobre los que el componente fundamental de PSP, la experiencia adquirida tras estimar continuamente basado en datos propios recogidos en un histórico, ejerce una influencia directa, que puede ser empleada además en las ecuaciones planteadas por Boehm para valorar la reutilización de código. La palabra de orden en este caso es calibración, ya que al evaluar varios de los parámetros expuestos al aplicar COCOMO II se abren las puertas para hacer uso de los resultados obtenidos por el método de Humphrey. De la misma forma pueden establecerse nexos en la estimación del esfuerzo al intercambiar la forma de obtener el tamaño de la aplicación, dada en líneas de código fuente (del inglés, Line of Code, LOC).

Por otra parte son las tradicionales Puntos de Función y Líneas de Código (LOC), las técnicas más utilizadas para estimar tamaño y tiempo de desarrollo de un proyecto de software, a pesar de estar tomando auge otras como Puntos de Características, MK II, Puntos de Objeto y Puntos de Casos de Uso que están ideadas para garantizar la estimación en los actuales modelos de desarrollo de software.

Por la etapa en la que se encuentra la investigación, no se muestran los resultados de la validación de la propuesta, sino el diseño del experimento que se utilizará.

Referencias

- BOEHM, B., Abts, C., BROWN, A., CHULANI, S., CLARK, B., HOROWITZ, E., MADACHY, R., REIFER, D., STEECE, B., Software Cost Estimation with COCOMO II, New Jersey, Prentice Hall, 2000, 544 p.
- BOEHM, B., COCOMO II Model Definition Manual. [en línea] [Consultado el 15 de marzo 2012]. Disponible en: [http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html].
- BOHEM, B., Software Engineering Economic, Prentice Hall, USA, 1981, ISBN 0-13-822122-7.
- CERRILLO, D., Estimación del software, Informe Escuela Superior de Informática, Universidad Castilla la Mancha, España, 2000.
- GÓMEZ, F., Aplicación para la estimación software basada en el modelo SLIM., Tesis Maestría, Universidad Politécnica de Madrid. España. 2008.
- HERNÁNDEZ LEÓN, R. A., COELLO GONZÁLEZ, S., El paradigma cuantitativo de la investigación científica, Ciudad de la Habana. Editorial Universitaria. 2002. 959-16-0343-6.
- International Society of Parametric Analysts, Parametric Estimating Handbook 4th Edition, Vienna, 2008, ISBN 0-9720204-7-0, p. 237.
- JONES, C., Estimating Software Costs., 2nd edition, McGraw-Hill, New York, 2007.
- JORGENSEN, M. y. Shepperd, M., A systematic review of software development cost estimation studies, IEEE Transactions on Software Engineering, 2007, 33(1): p. 33-53.
- KITCHENHAM and E. MENDES, Why comparative effort prediction studies may be invalid, Proceedings of the 5th International Conference on Predictor Models in Software Engineering. Vancouver (Canadá), 18-19 de mayo de 2009, 2009.
- MARBÁN, G., Modelo matemático paramétrico de estimación para proyectos de data mining (DMCOMO), Tesis doctoral, Universidad Politécnica de Madrid, España, 2003.
- NAVARRO, J., Planificación de un proyecto de Software, [en línea] [Consultado el: 12 de octubre de 2012]. Disponible en: [http://agu.inter.edu/jnavarro/comp3400Lec12PlanifPrySoft.pdf]
- PIÑERO PÉREZ, Y., "Metodología para la gestión de contratación en proyectos de desarrollo de Software Educativo", Tesis de Maestría, Ciudad de la Habana, Universidad de las Ciencias Informáticas, 2007.
- PRESSMAN, R., Software Engineering. A practitioner´s Approach. 7th Edition, New York, McGraw Hill, 2010, 889 p.
- ROBIOLO, G., Transacciones, Objetos de Entidad y Caminos: métricas de software, basadas en casos de uso, que mejoran la estimación temprana de esfuerzo. Tesis doctoral, Universidad Nacional de La Plata, Argentina, 2010.
- SMITH J., The Estimation of Effort Based on Use Cases, [en línea] Rational Software white paper, 1999. [Consultado el 18 de septiembre de 2012]. Disponible en: [http://sce.uhcl.edu/helm/rationalunifiedprocess/papers/pdf/TP171.pdf].
- SURI, P. K.; PALLAVI RANJAN. Comparative Analysis of Software Effort Estimation Techniques, International Journal of Computer Applications, 2012, 48(21): p. 12-19.