Revista Cubana de Ingeniería Vol. X, No. 1, enero - abril, 2019, pp.69 - 78

Artículo Original

Evaluación del impacto del uso de un framework en la estimación del esfuerzo de desarrollo del software

Evaluation of the impact of the use of a framework in the estimation of the software development effort

David Santana Ballate[1], Fernando Arteaga Céspedes[2], Vanessa Danae Muñoz Castillo[1], Anaisa Hernández González[1]
[1]Universidad Tecnológica de La Habana, José Antonio Echeverría, Cujae, La Habana, Cuba
[2]AXES Network Comany, Québec City, Canada
Correo electrónico: dsantana@ceis.cujae.edu.cu

Recibido: 27 de agosto de 2018 Aprobado: 29 de noviembre de 2018

Resumen
Actualmente el desarrollo de software se encuentra favorecido por herramientas que generan de forma automática una parte del código fuente. En muchas ocasiones, a la hora de realizar las estimaciones, no se tiene en cuenta dicho factor; obteniéndose resultados incorrectos. En el trabajo se presenta una aplicación que permite evaluar la relación que existe entre el uso de un *framework* y la generación de código fuente. El resultado obtenido permite conocer la cantidad de líneas de código fuente creadas por el equipo de desarrollo, las generadas por el marco de trabajo y las modificadas por los desarrolladores a partir de las generadas. Para ejemplificar se utiliza como *framework* de desarrollo a Yii 2.0.

Palabras claves**:** estimación del esfuerzo, Herramientas de generación de Código, Marcos de trabajo

Abstract

The Currently, software development is favored by tools that automatically generate a part of the source code. In many occasions, when making the estimates, this factor is not taken into account; getting incorrect results. In the work an application is presented that allows to evaluate the relationship that exists between the use of a framework and the generation of source code. The result obtained allows knowing the amount of source code lines created by the development team, those generated by the framework and those modified by the developers from the ones generated. To exemplify it is used as a development framework to Yii 2.0.

Key words**:** effort estimation, Code generation tools, Frameworks

INTRODUCCIÓN

En las últimas décadas, con el aumento del uso de Internet y de los sistemas computarizados, se ha elevado el índice de desarrollo de aplicaciones informáticas. Como resultado de la necesidad de aplicaciones informáticas, la industria del desarrollo de software ha estado experimentando un alto grado de competencia entre las empresas del sector [1,2]. Por tal razón, las empresas desarrolladoras de software, se han dado a la tarea de garantizar la calidad, tanto en el proceso de desarrollo de software, como en el producto que finalmente se obtiene.

La realización de estimaciones alejadas de la realidad, hacen que se incumpla con el presupuesto y con el tiempo, lo que atenta contra esta meta [3,4]. Mediante estudios realizados en países desarrollados, especialmente en los Estados Unidos y Europa, se ha demostrado la necesidad de medir la eficiencia y calidad del proceso y del producto de software. Dicha acción contribuye considerablemente a la detección de errores, defectos y fallos, así como en la corrección de los mismos [1].

Existen varias causas por las cuales los proyectos sufren afectaciones en cuanto al cumplimiento del tiempo y del presupuesto planificado. Una de dichas causas, es la realización de estimaciones lejanas a la realidad [1,5].

La estimación, en el mundo de la Industria de Software, se ha convertido en uno de los principales retos para la gestión de proyectos. Dentro de este proceso se estudian un grupo de variables e indicadores que proporcionan un grado de riesgo aceptable; aunque los resultados no sean exactos [1, 6,7].

La estimación requiere de experiencia, acceso a buena información histórica y valentía para comprometerse con predicciones cuantitativas cuando la información cualitativa es todo lo que existe [7]. Uno de los aspectos asociados a la calidad de los productos de software es la calidad de sus líneas de código fuente; por tal razón, es importante evaluar y medir dicho elemento [1, 6,7]

Actualmente, el desarrollo de software se encuentra favorecido por herramientas que generan de forma automática, parte del código fuente del sistema. En este sentido se destacan los marcos de trabajos. Estos marcos, a partir de un modelo de bases de datos, generan el mecanismo de acceso a la misma y varias de las vistas que se utilizan en la aplicación, disminuyendo considerablemente el esfuerzo requerido.

En muchas ocasiones, a la hora de realizar las estimaciones, no se tiene en cuenta dicho factor; obteniéndose entonces, resultados incorrectos**.**

La medición del código fuente es una acción bastante compleja para ser realizada de forma manual. Esto se debe fundamentalmente al elevado número de líneas de código que posee cualquier producto de software. Por tal razón, es recomendable realizar dicha medición de forma automatizada; logrando disminuir el esfuerzo humano.

El proceso de estimación de esfuerzo y tiempo de desarrollo de un proyecto informático se realiza, en muchas ocasiones, a través de métodos. Cuando estos métodos son bien empleados logran dar un resultado bastante acertado y confiable. Estos resultados se logran obtener mediante el estudio de diferentes variables e indicadores relacionados con el proyecto [7,8]. Los métodos de estimación basados en puntos de casos de uso y puntos de función, se encuentran entre los más utilizados para la estimación del esfuerzo y el tiempo de desarrollo. Pero, ¿existe alguna relación entre el uso de herramientas de generación de código (dígase, *IDE*, *Framework*, entre otros) y los métodos de estimación?

Del estudio del funcionamiento de los principales métodos se detectó que, por lo general, no evalúan el impacto de los *framework* en la generación automática de código fuente. En tal sentido, este trabajo presenta los resultados del desarrollo de una herramienta que permite identificar el porcentaje de código fuente generado por el *framework* y el implementado y modificado por los desarrolladores.

El valor práctico del trabajo realizado radica en la presentación de una herramienta informática que permita, a partir de la carga de una aplicación informática desarrollada usando un *framework*; identificar el porcentaje de impacto de la propia herramienta en la generación de código fuente, de forma automática. El uso de esta herramienta en un mayor número de aplicaciones, permitirá disponer de una base de datos que puede servir de referencia para ajustar los resultados de la estimación cuando se emplean herramientas de generación de código.

MATERIALES Y MÉTODOS

En la investigación realizada fue necesario profundizar en los métodos de estimación definidos en la literatura y utilizados por los desarrolladores de software. El objetivo de dicha revisión fue evidenciar la no contemplación del impacto que tienen las herramientas de desarrollo, sobre la generación de código fuente de forma automática.

Para evaluar el impacto del uso de *framework* o marco de trabajo en el desarrollo del producto informático, se elaboró una herramienta para la recolección de datos y un procedimiento de trabajo, los que serán descritos en el acápite siguiente.

Con este propósito, se trabajó en dos etapas, en las que se ejecutaron las tareas que se describen a continuación [9]:

1. Evaluar cómo los métodos de estimación que se utilizan en el desarrollo de software contemplan las bondades de las herramientas de generación de código automático.
    - Identificar los métodos de estimación utilizados en el desarrollo de software.
    - Identificar las herramientas utilizadas en la generación de código.
    - Estudiar el funcionamiento de los principales métodos de estimación, identificando los que analizan el impacto de las herramientas de generación de código automático en la fase de desarrollo del proceso de software.
2. Desarrollar la herramienta que permita identificar el porcentaje de código generado por el Framework Yii2 en proyectos de baja, mediana y alta complejidad.
    - Identificar los criterios de clasificaciones de proyectos informáticos, en cuanto al nivel de complejidad.
    - Modelar, diseñar e implementar un prototipo de herramienta que permita medir la generación de código automático en proyectos desarrollados por el Framework Yii2.
    - Probar la solución propuesta.

Durante el desarrollo del trabajo se emplearon los métodos fundamentales de investigación científica teóricos y empíricos como una vía para llegar a los resultados propuestos. Entre los métodos teóricos se empleó el análisis y síntesis de la información obtenida a partir de la revisión de la literatura, el estudio de los métodos de estimación y la observación de su uso en el desarrollo de software. Dentro de los métodos de investigación empíricos se utilizaron varios, entre los cuales se destaca: la revisión de la documentación existente para recopilar datos, conocimientos, ideas y opiniones de grupos para proponer o establecer relaciones entre las características de los sujetos, lugares, hechos y situaciones.

RESULTADOS Y DISCUSIÓN

Desarrollo del software basado en herramientas de generación de código

El auge del desarrollo de software ha estado favorecido, en gran medida, por la utilización de distintas herramientas que generan, de alguna manera, gran parte del código fuente de la aplicación. Entre estas herramientas, se encuentran los Entornos de Desarrollo Integrado (IDE, por sus siglas en inglés: Integrated Development Enviroment) y los Marcos de Trabajo (*Framework*, por sus siglas en inglés). En muchas ocasiones, dichas herramientas, a partir de un modelo de bases de datos, generan el mecanismo de acceso a los datos y varias de las vistas que se utilizan en la aplicación. En otros casos, las herramientas generan menos código, resumiéndose solo a adelantar la declaración de métodos, de clases y de algunos ficheros [10].

En los inicios de la programación, los desarrolladores tenían que generar todo el código fuente necesario para sus aplicaciones. Por tal razón, los costos asociados a su desarrollo eran muy altos, invirtiendo mucho tiempo en la generación del código. En muchas ocasiones, por el hecho de invertir mucho tiempo en la generación del código fuente, se dedicaba muy poco o ningún tiempo al proceso de diseño y aseguramiento de la calidad del software. Actualmente, con el uso de herramientas de generación de código, los desarrolladores invierten la mayor parte del tiempo en diseñar el funcionamiento de sus aplicaciones.

Una de las principales ventajas de los IDEs es la construcción de la Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés: Graphical User Interface), con la implementación de muy poco código fuente (a base de $click_s$). Este aspecto disminuye considerablemente el tiempo de desarrollo de la aplicación; pues prácticamente los desarrolladores solo deben concentrase en la implementación de la lógica de la aplicación.

Otro aspecto a analizar en los IDEs, es el alto grado de reutilización que permiten en las aplicaciones construidas. Este elemento es de suma importancia, porque en muchas ocasiones los desarrolladores utilizan *piezas* de otras aplicaciones, disminuyendo considerablemente el tiempo de desarrollo.

El concepto marco de trabajo, se define en el ámbito del desarrollo de software, como una estructura conceptual y tecnológica de soporte definida, con artefactos o módulos que sirven de organización y desarrollo en la aplicación. Estas soluciones se desarrollan utilizando patrones de diseño y tienen como características fundamentales la alta cohesión y el bajo acoplamiento.

Para lograr su propósito, los marcos de trabajos tienen desarrolladas piezas u objetos, que vinculan las necesidades del sistema con la funcionalidad de los marcos. La mayoría de los *frameworks* actuales implementan el patrón modelo-vista-controlador (MVC); donde los modelos permiten el acceso a datos, las vistas permiten la comunicación con los usuarios y los controladores, la comunicación entre modelos y vistas. Este patrón favorece en gran medida a los desarrolladores.

Actualmente, los marcos de trabajo llegan a incluir extensiones específicas de la industria, estándares de negocio y tecnología. Estos *frameworks* permiten comenzar el proyecto sobre una sólida arquitectura de aplicación definida, implementada y probada.

Métodos de estimación del esfuerzo

Para Roger Pressman [7], el tamaño del proyecto es uno de los factores importantes que pueden afectar la precisión y la eficacia de las estimaciones. A juicio de estos autores, existe una relación directa entre el tamaño y la interdependencia entre los diferentes elementos del software y la descomposición del problema, un enfoque importante para la estimación, se vuelve más difícil. La disponibilidad de información histórica, consideran que tiene una gran influencia en el riesgo de planificación.

Los métodos de estimación permiten realizar una predicción del costo en esfuerzo y tiempo de un proyecto informático. Estos métodos se agrupan en dos categorías principales: métodos paramétricos y métodos heurísticos. En el primer caso, se encuentran aquellos métodos donde se llega al resultado a través de la utilización de fórmulas o modelos matemáticos. Por su parte, en los métodos heurísticos, se logra la estimación siguiendo el juicio u opinión de un experto. Cada día que pasa aumenta el uso de los métodos heurísticos, lo cual no quiere decir que los paramétricos se encuentren en desuso; todo lo contrario [7, 8 ,11].

En [12], Barry Boehm introduce una jerarquía de modelos de estimación de software, con el nombre de COCOMO (Constructive Cost Model). Entre los modelos más utilizados, dentro de la jerarquía COCOMO, se encuentran: Puntos de Función, Puntos de Casos de Uso y Puntos de Objetos [13 -16]. Los dos primeros modelos corresponden a la jerarquía COCOMO I; mientras que el último mencionado, corresponde a la jerarquía COCOMO II. De los tres métodos mencionados, Puntos de Objetos es el que contempla, en su análisis, la influencia de las herramientas de generación de código automático.

El método de Puntos de Función estima el esfuerzo y tiempo de desarrollo a partir del tamaño de la aplicación desde el punto de vista funcional o del usuario, o sea, mide el tamaño lógico o funcional de una aplicación o proyecto de software basado en los requerimientos funcionales del usuario [6, 11, 17].

El método de Puntos de Casos de Uso se basa en el método de Puntos de Función y tiene como objetivo estimar el tiempo y el esfuerzo necesario para desarrollar un conjunto de casos de uso [17,18].

Los Puntos de Objeto se han propuesto debido a las prácticas utilizadas en la construcción de aplicaciones a partir de componentes; donde se utilizan ayudas en el desarrollo de interfaces y la reutilización de los componentes [19]. El procedimiento de cálculo es similar al de Puntos de Función. Este método toma en consideración un conjunto de indicadores que tienen un peso que puede favorecer o no el esfuerzo estimado. Pero, a diferencia de otros métodos, da relevancia a la reutilización, por lo que en alguna medida se podría valorar las facilidades que ofrecen las herramientas de generación de código.

Estimación de las líneas de código generadas, nuevas y modificadas

Entre las herramientas y técnicas que se proponen en Project Management Body of Knowledge (PMBOK) [20] para gestionar la finalización oportuna del proyecto, está el análisis de datos. La posibilidad de contar con información sobre el porcentaje de líneas de código fuente que son generadas por el *framework* y no requieren de modificaciones de los desarrolladores, puede ser un elemento a tomar en consideración para estimar la duración de las actividades.

Los proyectos informáticos pueden ser clasificados en función de muchos criterios diferentes. La complejidad del proyecto es un elemento fundamental antes de realizar la estimación de tiempo y esfuerzo del mismo. Atendiendo a la complejidad de un proyecto, una de los criterios más confiables permite situar un sistema en una de las siguientes tres categorías [12]:

- Orgánico o de baja complejidad: Son proyectos con menos 50 000 líneas de código. Cuentan con un área de conocimiento bien definida y sus requisitos no son rígidos (pueden ser cambiados siempre que el equipo demuestre que es favorable el cambio).

- Semiempotrado, semiacoplado, semilibre o de media complejidad: Son proyectos con mayor complejidad que los orgánicos. Cuentan con menos de 300 000 líneas de código, pero más de 50 000. Sus requisitos no son rígidos, pero cambiarlos es complicado y/o peligroso. El área de negocio no está tan bien definida como en proyectos de baja complejidad.
- Empotrados, rígidos o de alta complejidad: Son los proyectos que cuentan con mayor grado de complejidad. La bibliografía sobre el área de negocio asociada a los proyectos de esta clase es poca. Pueden contar con cualquier cantidad de líneas de código, pero sus requisitos son rígidos (sin posibilidad de cambiarlos).

Como este análisis se realiza una vez concluido el proyecto, se clasifica según el número de líneas de código.

Para obtener datos que permitan evaluar el impacto de la utilización de un *framework* en el tamaño de un proyecto estimado por las líneas de código fuente de la aplicación informática, se construyó un software que permite estimar la cantidad de líneas de código generadas por el *framework*, las nuevas escritas por el equipo de desarrollo y las generadas por el *framework* que requirieron ser modificadas por los programadores.

Este software para estimar necesita comparar la aplicación objeto de análisis con un proyecto base. Para entender cómo funciona, en la figura 1 se muestra el diagrama de actividades que describe el flujo principal y en las figuras 2 y 3 los detalles de las subactividades más complejas.

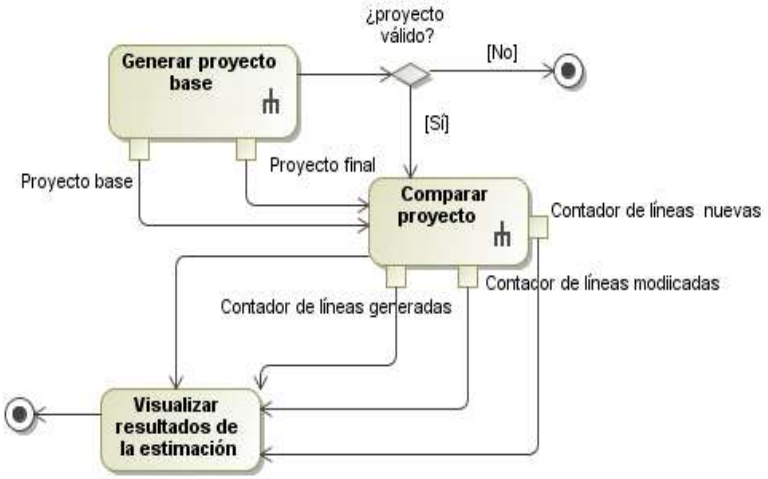

Fig. 1. Flujo para la estimación de las líneas de código

Como se aprecia en la figura 2, se requiere crear un proyecto base para la comparación. Por tal motivo, hay que especificar la ubicación del script de la base de datos y el código fuente, y el sistema tiene que identificar el gestor de base de datos y el *framework* utilizado en el desarrollo.

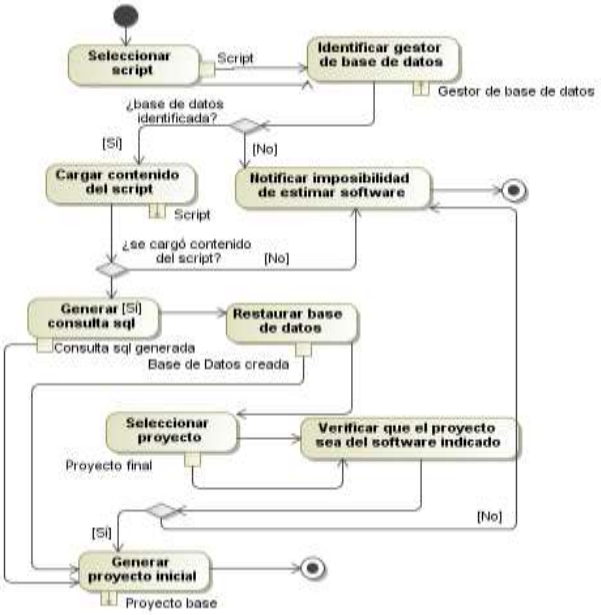

Fig. 2. Flujo para la generación del proyecto base

La generación del proyecto base, está contenida en un mecanismo de diseño (generar código fuente) que toma como base las tablas de la base de datos del proyecto a analizar. Para cada una de las tablas, se invoca a las funciones del *framework* que generan el código de un modelo y el código que responde a la creación, recuperación, actualización y eliminación de instancias.

Como resultado de la comparación de la aplicación objeto de análisis (proyecto seleccionado) con respecto al proyecto creado, se obtienen los contadores de las cantidades de líneas de código fuente nuevas, modificadas una vez generadas y generadas sin modificar posteriormente (figura 3).

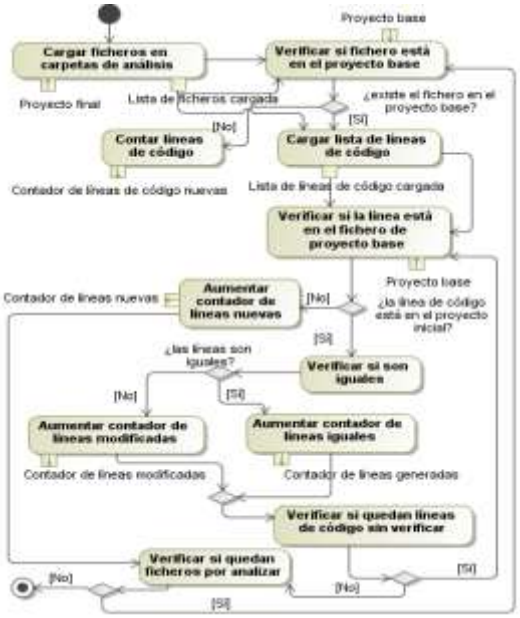

Fig. 3. Flujo para la comparación del proyecto

En la visualización se muestran los resultados cuantificados en los contadores.

En la figura 4 se muestra el código que implementa la comparación entre los proyectos, al cual se le aplicaron pruebas no funcionales o de caja blanca [7,21-22]. En particular, la técnica de cobertura de caminos, obteniéndose el grafo que se presenta en la figura 5.

```
public function CompareTwoFolders($finalFolder, $originalFolder){
    $GLOBALS['new_lines'] = 0;
    $GLOBALS['modified_lines'] = 0;
    $GLOBALS['total_lines'] = 0;

    $result = array();
    if (!is_dir($finalFolder)){
        array_push($result, ['problem'=>$finalFolder.' no es un directorio válido']);
    }
    elseif (!is_dir($originalFolder)){
        array_push($result, ['problem'=>$originalFolder.' no es un directorio válido']);
    }
    else{
        $baseDir = "destDir";
        $result = $this->svncompare($finalFolder, $originalFolder, $baseDir);
    }
    return $result;
}
```

Fig. 4. Método encargado de comparar dos proyectos

Como se puede observar, la comparación se basa en la utilización del componente *svncompare*.

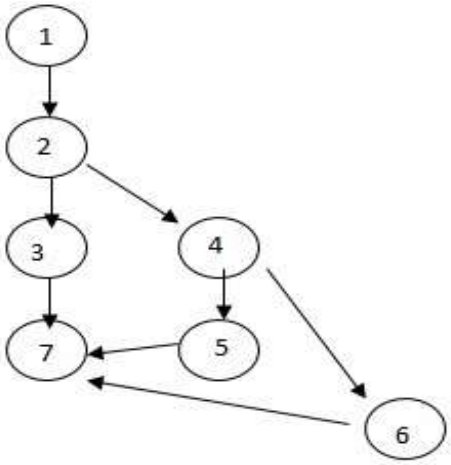

Fig. 5. Grafo asociado al método encargado de comparar dos proyectos

Los casos de prueba cubren todos los caminos y prueban los cursos 1-2-4-6-7, 1-2-3-7 y 1-2-4-5-7; obteniéndose los resultados esperados.

Con el objetivo de evaluar la viabilidad de la herramienta, se realizaron cinco experimentos donde se evaluaron en cada uno un proyecto desarrollado con el *framework* Yii 2.0. De acuerdo con la cantidad de líneas de código fuente, las aplicaciones tienen diferentes niveles de complejidad. En la tabla 1 se presenta la cantidad total de líneas de código fuente y en la figura 6 se gráfica el porcentaje de estas que fueron generadas por el *framework* y no sufrieron modificaciones (generadas), las generadas por el *framework* que el equipo de desarrollo cambió (modificadas) y las implementadas por los desarrolladores (nuevas).

Tabla 1. Cantidad de líneas de código de los proyectos comparados

Proyecto	Total de líneas de código
1	474 938
2	891 537
3	1 744 340
4	2 175 952
5	3 217 105

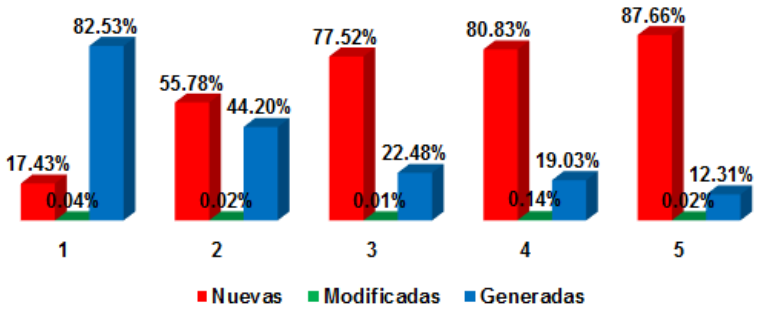

Fig. 6. Porcentaje de líneas de código generadas, nuevas y modificadas

Una vez analizados estos resultados, se puede elaborar una hipótesis asociada a la complejidad del proyecto y a la generación de código automático. Mientras mayor complejidad tenga el proyecto, mayor posibilidad de que el equipo de desarrollo implemente las funcionalidades requeridas auxiliándose menos del *framework*. Sin embargo, aplicaciones menos complejas, se pueden obtener en un porcentaje elevado, apoyándose en el *framework*. Su demostración requiere extender la aplicación de la herramienta desarrollada para disponer de información suficiente que la corrobore o no.

Por otra parte, resulta evidente que los desarrolladores cuando evalúan lo que genera el *framework*, lo asumen sin realizar modificaciones prácticamente.

CONCLUSIONES

El proceso de estimación de tiempo y esfuerzo de un proyecto informático actual se realiza de dos posibles maneras: siguiendo el criterio de expertos en el tema o basando la estimación en modelos matemáticos, conocidos como métodos de estimación.

A pesar del avance de la tecnología y de los logros en el campo de la calidad de software aún persisten debilidades a la hora de estimar el tiempo y los recursos requeridos en un proyecto de software.

La mayoría de los métodos de estimación no contemplan la influencia de las herramientas de desarrollo en la generación de código de forma automática; pudiendo dar valores alejados a la realidad.

Actualmente, en muchas ocasiones, la fase de desarrollo es menor que la de planeación y de diseño. Este aspecto se ve influenciado por las herramientas de trabajo, las cuales generan, de forma automática, gran parte del código de la solución.

La herramienta desarrolla una base para modificar el análisis de los métodos de estimación que no contemplan el porcentaje de generación de código automático por las herramientas de desarrollo. Esta modificación podría ser posible mediante el uso de dicha herramienta, analizando los resultados obtenidos en un grupo grande de proyectos.

REFERENCIAS

1. Dapazo G, Greiner C, Ferraro M, Medina Y, Pedrozo G, Lencina B. Medición y estimación del software: métodos y herramientas para mejorar la calidad del software. En WICC 2014 XVI Workshop de Investigadores en Ciencias de la Computación, Red de Universidades con Carreras en Informática (RedUNCI), Argentina, 2014. pp. 575-579. Disponible en: http://sedici.unlp.edu.ar/handle/10915/41605 [consultado en febrero 2018].
2. Blas MJ, Gonnet S, Leone, H. Definición de un esquema de calidad basado en el estándar ISO/IEC 25010. En ASSE 2016 17 Simposio Argentino de Ingeniería de Software, Santa Fé, Argentina, 2016. pp. 135-146. Disponible en: http://sedici.unlp.edu.ar/handle/10915/57158 [consultado en febrero 2018].
3. Almache MG, Raura G, Ruiz J, Fonseca E. Modelo neuronal de estimación para el esfuerzo de desarrollo en proyectos de software (MONEPS); Revista Latinoamericana de Ingeniería de Software, 2015. 3(3):148-154. ISSN 2314-2642.
4. Ruiz Constanten Y, Cordero Morales D. Estimación en proyectos de software integrando los métodos de Boehm y Humphrey. Revista Cubana de Ciencias Informáticas, 2013. 7(3):23-26. ISSN 2227-1899.
5. Alnajjar M, Abu Nase SS. Evaluating software engineering practices in Palestine. International Journal of Soft Computing, Mathematics and Control, 2014. (1):35-47. ISSN 2201-4160.
6. Dapozo G, Ferraro M, Medina Y, Pedrozo G, Lencina B. Análisis comparativo de métodos de estimación basados en puntos de función para proyectos web. En XX Congreso Argentino de Ciencias de la Computación. Buenos Aires, Argentina, 2014. Disponible en: http://sedici.unlp.edu.ar/handle/10915/42363 [consultado en febrero 2018].
7. Pressman R, Maxim B. Software Engineering: a practitioner´s approach. Eighth Edition. 2015, McGraw-Hill Education, pp. 977. ISBN 978-0-07-802212-8.
8. Lencina B, Medina Y. Dapozo G. Aplicación para estimar costos en proyectos de software. En ASSE 2016, 17 Simposio Argentino de Ingeniería de Software, Santa Fé, Argentina, 2016. pp. 181-192. Disponible en: http://sedici.unlp.edu.ar/handle/10915/57248 [consultado en febrero 2018].
9. Santana Ballate D, Artega Céspedes F. Evaluación del impacto del uso del framework yii2 en la estimación del esfuerzo de desarrollo de software. Director: Vanessa Muñoz Castillo y Anaisa Hernández González. Tesis de Ingeniería, Universidad Tecnológica de La Habana José Antonio Echeverría Cujae, La Habana, Cuba. 2018. Disponible en: http://tesis.cujae.edu.cu:8080/handle/123456789/8733 [consultado en octubre 2018].
10. Sommerville I. Software Engineering. Ninth Edition, 2011. Pearson Education, Inc., publishing as Addison-Wesley, p. 790. ISBN 978-0-13-703515-1.
11. Pedrozo G, Greiner C. Herramienta para apoyar la estimación en el desarrollo de aplicaciones web. En XIX Concurso de trabajos estudiantiles (EST2016)-JAIIO 45, Argentica, 2016. pp. 116-130). Disponible en: http://sedici.unlp.edu.ar/handle/10915/58188 [consultado en febrero 2018].
12. Boehm B . Software Engineering Economics. IEEE transactions on Software Engineering. 1984, (1): 4-21. ISSN 0098-5589.
13. Antúnez T, Valdovinos R, Marcial J, Ramos M, Herrera E. Estimación de costos de desarrollo, caso de estudio: Sistema de Gestión de Calidad del Reactor TRIGA Mark III. Revista Cubana de Ciencias Informáticas, 2016. 10(1):215-228. ISSN 2227-1899.
14. Shida T, Tsuda K. A Study of software estimation factos extracted using covariance structur analysis. Procedia Computer Science, 2017. (112):1378-1387. ISSN 1877-0509.
15. Silhavy R, Silhavy P, Prokopova Z. Evaluating subset selection methods for use case points estimation. Information and Software Technology, 2018. 97:1-9. ISSN 0950-5849.
16. Dewi RS, Subriadi, AP. A modification complexity factor in function points method for software cost estimation towards public service application. Procedia Computer Science, 2017. 124:425-422. ISSN 1877-0509.
17. Barahona Rodríguez CY, Arias Rojas DS, Chía Rodríguez PA. Técnica híbrida de estimación basada *en el análisis de puntos de función y puntos de casos* de uso. En 4to Congreso Internacional AmITIC 2017, Popayán, Colombia 2017, pp. 102-109. Disponible en: revistas.utp.ac.pa/index.php/memoutp/article/download/1477/2124 [consultado en febrero 2018].

18. Remón C, Thomas PJ. Análisis de estimación de esfuerzo aplicando puntos de caso de uso. En XVI Congreso Argentino de Ciencas de la Computación CACIC 2010, Argentina, 2010, pp. 577-586. Disponible en: http://sedici.unlp.edu.ar/handle/10915/19290 [consultado en febrero 2018].
19. Dolado Cosín JJ. Medición de especificaciones de software. Novática, 1999. 137:17-19. ISSN 0211-2124.
20. Project Management Institute: A guide to the Project Management Body of Knowledge (PMBOK®GUIDE). Sixth Edition. 2017, Project Management Institute, p. 756. ISBN 978-1-62-825382-5.
21. Spillner A, Linz T, Schaefer H. Software Testing Foundations: a study guide for the certified tester exam. Four Edition. 2014, Rocky Nook, p. 304. ISBN 978-1-93-753842-2.
22. Barrientos P. Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. Director: Claudia Pons. Tesis de maestría. Universidad Nacional de La Plata, Argentina. 2014. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/34969/Documento_completo__.pdf?