# ESTIMACIÓN TEMPRANA DEL TIEMPO DE DESARROLLO EN PROYECTOS DE SOFTWARE UTILIZANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Yaimara Granados Hondares, Gheisa Lucía Ferreira Lorenzo,

Jarvin Antón Vargas, Luis Quintero Domínguez

Universidad Central "Marta Abreu" de Las Villas

ygranados@uclv.edu.cu; gheisa@uclv.edu.cu;

janton@uclv.edu.cu; lqdominguez@uclv.edu.cu

## Resumen

La estimación de costos, esfuerzo y tiempo de desarrollo en proyectos de software, ha sido ampliamente estudiada y se han propuesto múltiples métodos que van desde aquellos basados en ecuaciones y modelos de estimación como COCOMO, hasta otros que en los últimos años han incorporado técnicas de Inteligencia Artificial.

Este trabajo trata el tema de la estimación temporal en tareas de implementación utilizando como marco de trabajo el Redmine, una aplicación web para administrar proyectos de software. Se logra la programación en Ruby on rails de un plug-in que captura los datos de variables de estimación previamente seleccionadas. Esta captura de información es utilizada posteriormente como entrada a algoritmos del WEKA que evalúan los resultados de la clasificación, ofreciéndose un análisis comparativo de los mismos. Finalmente se realiza un proceso de selección de rasgos para llegar a resultados más favorables que permiten realizar una propuesta de implementación sobre Redmine de un algoritmo para la estimación del tiempo de desarrollo que pueda ser utilizado en la gestión de proyectos de software.

## Palabras clave

Estimación temporal, variables de estimación, gestión de proyectos de software.

## *Abstract*

*The estimation of time in project development (including the cost and effort) has been widely studied and many methods have been proposed ranging from those based on equations estimation models like COCOMO, to others that in recent years have incorporated Artificial Intelligence techniques.*

*This paper addresses the issue of temporary estimate deployment tasks using as a framework for the Redmine, a web application to manage software projects. Is achieved in Ruby programming of a plug-in that captures the data of variables estimation previously selected. This data capture is used as input to WEKA algorithms that evaluate the classification results, offering a comparative analysis of them. We carried out a feature selection process to achieve more favourable results than a proposal to*

*allow Redmine implementation on an algorithm for estimating the development time that can be used in managing software projects.*

## Key words

*Time estimation, variables estimation, managing software projects.*

## 1. Introducción

Dentro de la disciplina de Gestión de Proyectos, la planificación es una de las etapas más importantes en el desarrollo de cualquier proyecto de software, comprende actividades como *ámbito de software*, *recursos* y *estimación*.

La estimación es punto clave para lograr una buena planificación, su objetivo es predecir las variables involucradas en el proyecto con determinado grado de certeza, tratando así de aportar una predicción de algún indicador importante para la gestión de proyectos de software.

En muchas ocasiones las fechas de entrega y los plazos fijados para darle seguimiento al proyecto se hace de forma arbitraria, o por premura del cliente o por comodidad del equipo de desarrollo para tener más tiempo de trabajo y lograr así un producto de calidad. Ambos aspectos solo pueden acarrear consecuencias negativas, incluso el fracaso del proyecto, si no se atienden ambas necesidades, cliente y desarrollador, y se realiza una predicción de forma acertada de la duración del proyecto. De ahí que la estimación sea importante no solo para predecir el valor de variables concretas dentro de un proyecto, sino para determinar su **viabilidad;** no tiene sentido iniciar un proyecto que está destinado al fracaso por no contar con el tiempo necesario para llevarlo a cabo.

La existencia de técnicas de estimación ha erradicado en gran medida estas dificultades. Puntos de Función, COCOMO, Casos de Uso, por mencionar algunas, basan su funcionamiento en la estimación de costos, esfuerzo y duración de proyectos de software, lo que posibilita contar, en el inicio del ciclo de vida del proyecto, con datos que garantizarán en gran medida el éxito.

En los últimos años la tendencia a buscar nuevas formas de estimar proyectos de software ha marcado un camino hacia la utilización de algoritmos inteligentes.

A partir de un estudio realizado en el Centro de Desarrollo de Software UCI – Villa Clara, se determinó la necesidad de prestar especial atención a la estimación de tiempo

de tareas de implementación, por ser la etapa de implementación la que requiere mayor esfuerzo y dedicación para lograr un producto de calidad.

Redmine, aplicación Web para la administración de proyectos, es utilizada en el Centro de Desarrollo para gestionar todo lo referente a sus proyectos. Allí se almacena, entre otros datos importantes, el conjunto de tareas dedicadas a la implementación. Una necesidad actual del centro es estimar el tiempo de duración de cada una de esas tareas, para poder controlar la ejecución de las mismas, pero el RedMine no incluye entre sus funcionalidades la estimación, por lo que es factible realizar adiciones que permitan lograr este propósito.

Objetivo:

Estimar el tiempo de duración de tareas de implementación sobre la aplicación web para administrar proyectos, Redmine.

## 2. Técnicas de estimación

### *Puntos de función*

Los Puntos de Función constituyen una métrica que permite traducir en un número, el tamaño de la funcionalidad entregada al usuario, independientemente de la tecnología utilizada para la construcción y explotación del software, a través de una suma ponderada de las características del producto.

Este método calcula los puntos de función de un sistema descomponiendo al mismo en cinco funciones principales (entradas, salidas, consultas, ficheros internos y externos), asignándoles valores de acuerdo a su complejidad y en función de la cantidad de cada uno de ellos y de determinados factores de ajuste de la complejidad se llega a determinar el tamaño del software.

El método de Puntos de Función presenta un aspecto crítico en lo referido a la reutilización de módulos preexistentes (por ejemplo varias salidas similares que poseen una misma estructura con variaciones propias de cada una de ellas). Además carece de precisión cuando se trata de proyectos pequeños, por debajo de 100 puntos de función resulta poco fiable. [2]

### *COCOMO*

Este método de estimación ha evolucionado desde COCOMO 81 hasta COCOMO II atendiendo a la evolución también de las metodologías de desarrollo. Esta técnica requiere de un dato elemental determinado por la cantidad de sentencias de código del proyecto a la que posteriormente se aplican diferentes algoritmos. Fue desarrollado por Barry W. Boehm [1].

COCOMO basa la estimación en la cantidad de líneas de código (LOC) lo cual debilita la certeza de la técnica pues es sumamente difícil predecir la cantidad de líneas de código a utilizar en etapas tempranas del desarrollo de software.

***Puntos de Casos de Uso***

Por su parte la estimación por puntos de casos de uso, aunque fue pensada a partir de los puntos de función, utiliza los actores y casos de uso para calcular el esfuerzo que significará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan factores de entorno y de complejidad técnica para ajustar el resultado. [3]

## 3. Inteligencia artificial y la estimación en proyectos de software

Actualmente el estudio de las técnicas de estimación se ha centrado con fuerza en la utilización de técnicas de inteligencia artificial con el fin de lograr una estimación más confiable y exacta.

Para lograr este propósito, y utilizando las facilidades del Redmine, se diseñó e implementó un plug-in que fuera capaz de capturar información adicional de las tareas de implementación, que sería considerada en la estimación.

***PLUG-IN para el REDMINE***

El plug-in para el REDMINE fue programado en Ruby on Rails, permite realizar tres funciones principales:

- Visualización de las tareas: Muestra las características de las tareas con los valores que poseen hasta el momento las variables. Las mismas serán tomadas

en cuenta para la estimación del tiempo que demora el desarrollador en realizar la tarea.

- Edición de la información de una tarea: Cuando se visualiza la información de una tarea, también se brinda la posibilidad de editar la información contenida en ella. Cuando se selecciona Editar la información de una tarea, se obtiene una ventana que indica cada uno de los parámetros, con su valor actual (NS/ND en caso de que no se haya definido) y permite cambiar su valor seleccionándolo desde una lista de valores predefinida. Luego de actualizados los datos, estos deben ser guardados.

- Exportar la información a formato .ARFF para poder ser analizados por algoritmos del repositorio WEKA.

- Estimar el tiempo de duración de una tarea de implementación.

Con la utilización del plug-in se recopiló información sobre proyectos de software reales en el Centro de Desarrollo de Software UCI – Villa Clara y se definió una base de casos con 92 ejemplos. Cada caso se caracteriza por 20 rasgos, uno de los cuales se refiere al rasgo objetivo (clase), Tiempo Real.

**Definición de las variables de estimación**

Para la recopilación de datos que posteriormente serán usados para la selección del algoritmo de estimación de tiempo, se definieron 20 variables [5], consideradas como las que tienen el mayor peso para la estimación del tiempo de duración de una tarea de implementación. Se define un dominio para cada una de las variables, que después se considera en la definición de los datos para el WEKA [7].

**Algoritmos seleccionados para la clasificación**

A partir de la base de casos, que almacena cada ejemplo como un par atributo-valor, se realiza la selección del algoritmo de clasificación. Para ello se utiliza la herramienta de aprendizaje automático WEKA, la cual tiene implementados estos algoritmos; así como facilidades para una evaluación comparativa entre varios modelos.

Se seleccionaron 4 algoritmos del WEKA (LinearRegresion, LeastMedSq, MLP, M5rules) para realizar la clasificación de la muestra, obteniendo los resultados que aparecen en la Tabla 1.

| | **Regresión lineal** | **Mínimos cuadrados** | **MLP** | **M5Rules** |
|---|---|---|---|---|
| **Correlation coefficient** | 0.1498 | 0.4471 | 0.615 | 0.7497 |
| **Mean absolute error** | 16.5979 | 10.2237 | 8.6159 | 5.9233 |
| **Root mean squared error** | 48.9536 | 20.9893 | 12.8757 | 9.0892 |
| **Relative absolute error** | 156.9421 % | 96.6704 % | 81.4682 % | 56.0081 % |
| **Root relative squared error** | 371.3186 % | 159.2061 % | 97.6637 % | 68.9424 % |
| **Total Number of Instances** | 92 | 92 | 92 | 92 |

Tabla 1: Resumen de los Resultados de la Primera Clasificación

Como se puede observar, de todos los algoritmos utilizados, el M5Rules es el que tuvo un mayor coeficiente de correlación y un menor error absoluto, por lo que se escogió este algoritmo para realizar la estimación de tiempo.

Para eliminar variables que pueden enturbiar la corrida del algoritmo de clasificación se realizó la selección de rasgos, con el WEKA, para determinar las variables que en realidad influyen en la corrida del M5Rules. Para esto se utiliza el evaluador de atributos WrapperSubsetEval ya que permite seleccionar para qué algoritmo de clasificación se está realizando la selección de rasgos (en este caso para el M5Rules) y se emplean 2 métodos de búsqueda (BestFirst y GeneticSearch). Los resultados obtenidos se muestran en la Tabla 2.

| | **WrapperSubsetEval con BestFirst(6)** | **WrapperSubsetEval con GeneticSearch(10)** |
|---|---|---|
| Variables | conReuso<br>eficiencia<br>fiabilidad<br>tiempoEstimado<br>herramientasSoftware<br>tiempoReal | complejidad<br>conocimientoLenguaje<br>eficiencia<br>tiempo<br>estimado<br>herramientasSoftware<br>variables<br>facilidadPrueba<br>integridad<br>objetivosRendimiento<br>tiempoReal |

Tabla 2: Resultados de la selección de rasgos.

Posteriormente se realizó la corrida del M5Rules utilizando solo las variables que se obtuvieron de la selección de rasgos. La Tabla 3 muestra los resultados obtenidos.

| | M5Rules (con las variables que dio el método BestFirst) | M5Rules (con las variables que dio el método GeneticSearch) |
|---|---|---|
| **Correlation coefficient** | 0.8328 | 0.8517 |
| **Root mean squared error** | 7.2469 | 6.8857 |
| **Relative absolute error** | 46.3019 % | 41.773 % |
| **Root relative squared error** | 54.9687 % | 52.2291 % |
| **Total Number of Instances** | 92 | 92 |

Tabla 3: Resumen de los Resultados de la Clasificación luego de la selección de rasgos.

Tras los resultados arrojados se decidió tomar, para la estimación del tiempo de las tareas de implementación, las dos reglas generadas a partir de las variables obtenidas con el método GeneticSearch. Estas reglas fueron implementadas, brindando así la funcionalidad al RedMine, de estimar. Algunos resultados de prueba se muestran en la Figura 1.


Figura1: Plug-in para la gestión de tareas de implementación.

## Conclusiones

Como resultado de este trabajo se implementó un plug-in para el Redmine que permite gestionar la información de las tareas asociadas con los valores de las variables de estimación seleccionadas, así como exportar al formato .arff para el posterior análisis en el WEKA.

A partir de los datos de entrada al WEKA se realizó un análisis que permitió la generación de reglas para estimar la duración de las tareas.

Actualmente se trabaja en la instalación del plug-in en el Centro de Desarrollo de Software UCI-Villa Clara, para aumentar la muestra de casos (92 hasta hoy) con el objetivo de estudiar los resultados.

## Referencias bibliográficas

[1] Boehm, Barry W. Software Engineering Economics. s.l. : Prentice Hall, 1981.

[2] International Function Point users Group. Function Point Counting Practices Manual. 1994, Vol. Release 4.0.

[3] Project estimation with Use Case Points. Clemmons, Roy K. s.l. : Crosstalk: The Journal of Defense Software Engineering., 2006.

[4] Pressman, R.S. Ingeniería del Software. Un enfoque práctico. s.l. : Mc Grow Hill, 1994.

[5] Velarde, Héctor J. Diploma de Estudios Avanzados. Departamento de Ciencias e Ingenierías Físicas y Formales. Arequipa : s.n., 2010.

[6] Frank, Ian H. Witten and Eibe. Data Mining: Practical machine learning tools with Java implementations. San Francisco : Morgan Kaufmann, 2000.