ISSN 2007-9737

MUREM: Un método multiplicativo de regresión para estimar el esfuerzo de desarrollo de software

Ma. del Refugio Ofelia Luna Sandoval, José Ruiz Ascencio

CENIDET, Departamento de Ciencias Computacionales, Cuernavaca, México

{oluna, josera}@cenidet.edu.mx

Resumen. En este artículo se presenta un método multiplicativo de regresión para estimar el esfuerzo de desarrollo de software. Este método, al que denominamos MUREM, es el resultado de, por un lado, establecer un conjunto de condiciones iniciales para enmarcar el proceso de estimación del esfuerzo de desarrollo de software y de, por otro lado, estipular las propiedades que, racionalmente, debe satisfacer la relación que se establece entre el esfuerzo de desarrollo y el tamaño del software. Para evaluar el desempeño de MUREM éste se comparó con tres modelos de regresión los cuales se consideran como métodos importantes para estimar el esfuerzo de desarrollo de software. En esta comparación se aplicó una batería de hipótesis y pruebas estadísticas a doce muestras extraídas de bases de datos públicas bien conocidas. Estas bases de datos sirven como punto de referencia para la comparación de métodos para estimar el esfuerzo de desarrollo de software. En la experimentación se encontró que MUREM genera estimaciones puntuales del esfuerzo de desarrollo más precisas que aquellas obtenidas por los otros métodos. MUREM corrige la heterocedasticidad y aumenta la proporción de muestras cuyos residuales presentan normalidad. Con esto MUREM genera intervalos de confianza y de predicción más adecuados que aquellos obtenidos por los otros métodos. Un resultado importante es que los residuales obtenidos por el modelo de regresión de MUREM satisfacen la prueba de ruido blanco gaussiano de media cero, con lo que se prueba que el error de estimación de dicho modelo es aleatorio.

Palabras clave. Estimación del esfuerzo de desarrollo de software, método para estimar el esfuerzo de desarrollo de software, método de estimación, método multiplicativo, modelo de regresión.

MUREM: A Multiplicative Regression Method for Software Development Effort Estimation

Abstract. In this paper a multiplicative regression method to estimate software development effort is presented. This method, which we call MUREM, is a result of, on the one hand, a set of initial conditions to frame the process of estimating software development effort and, on the other hand, a set of restrictions to be satisfied by the development effort as a function of software size. To evaluate the performance of MUREM, it was compared with three regression models which are considered as important methods for estimating software development effort. In this comparison a battery of hypothesis and standard statistical tests is applied to twelve samples taken from well-known public databases. These databases serve as benchmarks for comparing methods to estimate the software development effort. In the experimentation it was found that MUREM generates more accurate point estimates of the development effort than those achieved by the other methods. MUREM corrects the heteroscedasticity and increases the proportion of samples whose residuals show normality. MUREM thus generates more appropriate confidence and prediction intervals than those obtained by the other methods. An important result is that residuals obtained by the regression model of MUREM satisfy the test for zero mean additive white Gaussian noise which is proof that the estimation error of this model is random.

Keywords. Software development effort estimation, method for estimating software development effort, estimating method, multiplicative method, regression model.

1. Introducción

En la práctica actual los métodos formales de estimación del esfuerzo de desarrollo apoyan sus estimaciones en métodos basados en modelos empíricos de estimación. De acuerdo con Abdel *et al.*, un modelo empírico es aquel que usa los datos de proyectos previos para evaluar al proyecto nuevo y deriva fórmulas básicas a partir del análisis de las bases de proyectos históricos [1]. En contra parte, un modelo teórico es aquel que usa fórmulas basadas en supuestos generales [1]. Los autores de este artículo creamos un método de estimación del esfuerzo de desarrollo de software, al que denominamos como MUREM. MUREM, considerando el método científico en vigor, se basa en un modelo híbrido (teórico-empírico) de estimación. Esto es que, al principio, este método se formuló con supuestos generales, como se hace con un modelo teórico. Después, como se hace con un modelo empírico, en la elección de su fórmula básica se consideró el análisis de las bases de proyectos históricos que conforman este caso de estudio. Este caso de estudio está conformado por las bases de datos públicas de proyectos históricos de software: Albrecht, China, Desharnais, Finnish, Kemerer, Kitchenham, Maxwell, Mermaid, Miyasaki1 y NASA93. Obtenidas, en su mayoría, del Repositorio de Ingeniería de Software PROMISE (*PRedictOr Models In Software Engineering*). MUREM da buenos resultados (en términos de precisión) en este caso de estudio, cuando se compara con los métodos de regresión siguientes: 1) método de regresión lineal simple (OLS: *Ordinary Least Squares*), 2) método de regresión potencial (RP) de ajuste por mínimos cuadrados y 3) una red neuronal (RN), entrenada con retro propagación de alimentación hacia adelante (*feed-forward backpropagation network*). Estos métodos de regresión han generado, a través del tiempo, importantes modelos empíricos de estimación del esfuerzo de desarrollo de software [2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

El aspecto teórico de MUREM se fundamenta en los dos elementos siguientes: 1) elegir un modelo parsimonioso para realizar la estimación. Un modelo parsimonioso es aquel que usa el menor número de variables de entrada y de parámetros para realizar una representación adecuada de los datos [12] y, 2) una "ley *a priori*" que considera en su aplicación un conjunto de condiciones iniciales.

De acuerdo con los trabajos de [13, 14], la variable más significativa para predecir el esfuerzo de desarrollo es el tamaño del software. Por esta razón sólo se consideró esta variable para explicar la variabilidad del esfuerzo de desarrollo.

Nuestra ley *a priori* estipula las propiedades que, racionalmente, debe satisfacer la relación que se establece entre el tamaño del software y el esfuerzo de desarrollo. Esta ley considera en su aplicación un conjunto de condiciones iniciales, al que denominamos como "marco de condiciones iniciales". Este marco se define como el conjunto de circunstancias en las que se realiza el proceso de estimación del esfuerzo de desarrollo. Estas circunstancias dependen de las circunstancias en las que se está realizando el proceso de desarrollo del software nuevo.

Las circunstancias en las que se desarrolla un software dependen tanto de los factores humanos como de los tecnológicos [15]. Los autores de este artículo proponemos que se tomen en consideración aquellas condiciones básicas relacionadas con los principales elementos a considerar durante la administración efectiva de un proyecto de software. Estos elementos, de acuerdo con Pressman, son: el personal, el producto, el proceso y el proyecto [16].

El resto del material de este artículo está organizado como sigue: La sección 2 explica el fundamento teórico de MUREM. La sección 3 describe, brevemente, los métodos de estimación del esfuerzo que se usaron para comparar a MUREM. En la sección 4, se presentan la experimentación y los resultados obtenidos de esta comparación. Finalmente, en la sección 5 se dan las conclusiones y se sugieren los trabajos futuros.

2. Los fundamentos de MUREM

A continuación se describe, brevemente, lo que se entiende por marco de condiciones iniciales del proceso de estimación del esfuerzo y por ley *a priori* para la estimación del esfuerzo.

ISSN 2007-9737

2.1. Marco de condiciones iniciales para la estimación del esfuerzo de desarrollo

Llamamos "marco de condiciones iniciales del proceso de estimación del esfuerzo de desarrollo de software", al conjunto de circunstancias en el que ha de ocurrir el proceso de estimación del esfuerzo de desarrollo para un proyecto nuevo. Las condiciones en las que se desarrolla un software nuevo son diversas y dependen en mucho del giro de la empresa de software que lo está desarrollando. Si el modelo utiliza datos históricos para realizar la estimación, entonces estos datos necesariamente deberán considerar las condiciones en las que se realizó el proceso de desarrollo que los generó. Los autores de este artículo sugerimos que las condiciones que limitan el proceso de desarrollo de software tomen en cuenta los principales elementos a considerar durante la administración eficiente de un proyecto de software. Estos elementos, de acuerdo con Pressman, son: el personal, el producto, el proceso y el proyecto [16]. A continuación ejemplificamos cuáles condiciones iniciales se podrían considerar en cada uno de estos rubros.

a. Condiciones iniciales con respecto al personal.

– Nivel de educación del equipo de desarrollo.
– Experiencia del equipo en el desarrollo de software.
– Experiencia del equipo en el lenguaje de programación.
– Experiencia del equipo en la metodología de desarrollo.
– Experiencia del equipo en el dominio del proyecto.
– Experiencia del equipo en el uso de las herramientas de soporte.

b. Condiciones iniciales con respecto al producto de software.

– Tamaño del producto. Esta condición es muy importante ya que la estimación del esfuerzo está en función de la estimación del tamaño del producto nuevo. La estimación del tamaño se debe realizar tomando en cuenta el mismo estándar que se usó para medir el tamaño del software de los proyectos históricos.
– Tipo de producto. Por ejemplo: software de sistema, software de programación, software de aplicación, etc.
– Lenguaje e IDE de desarrollo.

c. Condiciones iniciales con respecto al proceso de desarrollo de software.

– Metodología de desarrollo (tradicional o ágil).
– Modelo de SDLC (cascada, espiral, por prototipos, una combinación de ellos, etc.).
– Complejidad de la programación.
– Complejidad en los datos.
– Uso de herramientas CASE.

d. Condiciones iniciales con respecto al proyecto de desarrollo de software.

– Tipo de proyecto. Por ejemplo: Administrativo, Bancario, Contabilidad, Educación, Electrónica, Finanzas, Medicina, Telecomunicaciones, Ventas, etc.
– Dominio de la práctica de software. Por ejemplo: proyectos muy grandes, proyectos aeroespaciales, aplicaciones de negocios, etc.

Éstas son sólo algunas de las condiciones básicas que se podrían considerar al momento de iniciar el proceso de desarrollo del software. Sin embargo, es imprescindible que, antes de realizar las estimaciones, cada empresa determine su propio marco de condiciones iniciales. Este marco deberá ser considerado en el criterio que se use para elegir el segmento de la base de proyectos históricos que se utilizará para estimar, tanto el tamaño como el esfuerzo de desarrollo del software nuevo.

Establecer el marco de condiciones iniciales para el proceso de estimación del esfuerzo de desarrollo no es suficiente. Además es necesario que se determinen las propiedades que, racionalmente, debe satisfacer la relación que se establece entre el esfuerzo de desarrollo y el tamaño del software. Estas propiedades se consideran en lo que nosotros denominamos como "ley *a priori* para la estimación del esfuerzo de desarrollo de software", la cual se describe a continuación.

2.2. Ley a priori para la estimación del esfuerzo de desarrollo de software

El proceso de estimación del esfuerzo de desarrollo de software es uno de los procesos del "Ciclo de Vida del Desarrollo de Software" (SDLC: *Software Development Life Cycle*). Un SDLC es un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de sus requerimientos hasta la finalización de su uso [17]. El "Principio de Incertidumbre en la Ingeniería de Software" (UPSE: *Uncertainty Principle in Software Engineering*), formulado en [18], estipula que la incertidumbre es inherente e inevitable en el SDLC y sus productos.

Para considerar la incertidumbre aleatoria inherente al proceso de estimación del esfuerzo de desarrollo de software, representamos a este proceso como un experimento aleatorio que se realiza bajo un mismo marco de condiciones iniciales. Es evidente que aunque este proceso se realice bajo el mismo marco de condiciones los valores de las medidas (o estimaciones) de los atributos involucrados en dicho proceso podrán cambiar. Así que definiremos el conjunto de todos los posibles resultados de dicho experimento (espacio muestral), Ω, como:

$$\Omega = \left\{ (x_i)_{i\in\{1,\dots,m\}} \middle| \begin{array}{l} x_i \text{ es el valor (medida o} \\ \text{estimación) del } i-\text{ésimo} \\ \text{atributo} \end{array} \right\},$$

donde $\Omega \subseteq \mathbf{R}^m$.

Cada vector numérico, s de Ω es una realización de un vector aleatorio $(X_1, \dots, X_m)$, definido como una función:

$$(X_1, \dots, X_m): \Omega \to \mathbf{R}^m,$$

donde a cada $s \in \Omega$ se le asocia un vector de m números reales $\big(X_1(s), \dots, X_m(s)\big)$ y $X_j\big((x_i)_{i\in\{1,\dots,m\}}\big) = x_j \ \forall j \in \{1, \dots, m\}$, es decir cada X_j es una variable aleatoria.

Estas m variables aleatorias representan los atributos del proceso de estimación del esfuerzo de desarrollo a los que se les puede asociar un número (medición o estimación).

De acuerdo con Jiang *et al.* [13] y Kemerer *et al.* [14] la variable más significativa para predecir el esfuerzo de desarrollo es el tamaño del software. Por consiguiente, una vez establecido el marco de condiciones iniciales, para estimar el esfuerzo de desarrollo únicamente se considerará el valor estimado del tamaño del software. Entonces el espacio muestral, Ω, quedará definido de la manera siguiente:

$$\Omega = \left\{ (x,y) \middle| \begin{array}{c} y = \text{esfuerzo de desarrollo} \\ \text{del software,} \\ x = \text{tamaño del software} \end{array} \right\},$$

donde $y > 0$, $x > 0$ y sus variables aleatorias correspondientes están definidas como: $Y: \Omega \to \mathbf{R}^+$, $X: \Omega \to \mathbf{R}^+$, con $Y\big((x,y)\big) = y$ y $X\big((x,y)\big) = x$, $\forall (x,y) \in \Omega$, respectivamente.

Debido a que la variabilidad del esfuerzo no es completamente explicada por el tamaño del software, el resto de las variables significativas para la estimación del esfuerzo se podrán considerar a través de la elección de las condiciones iniciales del proceso de estimación.

En nuestra ley asumimos que la variabilidad del esfuerzo de desarrollo (Y_t) es proporcional a su estimación ($\hat{Y}_t$) para un tamaño dado. Esto implica que los datos presentan heteroscedastidad. Matemáticamente representamos esta relación con la ecuación (1):

$$Var(Y_t) = c^2 \hat{Y}_t^2 \ \text{ para } \ t \in T = \{1, \dots, n\}, \qquad (1)$$

donde c es una constante.

Es decir la varianza del esfuerzo debe aumentar conforme aumenta el tamaño y el esfuerzo de desarrollo del software. En situaciones como esta, la gráfica de los datos de esfuerzo frente a los estimados se ensanchará como un abanico hacia la derecha. Este supuesto se justifica en el hecho de que pedir que la varianza sea fija para cualquier valor del tamaño del software implicará que los intervalos de confianza y de predicción para el esfuerzo también sean fijos. Esto es absurdo si consideramos que el tamaño del error de una estimación deberá aumentar conforme el software se vuelve más complejo.

En estas circunstancias de acuerdo con [19, 20] estamos esencialmente asumiendo una estructura multiplicativa de los errores de

estimación del esfuerzo. En otras palabras, el error es un múltiplo del promedio del esfuerzo de desarrollo. Modelamos estadísticamente esta relación con la ecuación (2):

$$Y_t = \widehat{Y_t} \cdot u_t \quad \text{para} \quad t \in T = \{1, \dots, n\}, \tag{2}$$

En el modelo (2) se supone que:

1. La variable Y_t se distribuye log-normalmente porque este tipo de distribuciones son comunes cuando los valores de sus medias son bajos, sus varianzas son grandes y los valores de la variable aleatoria dependiente no deben ser negativos.
2. $\{Y_t\}_{t \in T}$ es una sucesión de variables aleatorias independientes e idénticamente distribuidas (iid). Esto es, porque suponemos independencia estadística entre los experimentos aleatorios que se ejecutan cada vez que se ejecuta el proceso de estimación del esfuerzo de desarrollo.
3. $\widehat{Y_t}$, (el estimador de Y_t), es una variable aleatoria lognormal que está en función de X_t (la *t*-ésima variable aleatoria que mide el tamaño de un software) es decir, $\widehat{Y_t} = f(X_t)$, sustituyendo $f(X_t)$ en (2) tenemos que:

$$Y_t = f(X_t) \cdot u_t \text{ para } \quad t \in T = \{1, \dots, n\}, \tag{3}$$

   si $f(\cdot)$ es conocido excepto por un vector desconocido de parámetros, θ, a ser estimados tenemos que $f(X_t) = f(X_t, \theta)$ y el modelo es llamado paramétrico; en caso contrario es llamado no-paramétrico.
4. La desviación estándar de Y_t es proporcional a $f(X_t)$ ($SD(Y_t) = cf(X_t)$), es decir la varianza del esfuerzo debe aumentar conforme aumenta el esfuerzo mismo.
5. Se tienen almacenadas las mediciones de tamaño, X_t, y de esfuerzo de desarrollo, Y_t, de n proyectos históricos. Estos proyectos se realizaron bajo el mismo marco de condiciones iniciales que el del proyecto nuevo. Denotamos por $\{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$, a una muestra de tamaño n y, por $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} = \{x_t, y_t\}_1^n$, a las observaciones de dicha muestra. Cualquier especificación práctica de $f(\cdot)$ en (3) dependerá de la información disponible, $\{x_t, y_t\}_1^n$. Independientemente de esta información, $f(\cdot)$ siempre deberá ser positiva, continua, monótona estrictamente creciente y $f(0) \approx 0$. Pedimos que $f(\cdot)$ sea positiva porque el esfuerzo de desarrollo está restringido a valores positivos. $f(\cdot)$ debe ser monótona estrictamente creciente ya que, por lógica, si nosotros cuidamos que cada experimento aleatorio se realice bajo el mismo marco de condiciones iniciales, entonces siempre que el tamaño de un software sea mayor que el de otro, el desarrollo del primero deberá implicar mayor esfuerzo que el del segundo. Pedimos que $f(\cdot)$ sea continua para evitar que pequeñas variaciones del tamaño den lugar a grandes variaciones del esfuerzo. Pedimos que $f(0) \approx 0$, porque si ya se tiene bien especificado el marco de condiciones iniciales, las actividades que se deben realizar antes de iniciar la fase de codificación no deben implicar demasiado esfuerzo.
6. El término de error u_t, es una variable aleatoria lognormal que mide el error de la estimación de Y_t, su función de densidad de probabilidad está definida sobre un soporte (0,+∞), con media unitaria ($E(u_t) = 1$) y varianza constante pero desconocida, $\sigma_{u_t}^2$.
7. u_t, se supone independiente del tamaño del software, X_t.
8. $\{u_t\}_{t \in T}$ se supone independiente e idénticamente distribuida (iid) sobre todo el intervalo de datos.

Existe una variedad de métodos que corrigen la heterocedasticidad, sin embargo para la heterocedasticidad representada en la ecuación (1), de acuerdo con [19], la primera solución es transformar los datos originales tomando su logaritmo natural.

Si aplicamos la transformación logarítmica: $Z_t := ln(Y_t)$., $g(X_t) := ln\big(f(X_t)\big)$ y $\varepsilon_t := ln(u_t)$, obtenemos que los errores multiplicativos, u_t, se convierten en aditivos, ε_t, y la varianza de Z_t es constante:

$$Z_t = g(X_t) + \varepsilon_t, \tag{4}$$

donde Z_t, $g(X_t)$ y ε_t son variables aleatorias normales y, X_t y ε_t son independientes [21]. Dado que suponemos que ambas sucesiones $\{Y_t\}_{t \in T}$ y

$\{u_t\}_{t\in T}$, son lognormales, entonces $\{Z_t\}_{t\in T}$ y $\{\varepsilon_t\}_{t\in T,}$ son iid normales [22].

En todos los modelos de regresión y en particular para el modelo (4) el caso ideal es que el error ε_t sea aleatorio, es decir que sea creado por un proceso de ruido blanco gaussiano de media cero (*zero mean additive white Gaussian noise*, o *zero mean* AWGN), [23, 24, 25]. De acuerdo con Ruppert [26] y Fan [27], la secuencia $\{\varepsilon_t\}_{t\in T}$ es un AWGN de media cero si:

$$\varepsilon_i \text{ tiene distribución normal para toda } i; \tag{5}$$

$$Var(\varepsilon_i) = \sigma^2. \text{ (una constante) para toda } i; \tag{6}$$

$$Corr(\varepsilon_i, \varepsilon_j) = 0. \text{ para toda } i \neq j; \tag{7}$$

$$E(\varepsilon_i) = 0. \text{ para toda } i. \tag{8}$$

Por otro lado, dado que $f: \mathbf{R}^+ \to \mathbf{R}^+$, entonces $g: f(\mathbf{R}^+) \to \mathbf{R}$. Además, como las funciones $f(\cdot)$ y $ln(\cdot)$ son continuas y monótonas estrictamente crecientes, entonces $g(\cdot) = f \circ ln(\cdot)$, también lo será. Además de que dado que $f(0) \approx 0$ entonces $g(0) = ln(f(0)) \approx 1$.

Partiendo del supuesto de que $g(\cdot)$ existe y es calculable, ¿cómo arribamos a un valor de $g(\cdot)$?, ¿y cómo decidimos si este valor es óptimo o no?, y si es óptimo, ¿en qué sentido? Para responder a estas preguntas, primero precisamos elegir una función de costo que penalice el error de estimación. La estimación resultante $g(\cdot)$ será óptima sólo en el sentido de que conduzca al valor mínimo de costo. La elección de una función de costo diferente, generalmente, conducirá a una elección diferente para $g(\cdot)$. Cada una de ellas siendo óptima a su manera. El criterio de diseño que nosotros adoptamos es el criterio de error cuadrático medio. Hay varias buenas razones para elegir este criterio. La más sencilla es que este criterio, más que ningún otro, es susceptible de manipulaciones matemáticas. Adicionalmente, este criterio esencialmente intenta forzar a que el error de estimación tome valores cercanos a su media, es decir cero [28]. El error de estimación, ε_t, se define en (9) de la manera siguiente:

$$\varepsilon_t \coloneqq Z_t - \hat{Z}_t, \tag{9}$$

para determinar $\hat{Z}_t$ mínimo sobre todas las funciones $g(\cdot)$ posibles:

$$\min_{g(\cdot)} E(\varepsilon_t^2), \tag{10}$$

la solución está dada por el teorema siguiente, enunciado por Sayed [28].

Teorema. (Estimador óptimo de error cuadrático medio). El mínimo error cuadrático medio de Z_t dado X_t es la esperanza condicional de Z_t dado X_t, es decir, $\widehat{Z_t} = E(Z_t|X_t)$. La estimación resultante es $\hat{z}_t = g(x_t) = E(Z_t|(X_t = x_t))$. De tal forma que:

$$Z_t|(X_t = x_t) = g(x_t) + \varepsilon_t \quad t = 1, \dots, n, \tag{11}$$

la esperanza condicional puede verse como el concepto generalizado de regresión. En nuestro caso sólo hay un regresando (Z_t) y un regresor numérico (X_t).

De una gran variedad de modelos de regresión que se pueden elegir para realizar la estimación del esfuerzo de desarrollo, el más simple que podemos considerar para establecer la relación entre Z_t y X_t, es el lineal. Sin embargo, para tomar una decisión fundamentada en la información disponible en las muestras que conforman este caso de estudio, se realizó el diagrama de dispersión del tamaño vs el esfuerzo de desarrollo, para cada muestra. Ignorando el ruido y la heterocedasticidad, se determinó que se puede trazar una línea recta que divide la nube de puntos a la mitad. Por lo que se supuso que los datos tienen un comportamiento aproximadamente lineal. Esto significa que $f(\cdot)$ de la ecuación (3) se puede sustituir por $(aX_t + b)$, obteniendo la ecuación (12):

$$Y_t = f(X_t) \cdot u_t \quad = (aX_t + b) \cdot u_t \quad t = 1, \dots, n, \tag{12}$$

donde a y b son parámetros.

Como, por hipótesis, $f(\cdot)$ debe ser positiva, continua, monótona estrictamente creciente y $f(0) \approx 0$, entonces $a > 0$ y $\mathrm{b} \approx 0$. Esto significa que $f(X_t) \approx aX_t$, aplicando logaritmo natural a ambos lados de la ecuación (11) tenemos que:

$$Z_t = ln\big(f(X_t)\big) + \varepsilon_t = g(X_t) + \varepsilon_t,$$

es decir, $g(X_t) = \ln\big(f(X_t)\big) \approx \ln(aX_t) = \ln(X_t) + \ln(a)$, dado que a es un parámetro parece razonable estimar a $g(\cdot)$ con el modelo de regresión logarítmica (13):

$$g(X_t) = \mathrm{c}_1 \ln(X_t) + c_2, \tag{13}$$

donde c_1 y c_2 son parámetros que se calculan por el método de mínimos cuadrados. Se pide que $c_1 > 0$, para que $g(\cdot)$ satisfaga la propiedad de monotonía estrictamente creciente. Por ende:

$$f(X_t) = (\mathrm{e}^{c_2})(X_t)^{c_1}, \tag{14}$$

de tal manera que, conforme c_1 se acerque a uno, $f(\cdot)$ tenderá a ser lineal.

Para estimar los intervalos de confianza y de predicción de $f(\cdot)$ primero se tienen que calcular los intervalos de confianza y de predicción de la función $g(\cdot)$.

Supongamos que los errores tienen una distribución normal y su media (μ_{ε_t}) y desviación estándar (σ_{ε_t}) son desconocidos. Entonces para estimar el intervalo de confianza de $g(\cdot)$, de acuerdo con Helsel en [29], se usa (15) o (16), según sea el caso:

$$\hat{z}_t \pm t_{\left(\frac{\alpha}{2},n-1\right)}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}}\right), \tag{15}$$

para n < 30,

$$\hat{z}_t \pm z_{\frac{\alpha}{2}}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}}\right), \tag{16}$$

para $n \geq 30$.

Con las mismas condiciones para ε_t, de acuerdo con Helsel en [29], para estimar el intervalo de predicción de una observación nueva Z_{n+1} cuando $X_{n+1} = x_{n+1}$., se usa la ecuación (17) o (18), según sea el caso:

$$\hat{z}_t \pm t_{\left(\frac{\alpha}{2},n-1\right)}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}+1}\right), \tag{17}$$

para n < 30,

$$\hat{z}_t \pm z_{\frac{\alpha}{2}}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}+1}\right), \tag{18}$$

para $n \geq 30$.

Una vez estimado el intervalo de confianza de $g(\cdot)$, para estimar el intervalo de confianza de $f(\cdot)$, de acuerdo con [29], se usa (19 o 20), según sea el caso:

$$exp\left(\hat{z}_t \pm t_{\left(\frac{\alpha}{2},n-1\right)}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}}\right)\right), \tag{19}$$

para n < 30,

$$exp\left(\hat{z}_t \pm z_{\frac{\alpha}{2}}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}}\right)\right), \tag{20}$$

para $n \geq 30$.

Una vez estimado el intervalo de predicción de una observación nueva Z_{n+1} cuando $X_{n+1} = x_{n+1}$. Para estimar el intervalo de predicción para una observación nueva Y_{n+1} cuando $X_{n+1} = x_{n+1}$, de acuerdo con [29], se usa la ecuación (21 o 22), según sea el caso:

$$exp\left(\hat{z}_t \pm t_{\left(\frac{\alpha}{2},n-1\right)}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}+1}\right)\right), \tag{21}$$

para n < 30,

$$exp\left(\hat{z}_t \pm z_{\frac{\alpha}{2}}\left(s_{\varepsilon_t}\sqrt{\frac{1}{n}+1}\right)\right), \tag{22}$$

para $n \geq 30$.

En las ecuaciones (15-22) n es el tamaño muestral y s_{ε_t} es la desviación estándar muestral. En las ecuaciones (15, 17, 19 y 21), $t_{\left(\frac{\alpha}{2},n-1\right)}$, es el valor t de Student con $n-1$ grados de libertad y 100(1-α)% de confianza. En las ecuaciones (16, 18, 20 y 22), $z_{\frac{\alpha}{2}}$ es el valor z de la distribución normal estándar con 100(1-α)% de confianza.

Tabla 1. Modelos empíricos del método de regresión lineal para estimar el esfuerzo de desarrollo

Nombre	Modelo empírico
Farr & Zagorski (1964) [2, 37]	$MM = 2.8x_1 + 1.3x_2 + 33x_3 - 17x_4 + 10x_5 + x_6 - 188.$ x_1=número de instrucciones x_2=número de millas recorridas x_3=número de documentos entregados x_4=experiencia del programador del sistema x_5=número de consolas de visualización x_6=porcentaje de instrucciones nuevas
Nelson (1966) [3]	$MM = 9.15x_3 + 10.73x_8 + 0.51x_{26} + 0.46x_{30} + 0.4x_{41} + 7.28x_{42} - 21.45x_{48.1}$ $+13.53x_{48.5} + 12.35x_{51} + 58.82x_{53} + 30.61x_{56} + 29.55x_{72} + 0.54x_{75} - 25.20x_{76}.$ x_3=falta de conocimiento de los requerimientos operacionales x_8=estabilidad del diseño x_{26}=porcentaje de instrucciones matemáticas x_{30}=porcentaje de información almacenada y de funciones de recuperación x_{41}=número de subprogramas x_{42}=lenguaje de programación $x_{48.1}$=software de negocio $x_{48.5}$=software autónomo (*stand-alone*), sí=1, no=0 x_{51}=primer programa en la computadora x_{53}=componentes ADP desarrollados actualmente x_{56}=dispositivo utilizado de acceso aleatorio x_{72}=equipos diferentes para la programación y operación x_{75}=número de viajes por el hombre x_{76}=puntos de datos del programa desarrollado por la organización militar
Wolverton (1974) [33]	$C(k) = S_s(k)C_{i(k),j(k)},$ donde *k* va de 1 a *n*. $C(k) =$ costo de desarrollo del *k*-ésimo modulo. $S_s(k) =$número de líneas de código $C_{i(k),j(k)} =$ *(i(k), j(k))*, la *k*-ésima entrada de la matriz de costos de software. El costo del sistema completo es calculado como la suma de *C(k)*, de 1 hasta *n*.
COPMO (1980) [33]	$E = E_1(S) + E_2(M), E_1(S) = a + bS.$ $E_2(M) = cM^d.$ *E* =esfuerzo, S =tamaño de software, $E_1(S)$ =esfuerzo requerido para desarrollar un modulo. $E_2(M)$=esfuerzo requerido para coordinar a todos los miembros del equipo. M = es el promedio de número de personas asignadas al proyecto.
FPA (1983) [8, 35]	$MH = 53.2(FP) + 12773.$ FP = Puntos de función (*Function Points*)
COBRA (1998) [11]	Esfuerzo= α(Tamaño)

3. Comparación de MUREM con otros métodos

El desempeño de MUREM fue comparado con el de los métodos de regresión siguientes: 1) regresión lineal simple (OLS: *Ordinary Least Squares*), 2) regresión potencial (RP) ajustada por mínimos cuadrados y 3) una red neuronal (RN) entrenada con retropropagación de alimentación hacia adelante (*feed-forward backpropagation network*). A continuación se describe, brevemente, cada uno de ellos.

3.1. Método de regresión lineal simple (OLS)

La regresión lineal simple se usa para modelar la relación lineal entre dos variables [30]. Esta regresión se representa con la ecuación (23):

$$f(X_t) = \mathrm{c}_3 X_t + c_4, \tag{23}$$

donde c_3 y c_4 son parámetros que se calculan por el método de mínimos cuadrados. Para que $f(\cdot)$ satisfaga la propiedad de monotonía estrictamente creciente se pide que $c_3 > 0$.

De acuerdo con [31, 31 y 33], el método de regresión lineal (simple o múltiple) es un método clásico en la estimación del esfuerzo de desarrollo. Como muestra de esto se tienen los modelos empíricos de dicho método que se presentan en la Tabla 1.

3.2. Método de regresión potencial (RP)

El modelo de regresión RP es un método de regresión no lineal. Esto significa que el esfuerzo requerido para la implementación no necesariamente es directamente proporcional al tamaño del software. De acuerdo con [36], esta regresión se representa con la ecuación (24):

$$f(X_t) = \mathrm{c}_5 \cdot {X_t}^{c_6}, \tag{24}$$

donde c_5 y c_6 son dos parámetros de la misma que se calculan por el método de mínimos cuadrados. Se pide que $c_5 > 0$ y $c_6 > 0$ para que $f(\cdot)$ satisfaga la propiedad de monotonía estrictamente creciente.

La ecuación (24) es la forma más común de los modelos no lineales y ha sido utilizada, a través del tiempo, por una gran variedad de modelos empíricos para la estimación del esfuerzo de desarrollo. En la Tabla 2 se muestran algunos de estos modelos.

Uno de los algoritmos de aprendizaje supervisado de red neuronal más usado es el denominado como retropropagación de alimentación hacia adelante (*feed-forward backpropagation network*). Este algoritmo considera tres fases: 1) la alimentación hacia adelante del patrón de entrenamiento de entrada, 2) el cálculo y la retropropagación del error asociado y 3) el ajuste de los pesos. En este estudio comparativo se usó un modelo de red neuronal con este tipo de algoritmo de entrenamiento y fue implementado con la función, "*newff*", de la librería de redes neuronales de MATLAB R2012a.

La arquitectura de esta red está compuesta por una capa oculta con una neurona tangente sigmoidal hiperbólica (*tansig*), seguida por una capa de salida con una neurona puramente lineal (*purelin*). Se consideró solamente una neurona en la capa oculta para lograr que esta red generara funciones monótonas estrictamente crecientes. El algoritmo de entrenamiento de retropropagación que se eligió para esta red fue el de, "*trainbr*", denominado como regularización bayesiana. La regularización bayesiana tiene buenas cualidades de generalización. En la experimentación *trainbr* fue el que tuvo mejor desempeño.

4. Evaluación de MUREM y análisis de resultados experimentales

Para evaluar a MUREM, se comparó su desempeño con el obtenido por los tres métodos de estimación descritos en la sección anterior. Estos métodos se aplicaron a un mismo caso de estudio. Este caso de estudio está conformado por las bases de datos públicas de proyectos de software: Albrecht, China, Desharnais, Finnish, Kemerer, Kitchenham, Maxwell, Mermaid, Miyasaki1 y NASA93.

Tabla 2. Modelos empíricos que utilizan la función potencial para estimar el esfuerzo de desarrollo

Nombre	Modelo empírico
James (1974) [4]	Si el usuario conoce el número de líneas de código entregadas: $MM = 0.01(D)^{1.18}$. D = Miles de líneas de código entregadas (DSI: *Delivered Source Instructions*). Si el usuario conoce el número de líneas de operación (*operating instructions*): $MM = 0.01(O)^{1.24}$. O = Miles de líneas de operación.
GRC - General Research Corporation (1974-79) [6, 4, 37]	$C_s = 0.232(I)^{1.43}$. C_s = Costo total del esfuerzo de desarrollo de software. I = Número de instrucciones en lenguaje máquina.
Walston & Felix (1977) [35, 37]	$MM = 5.2(L)^{0.91}$. L =Miles de líneas de código
Doty (1977) [6, 35, 37]	$MM = 5.288(L)^{1.047}$ para $L \geq 10$, $MM = 2.060(L)^{1.047}(\prod_{i=1}^{14} F_i)$ para $L < 10$, F_i= Multiplicadores del esfuerzo L = Miles de líneas de código
Schneider (1978) [5]	$MM = 0.3(I)^{1.83}$. I = Miles de líneas de código
Bailey & Basili (1981) [7, 6, 37]	$MM = 0.73(DL)^{1.16} + 3.5$. DL = Número de líneas de código desarrolladas
CoCoMO I (1981) [6, 37]	**Básico*: $MM = a(L)^b$. L=miles de líneas de código Orgánico: $MM = 2.4(L)^{1.05}$ Medio: $MM = 3.0(L)^{1.12}$ Embebido: $MM = 3.6(L)^{1.20}$ **Intermedio*: $MM = a(L)^b m(X)$. a y b son parámetros y sus valores son derivados de los tres modos del modelo básico. $m(X)$=Es un factor de ajuste del esfuerzo y es el producto de 15 multiplicadores del esfuerzo. **Detallado*: El modelo detallado incluye todas las características del modelo intermedio con la diferencia de que el impacto de las variables de costo es evaluado en cada fase del proceso de desarrollo.
CoCoMO II (1995) [9, 35]	$MM = A(S)^E(\prod_{i=1}^{17} EM_i)$, donde $E = B + 0.01(\sum_{j=1}^{5} SF_j)$. S = miles de líneas de código. Los valores de A, B, EM_i y SF_j son obtenidos por calibración.

Estas bases, a pesar de tener la gran ventaja de ser ampliamente conocidas y accesibles, cumplen, de manera muy relajada, con el criterio de condiciones iniciales homogéneas. Sin embargo, aun así, MUREM fue capaz de obtener buenos resultados sin sacrificar la validez externa. En la Tabla 3 se muestran la cantidad de proyectos y las unidades que tiene cada una de estas bases para medir el tamaño del software y el esfuerzo de desarrollo. Considerando las unidades del tamaño del software, se obtuvieron 12 muestras etiquetadas como: KemererAFP, KemererKSLOC, AlbrechtAFP, AlbrechtKSLOC, Mermaid, Finnish, Miyasaki1, Maxwell, Desharnais, Nasa93, Kitchenham y China.

Esta comparación se realizó considerando los criterios siguientes:

1. Proporción de muestras que cumplen con la condición de positividad de $f(\cdot)$.
2. Precisión de la estimación puntual de $f(\cdot)$.

3. Proporción de muestras cuyos residuales pasaron la prueba de ruido blanco gaussiano de media cero.
4. Calidad de los intervalos de confianza y de predicción de $f(\cdot)$.

A continuación se describen, brevemente, los resultados obtenidos para cada uno de estos criterios.

4.1. Positividad de $f(\cdot)$

Para estimar los parámetros c_1, c_2, c_3, c_4, c_5 y c_6, de los modelos de regresión (13. 23, 24), respectivamente, se usó la librería *ezyfit* de MATLAB R2012a. En la Tabla 4 se muestran los valores estimados para estos parámetros.

En la Tabla 4 se ve que, dado que $c_5 > 0$ y $c_6 > 0$ para todas las muestras, se tiene que RP satisfizo la propiedad de positividad para el 100% de las muestras. Con OLS, el 58% de las muestras no cumplieron con la condición de positividad. Los valores de tamaño de estas muestras que caigan dentro del intervalo $\left]0, \frac{-c_4}{c_3}\right]$, tendrán valores no positivos del esfuerzo de desarrollo, lo que es antinatural. Con MUREM, independientemente de los valores obtenidos para c_1 y c_2, la estimación del esfuerzo siempre será positiva.

El método RN pasó la prueba de positividad para cada una de las muestras.

En la Tabla 4 también se ve que casi todos los valores de c_1 están muy cercanos a uno, corroborándose el supuesto de que la relación que se establece entre el esfuerzo de desarrollo y el tamaño del software es aproximadamente lineal.

4.2. Precisión de la estimación puntual de $f(\cdot)$

A sugerencia de [31, 39 y 40] se usaron los estadísticos (15) y (16) para medir la precisión de la estimación puntual del esfuerzo de desarrollo.

1. Magnitud promedio del error relativo (MMRE: *Mean Magnitude of Relative Error*), definida en (25):

$$\mathrm{MRE}_t = \left|\frac{y_t - \hat{y}_t}{y_t}\right|, \qquad \mathrm{MMRE} = \frac{1}{n}\sum_{t=1}^{n} \mathrm{MRE}_t \ . \tag{25}$$

2. Nivel de predicción *k* (Pred(*k*): *Prediction Level k*), definido en (26):

$$\mathrm{Pred}(k) = \frac{1}{n}\sum_{t}^{n} \begin{cases} 1 & \text{si } \mathrm{MRE}_t \le k \\ 0 & \text{en otro caso,} \end{cases} \tag{26}$$

si el MMRE es cercano a cero y el Pred(0.25) (para *k*=0.25) cercano a uno, entonces se considera que el método de estimación tiene una buena precisión puntual [31].

En la Tabla 5 se muestra el promedio obtenido del índice MMRE para cada uno de los métodos.

En la Tabla 6 se muestra el promedio obtenido del índice Pred(0.25) para cada uno de los métodos.

Para evaluar si existe una diferencia significativa entre las medias obtenidas para cada uno de los métodos, tanto para el MMRE (de la Tabla 5) como para el Pred(0.25) (de la Tabla 6), se plantearon las hipótesis siguientes:

$$H_0: \mu_{MUREM} = \mu_{OLS,} \qquad H_a: \mu_{MUREM} \neq \mu_{OLS}, \tag{27}$$

$$H_0: \mu_{MUREM} = \mu_{RP,} \qquad H_a: \mu_{MUREM} \neq \mu_{RP,} \tag{28}$$

$$H_0: \mu_{MUREM} = \mu_{RN}, \qquad H_a: \mu_{MUREM} \neq \mu_{RN}. \tag{29}$$

Las hipótesis (27, 28, 29) se contrastaron para ambos índices (MMRE y Pred(0.25). En estas pruebas se utilizó la prueba de hipótesis de igualdad de medias de dos muestras pequeñas para cola inferior (*the two-sample test of small samples for lower tail*) que viene definida en [40, 42]. Se obtuvieron los resultados siguientes:

1. Se aplicó la prueba de dos muestras pequeñas para cola inferior, en la modalidad de varianzas desconocidas y diferentes, para probar si la diferencia entre las medias del MMRE, era estadísticamente significativa. Con un nivel de significación de 0.05, se rechazó la hipótesis nula de cada una de las hipótesis (27, 28, 29). Esto implica que, para este caso de estudio, existe una relación significativa entre el uso de MUREM y la disminución del MMRE.

Tabla 3. Cantidad de proyectos en las bases que conforman este caso de estudio

BD	No. Proyectos	Unidades de tamaño del software	Unidades de esfuerzo
Kemerer	15	AFP KSLOC	Hombre-mes Hombre-mes
Albrecht	24	AFP SLOC	Hombre-mes Hombre-mes
Mermaid	30	AFP	Hombre-hora
Finnish	38	FP	Hombre-hora
Miyasaki1	48	KSLOC	Hombre-mes
Maxwell	62	FP	Hombre-hora
Desharnais	81	AFP	Hombre-hora
Nasa93	93	KSLOC	Hombre-mes
Kitchenham	145	AFP	Hombre-hora
China	499	AFP	Hombre-hora

Tabla 4. Valores estimados para los parámetros c_1, c_2, c_3, c_4, c_5 y c_6

	MUREM		OLS		RP	
Muestra	c_1	c_2	c_3	c_4	c_5	c_6
KemererAFP	0.90	-1.06	0.34	-121.57	2.06E-05	2.28
KemererKSLOC	0.81	0.97	1.39	-39.91	0.02	1.74
AlbrechtAFP	1.49	0.09	54.45	-13387.9	0.12	1.81
AlbrechtKSLOC	1.17	4.99	386.0	-1703.04	413.25	0.98
Mermaid	0.82	4.05	14.74	3354.4	468.42	0.53
Finnish	1.06	1.67	8.99	813.80	40.39	0.80
Miyasaki1	0.99	-0.07	1.92	-48.46	6.5E-05	2.75
Maxwell	0.87	3.24	11.27	633.12	8.66	1.04
Desharnais	0.94	3.04	17.56	-33.09	14.94	1.03
Nasa93	0.93	1.94	5.06	148.8	20.57	0.78
Kitchenham	0.67	3.61	6.18	-148.77	3.63	1.06
China	0.77	3.30	4.19	1881.5	64.31	0.70

Tabla 5. Medias del índice MMRE

Muestra	MUREM	OLS	RP	RN
KemererAFP	0.43	1.06	0.65	0.60
KemererKSLOC	0.49	0.67	0.66	0.60
AlbrechtAFP	0.53	0.90	0.47	0.71
AlbrechtKSLOC	0.44	0.63	0.71	1.25
Mermaid	0.93	2.48	2.06	2.73
Finnish	0.80	1.20	1.29	1.24
Miyasaki1	0.40	1.42	0.92	1.27
Maxwell	0.54	0.64	0.51	0.76
Desharnais	0.58	0.65	0.65	0.65
Nasa93	0.65	2.33	1.45	2.33
Kitchenham	0.68	0.81	0.72	2.36
China	1.04	2.36	1.79	1.75
Media=	**0.63**	**1.26**	**0.99**	**1.35**

Tabla 6. Medias del índice Pred(0.25)

Muestra	MUREM	OLS	RP	RN
KemererAFP	0.40	0.13	0.40	0.33
KemererKSLOC	0.40	0.33	0.13	0.27
AlbrechtAFP	0.54	0.33	0.29	0.50
AlbrechtKSLOC	0.38	0.38	0.33	0.46
Mermaid	0.13	0.17	0.23	0.17
Finnish	0.24	0.29	0.26	0.29
Miyasaki1	0.42	0.17	0.02	0.29
Maxwell	0.23	0.23	0.29	0.21
Desharnais	0.37	0.43	0.43	0.43
Nasa93	0.34	0.16	0.15	0.16
Kitchenham	0.28	0.40	0.37	0.16
China	0.21	0.20	0.19	0.18
Media=	**0.33**	**0.27**	**0.26**	**0.29**

2. Se aplicó la prueba de dos muestras pequeñas para cola inferior, en la modalidad de varianzas desconocidas pero iguales, para probar si la diferencia entre las medias del Pred(0.25), era estadísticamente significativa.

Con un nivel de significación de 0.05, se rechazó la hipótesis alterna de cada una de las hipótesis (27, 28, 29). Esto significa que las medias del índice Pred(0.25), se consideran estadísticamente iguales.

4.3. Proporción de muestras cuyo error es creado por un AWGN de media cero

Las hipótesis (30, 31, 32), se plantearon para contrastar el supuesto de AWGN de media cero en los errores:

Tabla 7. Proporción de muestras que cumplen con la propiedad de normalidad (KS) en los residuales

Muestra	KS MUREM	KS OLS	KS RP	KS RN
KemererAFP	1	1	1	1
KemererKSLOC	1	1	1	1
AlbrechtAFP	1	1	1	1
AlbrechtKSLOC	1	1	1	1
Mermaid	1	1	1	1
Finnish	1	1	1	1
Miyasaki1	1	0	0	0
Maxwell	1	1	1	1
Desharnais	1	1	1	1
Nasa93	1	0	0	0
Kitchenham	1	0	0	0
China	1	0	0	0
Proporción=	**1.00**	**0.67**	**0.67**	**0.67**

Tabla 8. Proporción de muestras que cumplen con la propiedad de normalidad (JB) en los residuales

Muestra	JB MUREM	JB OLS	JB RP	JB RN
KemererAFP	1	1	1	1
KemererKSLOC	1	0	1	0
AlbrechtAFP	0	1	1	0
AlbrechtKSLOC	1	0	0	1
Mermaid	1	0	0	0
Finnish	1	1	1	1
Miyasaki1	1	0	0	0
Maxwell	1	0	0	0
Desharnais	0	0	0	0
Nasa93	0	0	0	0
Kitchenham	0	0	0	0
China	1	0	0	0
Proporción=	**0.67**	**0.25**	**0.33**	**0.25**

$$H_0: \pi_{MUREM} = \pi_{OLS}, \quad H_a: \pi_{MUREM} \neq \pi_{OLS}, \tag{30}$$

$$H_0: \pi_{MUREM} = \pi_{RP}, \quad H_a: \pi_{MUREM} \neq \pi_{RP}, \tag{31}$$

$$H_0: \pi_{MUREM} = \pi_{RN},$$
$$H_a: \pi_{MUREM} \neq \pi_{RN}, \quad (32)$$

para contrastar estas hipótesis en todos los supuestos (5, 6, 7 ,8) de AWGN de media cero se usó la prueba Chi-cuadrado de Pearson para la independencia de variables. Esta prueba está definida en [43].

a. Contraste de normalidad en los errores.

Se probó el supuesto (5) de AWGN de media cero, el cual asume que los errores tienen distribución normal. Para ello se contrastaron, dos veces, las hipótesis (30, 31, 32). Primero se contrastaron utilizando el estadístico de Kolmogorov-Smirnov (KS), definido en [44], y después utilizando el estadístico de Jerque-Bera (JB), definido en [45].

En las Tablas 7 y 8 se presenta la proporción de muestras que pasaron la prueba de normalidad del error. En la Tabla 7 es con respecto al estadístico de Kolmogorov-Smirnov (KS) y en la Tabla 8 es con respecto al estadístico de Jerque-Bera (JB). En ambas tablas, la entrada 1, significa que no se rechaza la hipótesis de normalidad de los residuales y la entrada 0 significa que se rechaza.

En las tres pruebas de las hipótesis (30, 31) y (32) el valor-p (de la prueba Chi-cuadrado de Pearson), obtenido fue de $p = 0.028$. Este valor es lo suficientemente pequeño (menor que 0.05), para rechazar la hipótesis nula de las tres hipótesis. Esto significa que las diferencias encontradas entre las proporciones difícilmente pueden ser explicadas por el azar, siendo mayor la proporción de muestras cuyos residuales presentan normalidad cuando se aplica MUREM.

En las pruebas de las hipótesis (30, 32) el valor-p obtenido fue de $p = 0.04$. Este valor es lo suficientemente pequeño para rechazar la hipótesis nula en ambas hipótesis. Esto significa que la proporción obtenida por MUREM es estadíscamente mayor que las proporciones obtenidas con los métodos OLS y RN.

En la prueba de la hipótesis (31), el valor-p obtenido fue de $p = 0.102$. Este valor es lo suficientemente grande (mayor ó igual que 0.05) para rechazar la hipótesis alterna. Esto significa que la diferencia entre las proporciones obtenidas por los métodos MUREM y RP puede ser ocasionada por el azar.

b. Contraste de homocedasticidad en los errores.

Las hipótesis (30, 31, 32), se contrastaron para probar el supuesto (6) de AWGN de media cero. Este supuesto asume homocedasticidad en los errores.

Para probar la homocedasticidad de los residuales se aplicó una prueba de hipótesis basada en el estadístico de White. Esta prueba se describe en [46].

En la Tabla 9 se presenta la proporción de muestras que pasaron la prueba de White para la homocedasticidad. La entrada 1, significa que no se rechaza la hipótesis de homocedasticidad en los residuales y la entrada 0 significa que se rechaza.

En las pruebas de las hipótesis (30, 31, 32) los valores-p obtenidos fueron de: $p = 0.0093$, $p = 0.014$ y $p = 0.04$, respectivamente. Estos valores son lo suficientemente pequeños para rechazar la hipótesis nula de las tres hipótesis.

Esto significa que las diferencias encontradas entre las proporciones difícilmente pueden ser explicadas por el azar, siendo mayor la proporción de muestras cuyos residuales presentan homocedasticidad cuando se aplica MUREM.

Es importante hacer notar que si el supuesto de homocedasticidad en los errores no se cumple, entonces el estimador de mínimos cuadrados ordinarios pierde su eficiencia.

c. Contraste de ausencia de autocorrelación en los errores.

Las hipótesis (30, 32), se contrastaron para probar el supuesto (7) de AWGN de media cero, Este supuesto asume la ausencia de autocorrelación en los errores. La contrastación de la hipótesis (31) no se realizó porque en la experimentación resultó que $\pi_{MUREM} = \pi_{RP}$.

Para contrastar la existencia de autocorrelación (lineal) de primer orden en los residuales, se aplicó una prueba de hipótesis basada en el estadístico de Durbin-Watson (DW). Esta prueba se describe en [47].

En la Tabla 10 se presenta la proporción de muestras que pasaron la prueba de Durbin-Watson (DW) para cada uno de los métodos. La entrada 1, significa que no existe autocorrelación, el 0 significa que existe autocorrelación y el 2 significa que no se puede concluir nada.

En las pruebas de las hipótesis (30, 32), los valores-p obtenidos fueron de: $p = 0.3359$ y $p = 0.13$, respectivamente. Estos valores son lo suficientemente grandes para rechazar la hipótesis alterna de ambas hipótesis. Esto significa que la diferencia entre las proporciones puede ser explicada por el azar.

d. Contraste de supuesto de media cero de los errores.

La hipótesis (32) se contrastó para probar el supuesto (8) de AWGN de media cero. Este supuesto asume que los errores tienen media cero. Para determinar si la muestra de residuales cumple con este supuesto, se usó la prueba de hipótesis de media cero de los errores, con varianza desconocida. Esta prueba está definida en [48]. En la Tabla 11, se presenta la proporción de muestras que pasaron dicha prueba. La entrada 1, significa que no se rechaza la hipótesis de media cero de sus residuales y la entrada 0 significa que se rechaza. La contrastación de las hipótesis (30, 31), no se realizó porque en la experimentación resultó que:

$$\pi_{MUREM} = \pi_{OLS} \text{ y } \pi_{MUREM} = \pi_{RP}.$$

En la prueba de la hipótesis (32), el valor-p obtenido fue de $p = 0.3$. Este valor es lo suficientemente grande para rechazar la hipótesis alterna. Esto significa que la diferencia entre las proporciones puede ser explicada por el azar.

4.3. Calidad de los intervalos de confianza y de predicción de $f(\cdot)$

De acuerdo con [49], la precisión de un estimador por intervalo de confianza, se determina por el ancho de su intervalo de la manera siguiente: entre más angosto sea su intervalo, más preciso será su estimador.

En nuestro caso, comparar las longitudes de los intervalos de confianza, como se propone en [49], para determinar cuál método nos genera los resultados más precisos, es una empresa, relativamente difícil. Esto es porque a diferencia del resto de los métodos, la longitud del intervalo de confianza de MUREM no es fijo para todos los valores del tamaño. Esto se puede constatar en las Figuras 1 a 12.

Tabla 11. Proporción de muestras que cumplen con la propiedad de media cero de los residuales

Muestra	MUREM	OLS	RP	RN
KemererAFP	1	1	1	1
KemererKSLOC	1	1	1	1
AlbrechtAFP	1	1	1	1
AlbrechtKSLOC	1	1	1	1
Mermaid	1	1	1	1
Finnish	1	1	1	1
Miyasaki1	1	1	1	1
Maxwell	1	1	1	1
Desharnais	1	1	1	1
Nasa93	1	1	1	1
Kitchenham	1	1	1	0
China	1	1	1	1
Proporción=	**1**	**1**	**1**	**0.92**

En el método MUREM las longitudes mínimas del intervalo de predicción se presentan en los proyectos de menor tamaño y las longitudes máximas se presentan en los proyectos de mayor tamaño. Esto ocasiona que la mayoría de las observaciones que se salen de este intervalo correspondan con sistemas de software de tamaño pequeño.

En cambio, en el resto de los métodos, la longitud del intervalo de predicción es fijo para todos los valores del tamaño.

Esto ocasiona que la mayoría de las observaciones que se salen del intervalo de predicción sean aquellas que se correspondan con sistemas de software de tamaño grande. Esto aumenta, considerablemente, la pérdida económica asociada al tamaño del error de estimación. En las Figuras 1 a 12 se presentan los intervalos de confianza y de predicción, superpuestos a los datos de la muestra.

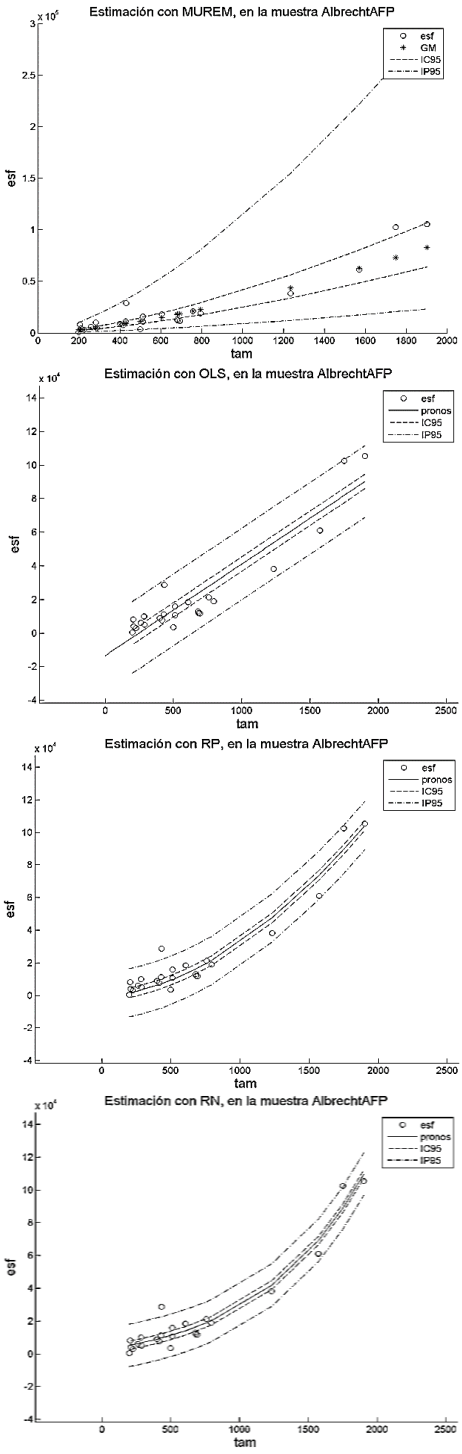


Fig. 1. Estimación del esfuerzo en la muestra AlbrechtAFP

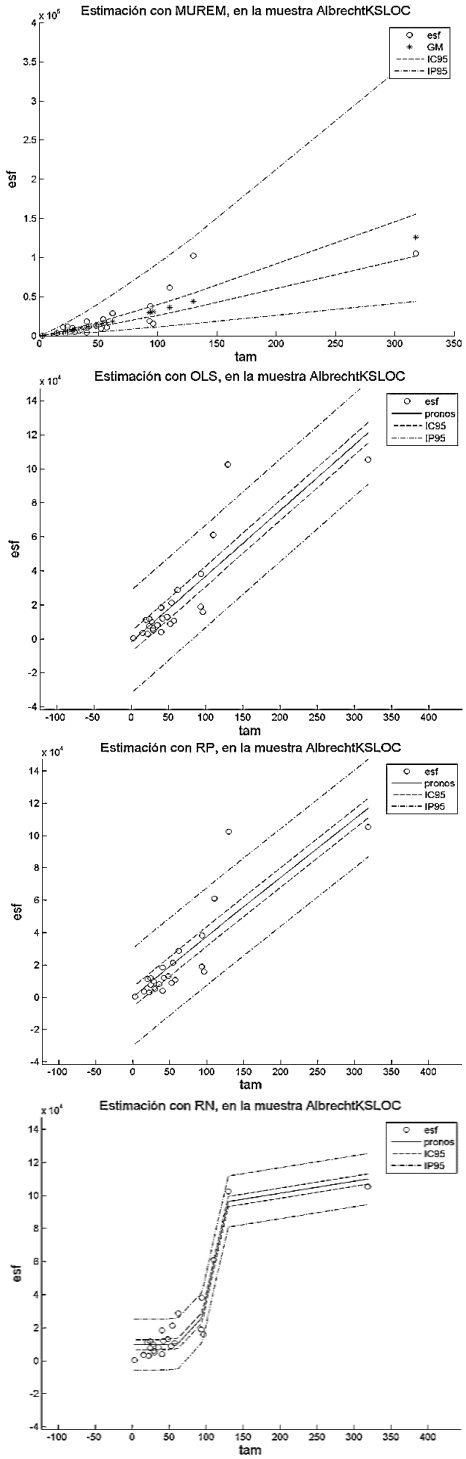


Fig 2. Estimación del esfuerzo en la muestra AlbrechtKSLOC

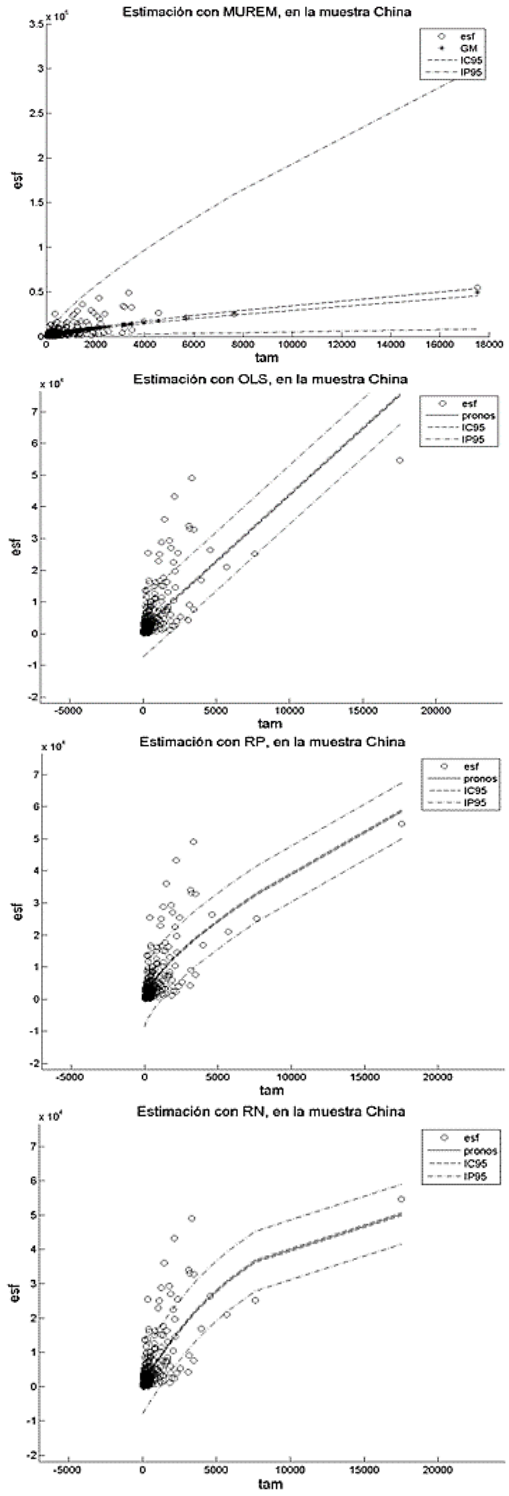


Fig. 3. Estimación del esfuerzo en la muestra China

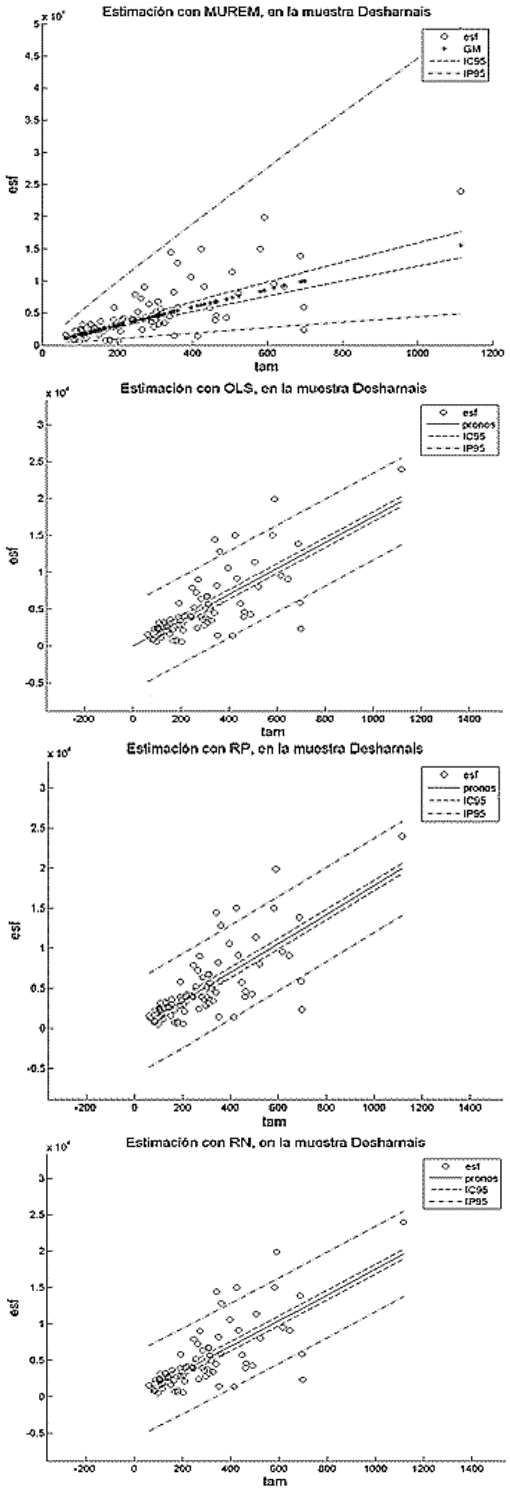


Fig. 4. Estimación del esfuerzo en la muestra Desharnais

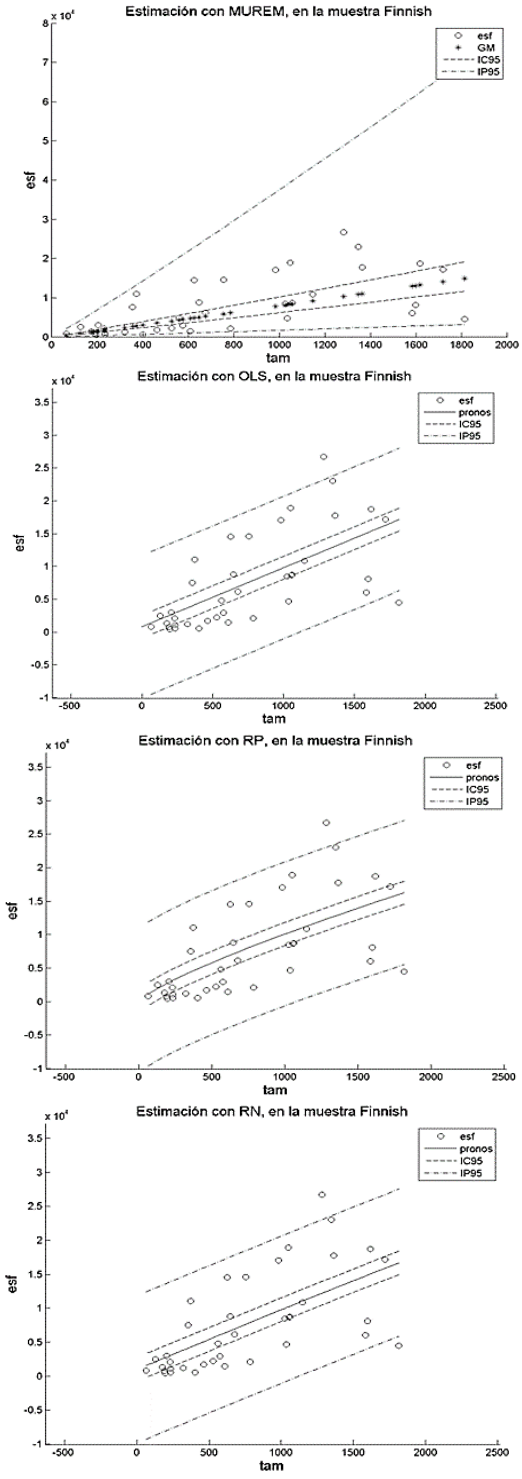


Fig. 5. Estimación del esfuerzo en la muestra Finnish

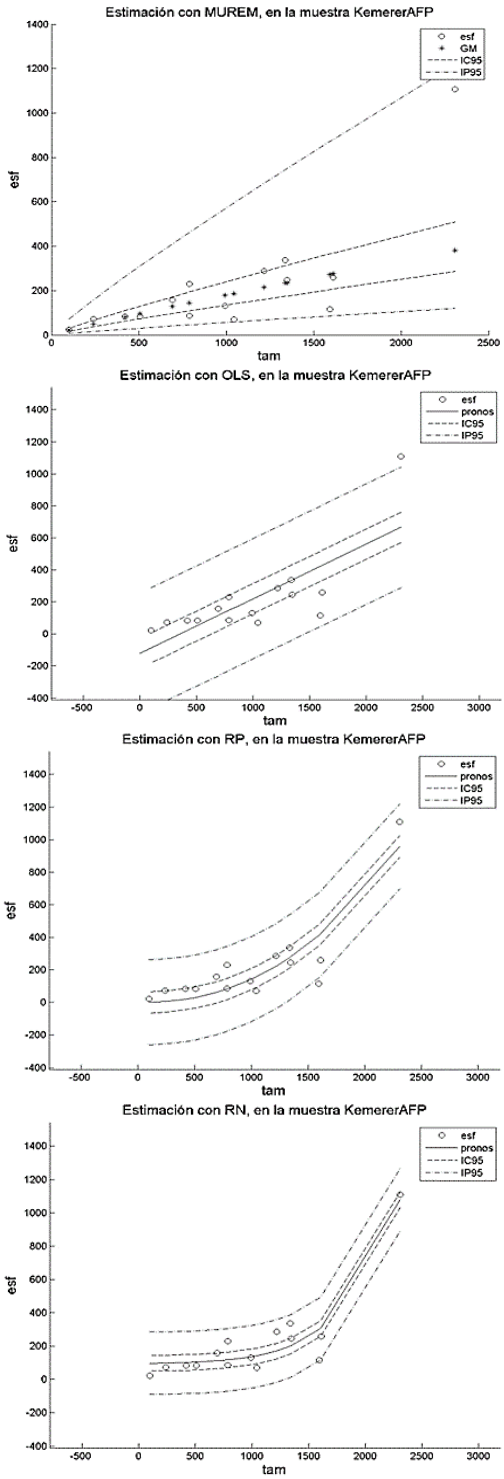
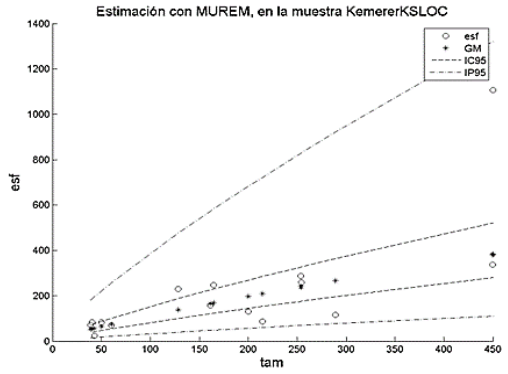
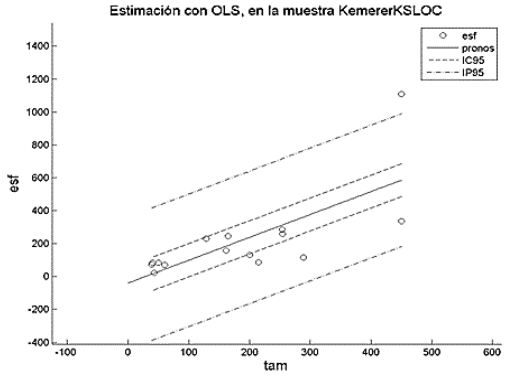
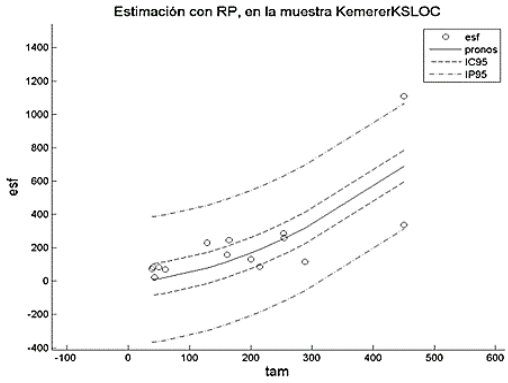


Fig. 6. Estimación del esfuerzo en la muestra KemererAFP

ISSN 2007-9737







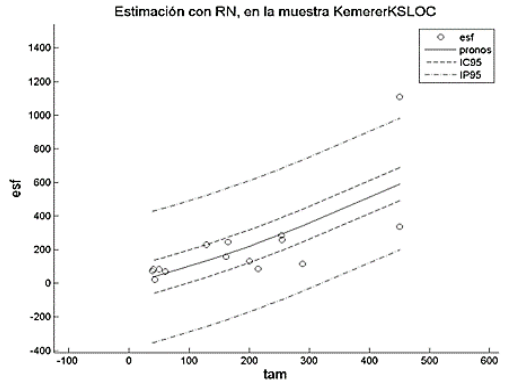
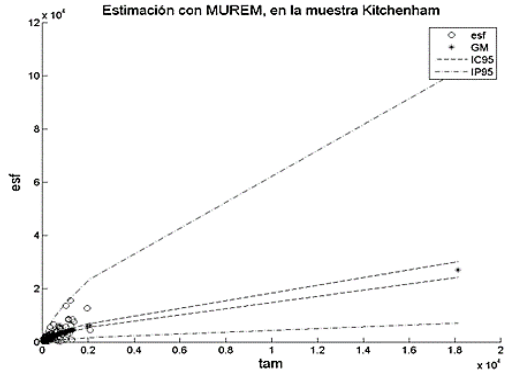
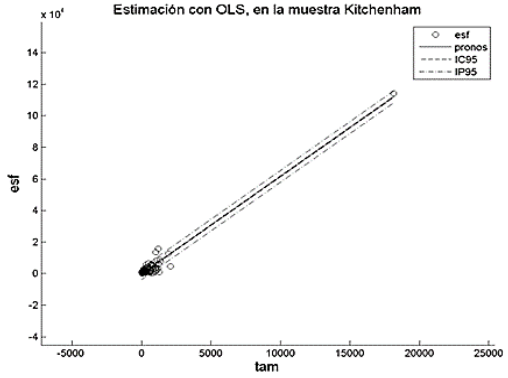
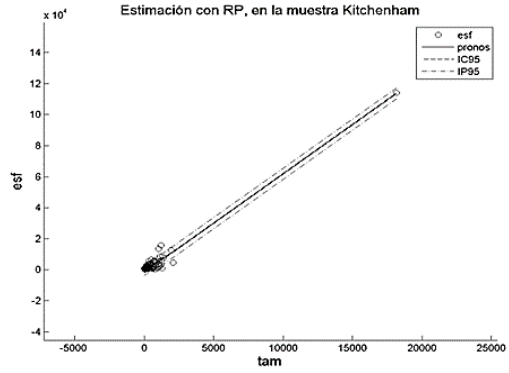


Fig. 7. Estimación del esfuerzo en la muestra KemererKSLOC







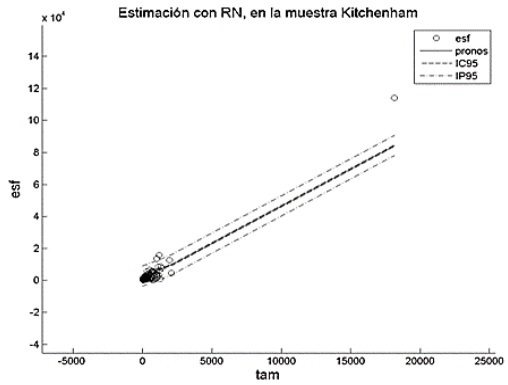


Fig. 8. Estimación del esfuerzo en la muestra Kitchenham

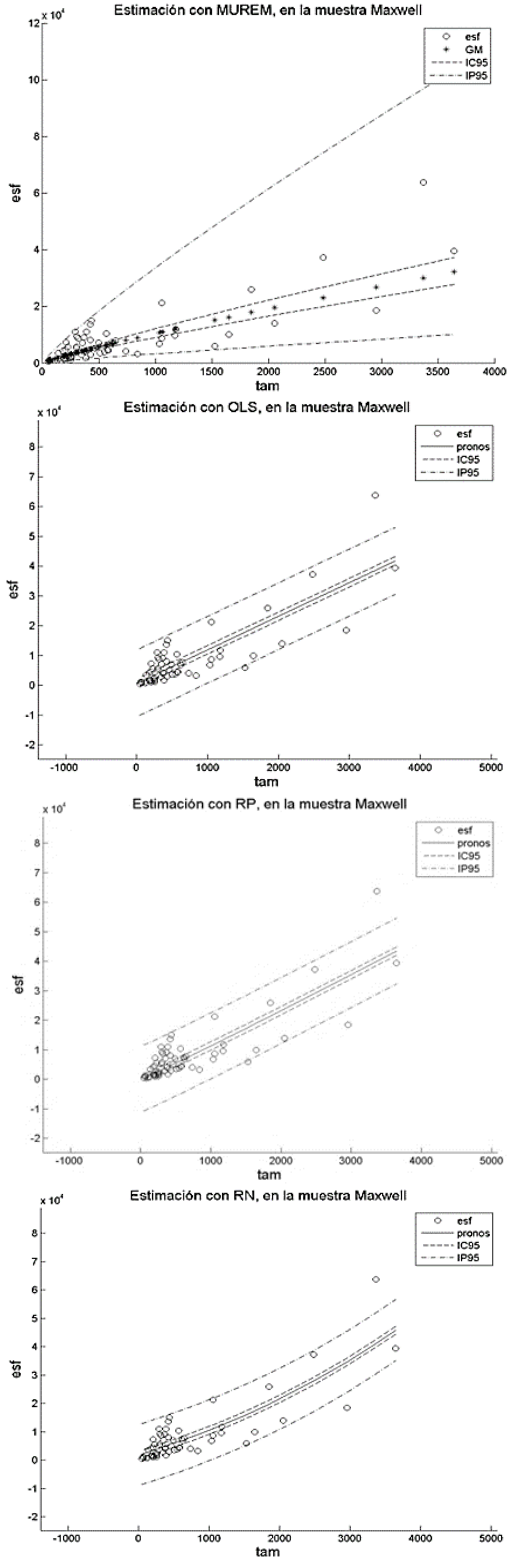


Fig. 9. Estimación del esfuerzo en la muestra Maxwell

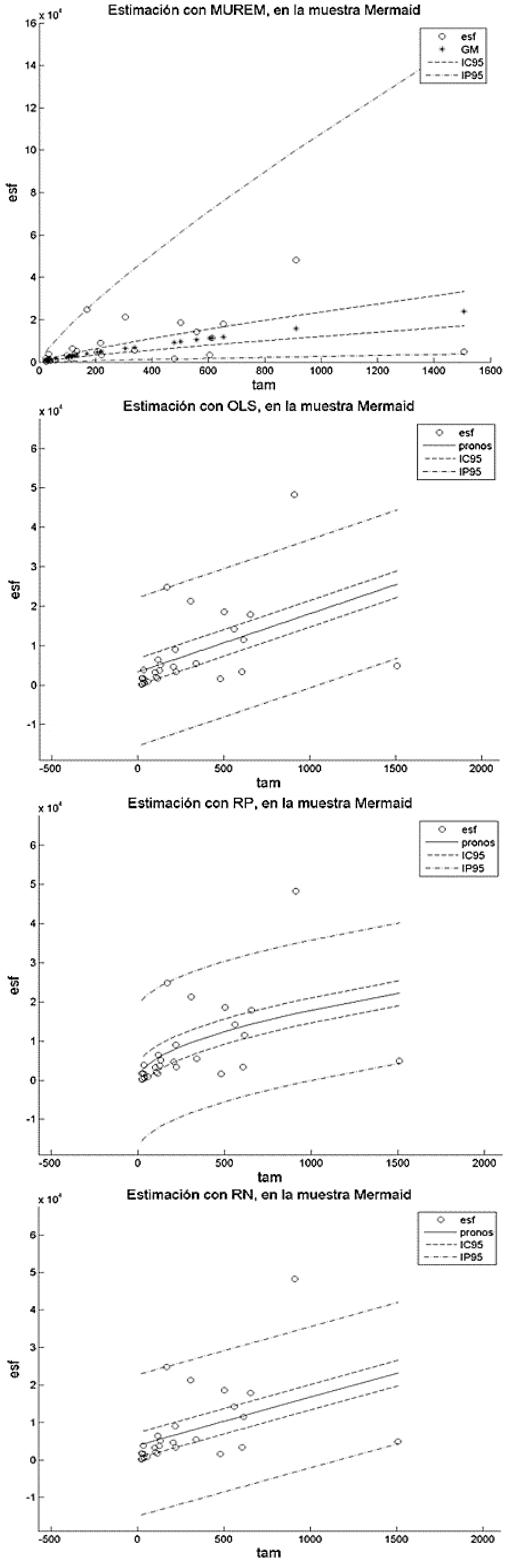


Fig. 10. Estimación del esfuerzo en la muestra Mermaid

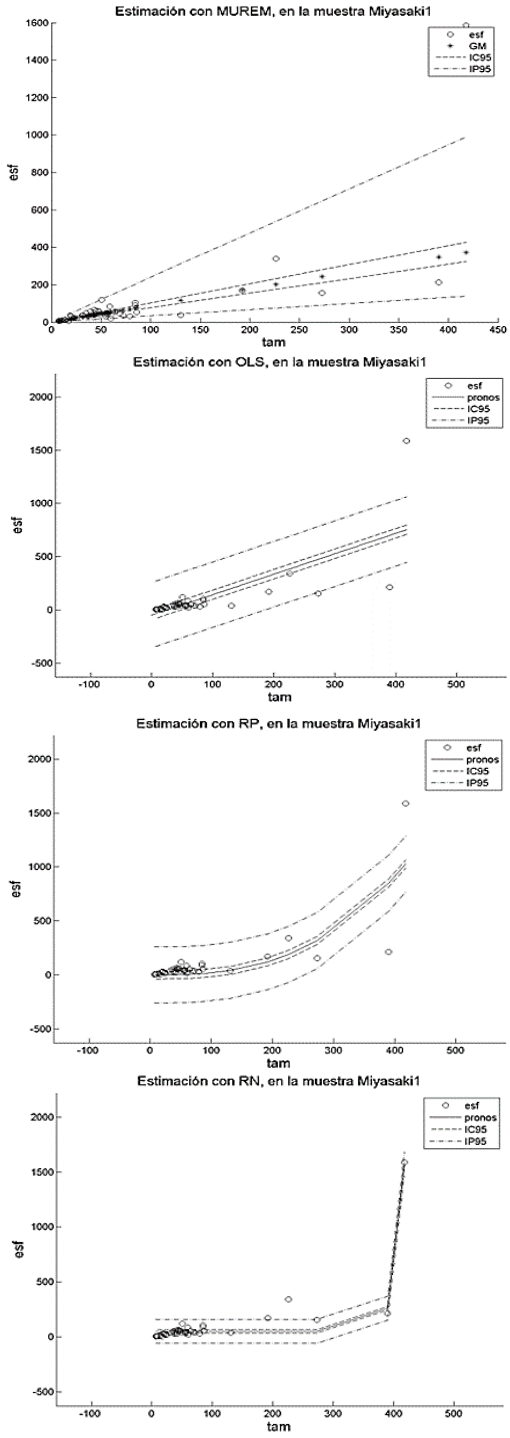


Fig. 11. Estimación del esfuerzo en la muestra Miyasaki1

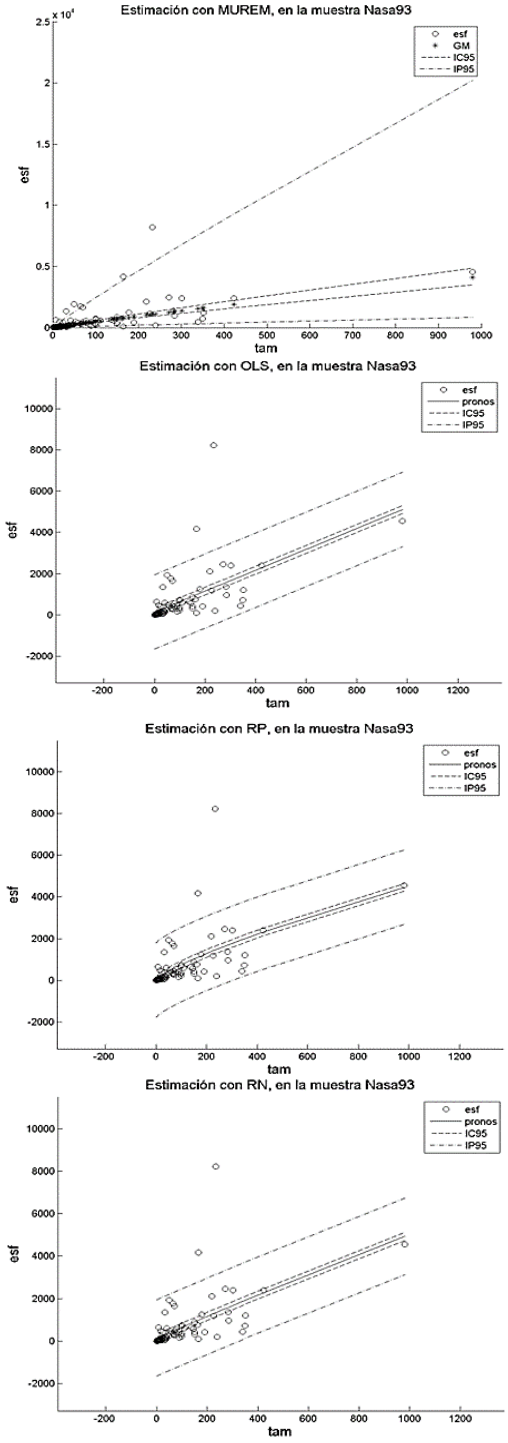


Fig. 12. Estimación del esfuerzo en la muestra Nasa93

Las leyendas: esf, IC95 y IP95, en las gráficas de las Figuras 1 a 12, representan, respectivamente, los valores del esfuerzo, del intervalo de confianza y del intervalo de predicción (ambos intervalos calculados con un 95% de confianza). La leyenda GM, en las gráficas del método MUREM, representa el valor de la estimación usando la ecuación (14): La leyenda pronos, en las gráficas

5. Conclusiones

Nuestro método de estimación del esfuerzo de desarrollo, denominado como MUREM, se sintetizó a partir de la teoría y se contrasta con la experiencia. Se basa en el establecimiento de un conjunto de condiciones iniciales que enmarca el proceso de estimación del esfuerzo de desarrollo, conjuntamente, con una ley *a priori* que estipula las propiedades que, racionalmente, debe satisfacer la relación que se establece entre el esfuerzo de desarrollo y el tamaño del software. Para elegir la fórmula básica de MUREM se buscó que, además de satisfacer nuestra ley *a priori*, se ajustara bien a las muestras extraidas de las bases de proyectos históricos de este caso de estudio. Este caso está conformado por las bases de datos públicas de proyectos históricos de software: Albrecht, China, Desharnais, Finnish, Kemerer, Kitchenham, Maxwell, Mermaid, Miyasaki1 y NASA93.

El desempeño de MUREM fue comparado con el de los métodos siguientes: regresión lineal simple (OLS), regresión potencial (RP) ajustada por mínimos cuadrados y una red neuronal (RN) entrenada con retropropagación de alimentación hacia adelante. De acuerdo con los resultados obtenidos en la experimentación se tiene que:

1. A diferencia de los métodos OLS, RP y RN, MUREM nos asegura que la estimación del esfuerzo de desarrollo será positiva para cualquier muestra. En esta experimentación, OLS logró que solamente el 42% de las muestras cumplieran con esta condición.
2. MUREM genera estimaciones puntuales más precisas que el resto de los métodos. Las medias del índice MMRE son de: 0.63, 1.26, 0.99 y 1.35, para MUREM, OLS, RP y RN, respectivamente. Con esto, MUREM redujo, casi a la mitad, los valores obtenidos por los otros métodos.
3. Los residuales obtenidos por MUREM satisfacen la prueba de ruido blanco gaussiano de media cero. Con esto se prueba que el error de estimación de dicho método es aleatorio. El resto de los métodos presentan problemas con la satisfacción de las propiedades de homocedasticidad y normalidad en sus errores.

Las proporciones de las muestras cuyos residuales presentan homocedasticidad son de: 0.75, 0.08, 0.25 y 0.33 para MUREM, OLS, RP y RN respectivamente. Con esto se muestra que MUREM funciona bien para corregir la heterocedasticidad de los datos.

Las proporciones de las muestras cuyos residuales presentan normalidad, cuando se aplica el estadístico de Kolmogorov-Smirnov (KS), son de: 1.00, 0.67, 0.67 y 0.67; para MUREM, OLS, RP y RN, respectivamente. Con esto se muestra que MUREM aumenta, aproximadamente, el 33% de las muestras que cumplen con esta propiedad. La normalidad del error es un supuesto teórico en la deducción de las fórmulas que se usan para estimar los intervalos de confianza y de predicción. Si el error no cumple con este supuesto no se garantiza que la estimación de los intervalos sea precisa.

A diferencia de los métodos OLS, RP y RN, MUREM genera intervalos de predicción pequeños, para tamaños pequeños del software, y grandes, para tamaños grandes. Con esto se disminuye considerablemente la pérdida económica asociada al tamaño del error de estimación.

6. Trabajo futuro

Como trabajo futuro se propone comparar a MUREM con aquellos métodos nuevos de estimación del esfuerzo de desarrollo que, desde su creación, cumplan con la propiedad de monotonía creciente de MUREM.

Agradecimiento

Este trabajo fue parcialmente patrocinado por el Consejo Nacional de Ciencia y Tecnología (CONACyT) de México.

Referencias

1. **Abdel, T. & Madnick, S. (1986).** Impact of Schedule Estimation on Software Project Behavior. *IEEE Software*.
2. **Farr, L. & Nanus, B. (1964*). Factors that affect the cost of computer programming*. Technical Report TM-1447/000/02, System Development Corporation, Santa Monica, California.
3. **Nelson, R. (1966).** *Management Hand Book for the Estimation of Computer Programming Costs*. Systems Development Corp.
4. **James, T. J. (1977).** Software Cost Estimating Methodology. *IEEE Proceedings of National Aerospace Electronic Conference*.
5. **Schneider, V. (1978).** Prediction of Software Effort and Project Duration- Four New Formulas. *ACM Sigplan,* Vol. 13, No. 6.
6. **Boehm, B. (1981).** *Software Engineering Economics*. Englewood Cliffs.
7. **Bailey, J. J. & Basili, V. R. (1981).** A Meta-model for Software Development Resource Expenditures. *Proc. 5th Int. Conf. Software Eng.,* IEEE/ACM/NBS.
8. **Albrecht, A. & Gaffney, J. (1983).** Software Function, Source Lines of Code, and Development Effort Prediction: a Software Science Validation. *IEEE Transactions on Software Engineering*. DOI:10.1109/TSE.1983.235271.
9. **Boehm, B., Clark, B., Horowitz, E., Madachy, R., Selby, R., & Westland, C. (1995).** Cost Model for Future Software Life Cycle Processes: COCOMO 2.0, Vol. 1, AG Science Publishers, Amsterdam (Holanda), *Annals of Software Engineering Special Volume on Software Process and product Measurement*.
10. **Srinivasan, K. & Fisher, D. (1995).** Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*.
11. **Briand, L., El Emam, K., & Bomarius, F. (1998).** COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment. *International Software Engineering Research Network Technical Report ISERN-9724*.
12. **Burnham, K. P. & Anderson, D. R. (2002).** *Model Selection and Multimodel Inference. A Practical InformationTheoretic Approach,* Springer.
13. **Jiang, Z. & Naudé, P. (2007).** An Examination of the Factors Influencing Software Development Effort. *Journal of Computer, Information, and Systems Science, and Engineering.*
14. **Kemerer, C. F. & Patrick, M. W. (1992).** Staffing Factors in Software Cost Estimation Models. Editor **Keyes, J**., *The Handbook of Software Engineering Productivity*, McGrawHill.
15. **Jarillo, P. I., Enríquez, C., & Sánchez, R. A. (2015).** Identificación del Factor Humano en el Seguimiento de Procesos de Software en un Medio Ambiente Universitario. *Computación y Sistemas,* Vol. 19, No. 3, 2015, pp. 577–588.
16. **Pressman, R.S. (2010).** *Software Engineering. A Practioner's Approach.* Seventh Edition, McGraw Hill.
17. **ISO/IEC FDIS 12207 (2007).** *Systems and software engineering Software life cycle processes.*
18. **Ziv, H. & Richardson, D.J. (1996).** The Uncertainty Principle in Software Engineering. *Submitted to ICSE'97,* 19th International Conference on Software Engineering.
19. **Cartensen, B. (2010).** *Comparing Clinical Measurement Methods: A practical guide*. Wiley.
20. **Chris, B. (2008).** *Introductory Econometrics for Finance.* Cambridge University Press.
21. **Goos, P. & Meintrup, D. (2015).** *Statistics with JMP: Graphs, Descriptive Statistics and Probability.* John Wiley & Sons.
22. **Satchell, S. & Knight, J. (2011).** *Forecasting Volatility in the Financial Markets.* Butterworth-Heinemann.
23. **Chitode, J.S. (2009).** *Information Theory & Coding*. Technical Publications.
24. **Jazwinski, A.H. (2007).** *Stochastic Processes and Filtering Theory.* Courier Corporation.
25. **Schulze, H. & Lueders, Ch. (2005).** *Theory and Applications of OFDM and CDMA: Wideband Wireless Communications.* John Wiley & Sons.
26. **Ruppert, D. (2004).** *Statistics and Finance: An Introduction.* Springer Science & Business Media.
27. **Fan, J. & Yao, Q. (2008).** *Nonlinear Time Series: Nonparametric and Parametric Methods.* Springer Science & Business Media.
28. **Sayed, A. H. (2008).** *Adaptive Filters.* Wiley Online Library.
29. **Helsel, D. R. & Hirsch, R. M. (2002).** *Statistical Methods in Water Resources. Science for changing world.*

30. **Yan, X. (2009).** *Linear Regression Analysis: Theory and Computing.* World Scientific.
31. **Briand, L. (2001).** *Resource Estimation in Software Engineering.* Wiley, Online Library.
32. **López, C., Yáñez, C., Gutiérrez, A., & Riverón, E. (2007).** Adequacy Checking of Personal Software Development Effort Estimation Models Based upon Fuzzy Logic: A Replicated Experiment. *Computación y Sistemas*.
33. **Popović, J. & Bojić, D. (2012).** *A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle.*
34. **Pfleeger, S. L. (1991).** Software Engineering. *The Production of Quality Software,* Maxwell Macmillan, Second edition.
35. **Pressman, R. S. (2005).** *Software Engineering. A Practitioner's Approach*. McGraw Hill.
36. **Fernández, S., Cordero, J. M., & Córdoba, A. (2002).** *Estadística descriptiva.* ESIC Editorial.
37. **Abbas, S. A., Liao, X., Rehman, A. U., Azam, A., & Abdullah, M. I. (2012).** Cost Estimation: A Survey of Well-known Historic Cost Estimation Techniques. *Journal of Emerging Trends in Computing and Information Sciences*.
38. **Jayne, Ch., Yue, S., & Iliadis, L. (2012).** Engineering Applications of Neural Networks: *13th International Conference EANN 2012*, London, UK, September 20–23, Springer.
39. **Myrtveit, I., Stensrud, E., & Shepperd, M. (2005).** Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Transactions on Software Engineering*. DOI:10.1109/TSE.2005.58
40. **Lokan, Ch. & Mendes, E. (2009).** Applying Moving Windows to Software Effort Estimation. *Third International Symposium on Empirical Software Engineering and Measurement*.
41. **Macfie, B. P. & Nufrio, P. M. (2006).** *Applied Statistics for Public Policy.* M.E. Sharpe.
42. **Woolson, R. F. & Clarke, W. R. (2011).** *Statistical Methods for the Analysis of Biomedical Data*. John Wiley & Sons.
43. **Schuenemeyer, J. & Drew, L. (2011).** *Statistics for Earth and Environmental Scientists.* John Wiley & Sons.
44. **Agarwal, B. L. (2007).** *Programmed Statistics (Question-Answers).* New Age International.
45. **Whaley, R. E. (2007).** *Derivatives: Markets, Valuation, and Risk Management*. John Wiley & Sons.
46. **Pedace, R. (2013).** *Econometrics for Dummies.* John Wiley & Sons.
47. **Wooldridge, J. (2008).** *Introductory Econometrics: A Modern Approach.* Cengage Learning.
48. **Krishnan, V. (2015).** *Probability and Random Processes*. John Wiley & Sons.
49. **Bernstein, S. & Bernstein, R. (1999).** *Theory and Problems of Elements of Statistics II. Inferential Statistics.* Schaum's Outline Series, McGraw-Hill.

Ma. del Refugio Ofelia Luna Sandoval received a M.Sc. degree in Computer Science from the Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), México, in 2011. She is currently a doctoral student at the CENIDET and her current research interests are applications of computational intelligence techniques to Software Engineering.

José Ruiz Ascencio received the B. Sc. degree in physics from (UNAM) in 1971, a M.Sc. degree from Stanford in 1973 and the D. Phil. degree from the University of Sussex in 1989. He was researcher at the Institute of Applied Mathematics, IIMAS-UNAM, full-time lecturer at the Autonomous University of Barcelona, automation project leader for Allen-Bradley, researcher at the Instituto Tecnológico de Monterrey, and invited scholar at McGill University's Center for Intelligent Machines (2003 and 2010). He joined CENIDET in 1995, where he is a member of the Artificial Intelligence Group. His current interests are machine vision and computational intelligence.

Artículo recibido el 12/05/2016; aceptado el 28/06/2016.
Autor de correspondencia es Ma. del Refugio Ofelia Luna Sandoval.