Revista Cubana de Ciencias Informáticas
ISSN: 1994-156
ISSN: 2227-1899
Editorial Ediciones Futuro

Ferreira Lorenzo, Gheisa Lucía; Gálvez Lío, Daniel;
Quintero Domínguez, Luis Alberto; Antón Vargas, Jarvin

Estimación del esfuerzo en proyectos de software utilizando técnicas de inteligencia artificial

Revista Cubana de Ciencias Informáticas, vol. 8, núm. 4, 2014, Octubre-Diciembre, pp. 1-20
Editorial Ediciones Futuro

Disponible en: https://www.redalyc.org/articulo.oa?id=378368201001

Revista Cubana de Ciencias Informáticas
ISSN: 2227-1899 | RNPS: 2301
Vol. 8, No. 4, Octubre-Diciembre, 2014
Págs. 1-20
http://rcci.uci.cu

Tipo de artículo: Artículo de revisión
Temática: Ingeniería y gestión de software
Recibido: 4/03/2014 | Aceptado: 17/06/2014

Estimación del esfuerzo en proyectos de software utilizando técnicas de inteligencia artificial

Effort estimation of software projects using artificial intelligence techniques

Gheisa Lucía Ferreira Lorenzo[1*], Daniel Gálvez Lío[1], Luis Alberto Quintero Domínguez[1], Jarvin Antón Vargas[1]

[1] Universidad Central "Marta Abreu" de Las Villas. Carretera a Camajuaní km 5 ½. Santa Clara. Villa Clara. {dgalvez, lqdominguez, janton}@uclv.edu.cu

[*] Autor para correspondencia: gheisa@uclv.edu.cu

Resumen

Los modelos algorítmicos de estimación de costo y esfuerzo, basados en el análisis regresivo de datos históricos abundan en la literatura especializada. Entre los más populares se encuentran COCOMO, SLIM, Puntos de Función. No obstante, desde los años 90, los modelos basados en técnicas de Inteligencia Artificial, fundamentalmente en técnicas de Aprendizaje Automático, han sido utilizados para mejorar la precisión de las estimaciones. Estos modelos se fundamentan en el uso de datos recogidos en proyectos anteriores en los que se realizaron estimaciones y la aplicación de diferentes técnicas de extracción de conocimiento, con el objetivo de realizar estimaciones de manera más eficiente, eficaz y, si fuera posible, con mayor precisión. El objetivo de este artículo consiste en presentar el análisis de algunas de estas técnicas, y cómo ellas han sido aplicadas en la estimación del esfuerzo en proyectos de software.

Palabras clave: estimación de esfuerzo, gestión de proyectos, ingeniería de software, inteligencia artificial.

Abstract

Algorithmic models of cost and effort estimation based on regression analysis of historical data abound in the literature. Among the most popular models are COCOMO, SLIM, Function Points. However, since the 90's, models based on Artificial Intelligence techniques, mainly in Machine Learning techniques have been used to improve the accuracy of the estimates. These models are based on the use of data collected in previous projects in which estimates

were made and the application of different knowledge extraction techniques, in order to make estimates more efficient, effective and, if possible, more precise. The aim of this paper is to present an analysis of some of these techniques and how they have been applied in estimating the effort in software projects.
Keywords: *artificial intelligence, effort estimation, project management, software engineering.*

Introducción

La estimación del esfuerzo de producción es una necesidad en todas las áreas de la industria y una actividad imprescindible para el estudio de viabilidad de los proyectos. Desde el momento en que las empresas comenzaron a considerar las aplicaciones informáticas como productos industriales, aparecieron algunos problemas fundamentales que resolver: el cumplimiento de los plazos de entrega dentro de costos establecidos manteniendo niveles de calidad; así como poder realizar un seguimiento y control de la evolución de los proyectos. Por lo que el establecimiento de métodos que permitan determinar y, posteriormente, alcanzar estos objetivos de una forma lo más real y exacta posible ha sido un factor cada vez más importante para la Ingeniería Informática en su conjunto. Dichos métodos se han fundamentado en conocimientos adquiridos por distintas disciplinas de esta ciencia, desde la Ingeniería del Software hasta la Inteligencia Artificial.

Desde los años sesenta hasta hoy día se han revisado y publicado numerosos modelos de estimación en Ingeniería del Software, ((Kemerer, 1987), (Finnie *et al.*, 1997), (Boehm, 2000), (Jørgensen and Shepperd, 2007)). También se han propuesto distintas clasificaciones de los mismos en estudios doctorales realizados, sobre la base de diferentes criterios ((Crespo Yáñez, 2003), (Salvetto de León, 2006), entre otros). Por ejemplo, una clasificación actualizada se establece en este último trabajo donde los modelos para la estimación aparecen clasificados en: modelos matemáticos paramétricos, estimación basada en analogías, modelos dinámicos y modelos basados en técnicas de Inteligencia Artificial (IA). Por el interés que resulta la aplicación de las técnicas de IA en problemas de Ingeniería de Software, los modelos basados en estas técnicas constituyen el objetivo principal del estudio presentado en el artículo. Dicho estudio incorpora aquellos modelos para la estimación de esfuerzo de proyectos de software que con más frecuencia fueron reportados en la literatura científica analizada, estos son:

- Razonamiento Basado en Casos (RBC)
- Algoritmos Genéticos (AG)
- Programación Genética (PG)

- Support Vector Regression (SVR)
- Redes Neuronales Artificiales (RNA)
- Sistemas de Inferencia Borrosos (SIB) y Lógica Difusa (LD)
- Árboles de Decisión (AD)

Desarrollo

Razonamiento Basado en Casos (RBC)

Según (Mukhopadhyay *et al.,* 1992), los modelos matemáticos paramétricos existentes han fallado en el intento de producir estimaciones precisas del esfuerzo de desarrollo del software, afirmándose que las estimaciones analíticas no son suficientes. En su investigación, trata de resolver el problema de la estimación utilizando RBC cuyo principio fundamental es el razonamiento por analogías. La estimación por analogía es el proceso de encontrar uno o más proyectos similares a otro que va a ser estimado y entonces derivar el estimado, desde los valores de esos proyectos. Según (Jørgensen *et al.,* 2003) este tipo de estimación puede ser desarrollada como:

- Estimación puramente por el experto (la "base de datos" de los proyectos previos está en la cabeza del experto).
- Estimación por el experto, informalmente soportada por una base de datos conteniendo información acerca de proyectos concluidos.
- Estimación basada en el uso de algoritmos de clusterización para encontrar proyectos similares. Modelos de estimación de este tipo pueden encontrarse en (Briand *et al.,* 1992), (Shepperd and Schofield, 1997) y (Walkerden and Jeffery, 1999).

En esta última clasificación se ubica el RBC, ratificándose en (Sun-Jen and Nan-Hsing, 2006) que es un enfoque de estimación por analogía, y un proceso cíclico compuesto fundamentalmente por cuatro estados:

1. Recuperar el proyecto más similar (caso).
2. Reusar este proyecto para intentar resolver el problema.
3. Revisar la solución sugerida si es necesario.
4. Retener la solución y el nuevo problema como un nuevo proyecto.

En el modelo computacional Estor propuesto en (Mukhopadhyay *et al.,* 1992) se aplican estos procesos y se obtiene el conocimiento específico del dominio de una base de conocimientos que incluye la representación de los casos (es

decir, proyectos de software), el conocimiento para seleccionar un caso análogo apropiado para cada caso objetivo (base de reglas) y el conocimiento para ajustar la estimación basado en la interacción entre la representación del caso fuente y el caso objetivo (heurística de selección).

En Estor, se tomaron en consideración las estimaciones realizadas a diez proyectos de software terminados (según un estudio que aparece en (Kemerer, 1987)), contemplándose 37 factores del proyecto además del esfuerzo real de desarrollo. Se consideró como proyecto objetivo o "experto de referencia" para la construcción de Estor, uno de los proyectos de una compañía con varios años de experiencia en la estimación de software. Para ilustrar en la práctica este enfoque, se tomó un proyecto que tardó 277 personas-mes en completarse y se realizaron cuatro tipos de estimaciones: por juicio del experto, por Estor, utilizando COCOMO y por Puntos de Función (PF) a 15 proyectos de software. La estimación del experto aportó un valor de 250 personas-mes, la solución por Estor fue de 287 personas-mes. COCOMO estimó 1238.6 personas-mes y con PF los resultados fueron de 344.3 personas-mes.

La comparación del rendimiento fue realizada en términos de precisión y consistencia. La medida de precisión utilizada fue el promedio de la magnitud del error relativo (MMRE) (Srinivasan and Fisher, 1995). La consistencia fue medida como el coeficiente de correlación entre el esfuerzo real y el esfuerzo estimado a través de una muestra de problemas que en este caso resultó ser de 15 proyectos (10 proyectos fueron utilizados como conjunto de entrenamiento y los 5 restantes como prueba). La magnitud del error relativo de Estor (52.79 %) no fue tan buena como la del experto humano, pero si fue superior a la correspondiente a COCOMO y PF. Lo mismo ocurrió con el coeficiente de correlación, confirmándose que los resultados del experto y de Estor son igualmente consistentes y a su vez más consistentes que los que se obtuvieron con los modelos algorítmicos.

Es de esperar que los cuatro métodos empleados en la estimación no sean igualmente precisos y consistentes debido a sus características intrínsecas. Sin embargo, para Estor y para el RBC en general, quedan cuestiones por analizar como:

- ¿Qué atributos o características del software el experto debe utilizar, que no necesariamente estén restringidos a las entradas utilizadas en los métodos como COCOMO y PF?
- ¿Qué reglas debe utilizar el experto?
- ¿Se deben considerar todos los atributos o una elección de los mismos?
- ¿Cuáles atributos son más importantes para ajustar el conocimiento?

- ¿Qué cantidad de casos es la apropiada para garantizar un buen rendimiento del modelo?

Algunas respuestas a las interrogantes anteriores son consideradas en otra aplicación de RBC en la estimación de esfuerzo en proyectos de software analizada en (Finnie *et al.*, 1997). En este caso la solución por RBC es comparada con otros enfoques: PF, RNA y modelos de regresión. De manera similar a la discusión anterior, el rendimiento se determina en términos de precisión a partir del MMRE. Aquí se dispone de un número mayor de proyectos (299 en total), 249 de ellos tomados como "casos de entrenamiento" y los 50 restantes como casos de prueba para evaluar la calidad del modelo. Esto puede indicar que la ampliación del tamaño de la muestra puede ser beneficiosa para garantizar mejores resultados del modelo. Los datos de los proyectos fueron tomados de un estudio estadístico referenciado en (Desharnais and Al., 1990).

El concepto clave utilizado para determinar la adaptación de los casos fue el de identificar factores o características del software que contribuyeran a mostrar diferencias significativas en la productividad entre los casos. Los factores o características generales del sistema (cgs) provienen de la estimación por PF y son evaluados de 0 a 5. Con estos factores, el tamaño del software (medido en PF sin ajustar) y la proporción de entradas, salidas, peticiones, archivos internos y externos del total de PF sin ajustar, se desarrolló un sistema de RBC que para los 50 casos de prueba tuvo un MMRE de 0.362 con un 80% de casos estimados con error menor que 0.5. Nótese que tanto el tamaño de la base de casos así como los atributos utilizados aportan diferentes resultados y en este último caso, mejores, en la estimación. Por otra parte el RBC permite el desarrollo de una base de casos dinámica con los datos de nuevos proyectos que se incorporan automáticamente a la misma y se convierten en disponibles para el análisis de una nueva solución.

En (Burgess and Lefley, 2001) también se utiliza el RBC para la estimación y se compara con otras técnicas: RNA y PG. En esa investigación se utilizan nueve variables independientes tomadas del conocido repositorio de proyectos de Desharnais de finales de los '80 y una variable dependiente, el esfuerzo, analizándose 81 proyectos. Debe destacarse que en la evaluación de los resultados se consideran otras medidas de precisión como el cuadrado del error medio ajustado y la magnitud del error relativo medio balanceado. También se utilizan medidas cualitativas como recursos empleados (tiempo y memoria para el entrenamiento y la consulta), transparencia de la solución o decisión, generalidad, robustez, rango de convergencia y predicción más allá del aprendizaje del conjunto de datos. Los resultados son favorables y se insiste en las bondades del RBC dadas por la posibilidad de proporcionar un orden a los

proyectos por grados de similaridad con el proyecto objetivo. Esto suministra un valor explicativo significativo cuando se consideran los usuarios finales.

Referencias al RBC igualmente pueden encontrarse en (Patnaik *et al.,* 2004) y (Adekile, 2008). Se ha utilizado también este enfoque como método de comparación en (MacDonell and Shepperd, 2003), (Sun-Jen and Nan-Hsing, 2006) y (Sun-Jen *et al.*, 2008).

De acuerdo con (Wen *et al.*, 2012) puede afirmarse que el RBC es la técnica de aprendizaje automático que más se ha reportado en la literatura especializada. Más recientemente se han encontrado referencias donde el RBC se combina con enfoques heurísticos como la optimización basada en enjambres de partículas, para la optimización de los pesos en la función de similaridad con resultados satisfactorios (Wu *et al.*, 2010). También se evalúa el grado de similaridad entre dos proyectos utilizando la lógica difusa como aparece en (Idri *et al.*, 2000).

Algoritmos Genéticos (AG)

Los AG fueron desarrollados como una técnica alternativa, basada en la teoría de la selección natural, para afrontar problemas generales de optimización con largos espacios de búsqueda. Tienen la ventaja de que no necesitan ningún conocimiento anterior, opinión de un experto o lógica relacionada con el problema en particular a resolver (Burgess and Lefley, 2001). Los AG ocasionalmente pueden producir la solución óptima, pero para la mayoría de los problemas con grandes espacios de búsqueda, una buena aproximación al óptimo es la salida más probable.

En la bibliografía revisada relativa a la gestión de proyectos y en particular a la estimación del esfuerzo, los AG aparecen en combinación con otras técnicas. Por ejemplo, en (Oliveira *et al.*, 2010) se propone el uso de las técnicas *Support Vector Machine* (SVM), red neuronal MLP y *model trees* M5P tratadas posteriormente en este artículo. Los AG, en este caso, han sido utilizados para seleccionar el subconjunto de rasgos de entrada y los parámetros óptimos de las técnicas de aprendizaje automático. Al evaluar los resultados, los autores concluyen que las variantes de los métodos basadas en AG fueron capaces de mejorar la precisión de las tres técnicas de aprendizaje automático consideradas. Además, el uso de AG logró una significativa reducción del número de atributos de entrada para dos de las seis bases de datos de proyectos analizadas.

En (Huang and Chiu, 2006) se utilizan los AG combinados con RBC. En (Huang *et al.*, 2008) también se aplican AG. En esta ocasión se combinan con Análisis Relacional Gris (*Grey Relational Analisys* - GRA). Un modelo de estimación de esfuerzo de software utilizando GRA identifica uno o más proyectos históricos que son similares al proyecto que está siendo estimado y deriva una estimación para él. El propósito de utilizar AG en este enfoque es para mejorar u optimizar los pesos de cada uno de los rasgos involucrados en las medidas de similitud en GRA. En este estudio se utilizaron dos bases de datos de proyectos: COCOMO y Albrecht. Se obtuvieron resultados superiores en comparación con modelos de RNA y RBC, aunque se reconoce que la integración GRA y AG para la estimación de esfuerzo presenta una complejidad superior del modelo que puede afectar su uso en la práctica.

Programación Genética (PG)

PG es una extensión de los AG que elimina la restricción de que un cromosoma sea una cadena binaria de tamaño restringido. Generalmente en PG el cromosoma es un tipo de programa que es ejecutado para obtener los resultados requeridos. Una forma simple de estos programas es un árbol binario que contiene operandos y operadores, es decir cada solución es una expresión algebraica que puede ser evaluada (Burgess and Lefley, 2001).

En (Afzal and Torkar, 2011) se menciona que (Dolado et. al, 1998) utiliza PG y otras técnicas de aprendizaje automático como RNA para estimar esfuerzo, tomando como variables independientes las líneas de código y los puntos de función. Para esto emplea la información de cinco bases de proyectos, concluyéndose que RNA y PG mostraron ser métodos mucho más flexibles en comparación con las técnicas utilizadas. En (Burgess and Lefley, 2001) se muestra una investigación donde se utiliza PG para la estimación de esfuerzo. Aquí se realiza una comparación de los resultados obtenidos con PG y los obtenidos con otros modelos basados en RNA y RBC. En este trabajo se llega a la conclusión de que el sistema construido utilizando PG puede mejorar la exactitud de la predicción con respecto a las otras técnicas. Sin embargo, se resalta la necesidad de realizar un mayor trabajo para mejorar la consistencia de la estimación y para determinar cuáles son las medidas más apropiadas para la selección del mejor modelo en la práctica. Además se destaca la complejidad de su configuración e interpretación.

En (Afzal and Torkar, 2011) se hace una revisión de la utilización de PG en la Ingeniería de Software. Específicamente en la estimación de esfuerzo se precisa que los resultados no son concluyentes para decir que PG es un método efectivo, sobre todo debido a que cuando se optimiza una medida de precisión se degradan las otras.

Support Vector Regression (SVR)

El algoritmo Support Vector (SV) es una generalización no lineal del algoritmo Generalized Portrait desarrollado en Rusia en los años sesenta. En poco tiempo los clasificadores SV se volvieron competitivos con los mejores sistemas de Reconocimiento Óptico de Caracteres, pero también en regresión, tomando el nombre de SVR, fueron obtenidos excelentes resultados rápidamente (Smola and Scholkopf, 2004).

SVR ha superado el comportamiento de otras técnicas más tradicionales en la solución de varios problemas. Por esto en (Oliveira, 2006) se propone el uso de SVR para estimar el esfuerzo en el desarrollo de proyectos de software. Para determinar la efectividad de la estimación en dicha investigación, se realizó una comparación de los resultados obtenidos con SVR y los obtenidos con regresión lineal y la red neuronal Radial Basis Function (RBF), sobre la base de proyectos de la NASA. Luego de analizados los resultados se llegó a la conclusión de que SVR logró mejorar la precisión de la estimación por encima de los otros dos métodos.

En (Corazza *et al.,* 2011a) se analiza el empleo de SVR para la estimación de esfuerzo en el desarrollo de proyectos Web y se comparan varias configuraciones de SVR para determinar la que brinda el mejor comportamiento. Aquí se confrontan los resultados obtenidos utilizando dos tipos de funciones kernel: Polinomial y RBF. Además, para determinar si la aplicación de una de estas transformaciones kernel puede mejorar la exactitud de la predicción, también se consideró SVR sin una transformación kernel, conocida como kernel Lineal. Los resultados obtenidos con SVR fueron comparados con los brindados por RBC, que es una de las técnicas más utilizadas para la estimación de esfuerzo en proyectos de software, para determinar si SVR lograba un mejor desempeño. Después de examinar los resultados se determinó que el uso del kernel RBF brindó el mejor resultado. Luego de confrontar el comportamiento de esta configuración de SVR con el obtenido por RBC, se llegó a la conclusión de que SVR fue significativamente superior en la precisión de la estimación. Otro resultado interesante, también para proyectos Web, puede encontrarse en (Corazza *et al.,* 2009c) donde se utilizaron seis configuraciones diferentes para cada kernel, con dos conjuntos de entrenamiento cada uno de 130 proyectos de software.

Seleccionar la mejor combinación de valores de los parámetros en SVR, puede tener una gran influencia en su efectividad. Sin embargo no hay líneas directrices generales disponibles para seleccionar estos parámetros, algo que también depende de las características de los datos que son usados. Esto motivó el trabajo descrito en (Corazza *et al.*, 2011b), donde se propone el uso de la metaheurística Búsqueda Tabú (Tabu Search - TS) para obtener los valores

óptimos de los mismos. Aquí se diseñó TS para buscar los parámetros tanto del algoritmo SVR como de la función kernel empleada, en este caso RBF. Fueron tomadas en consideración varias medidas de comparación para evaluar ambos aspectos, la efectividad de TS para definir los parámetros de SVR y la exactitud de la predicción de la metodología propuesta con respecto a otras técnicas de estimación de esfuerzo ampliamente utilizadas, como RBC, SVR (sin uso de TS), etc. En el mencionado estudio se determinó que el uso de TS permite obtener los parámetros idóneos para ejecutar SVR. Además se concluyó que la combinación de TS y SVR superó significativamente el comportamiento de todas las demás técnicas con las que fue comparada.

SVR ha seguido utilizándose como técnica para estimar de manera eficiente y precisa el esfuerzo en proyectos de desarrollo de software. Un modelo de inferencia evolutivo de SVR es utilizado en (Jui-Sheng *et al.,* 2012) para la estimación de personas-horas en proyectos de desarrollo de sistemas ERP (*Enterprise Resource Planning*). Aquí, el modelo propuesto es un modelo de inteligencia híbrido que integra una máquina de soporte vectorial (que proporciona el aprendizaje) con un algoritmo genético (que minimiza los errores). Los resultados analíticos confirman que el modelo híbrido propuesto, proporciona resultados para el esfuerzo de desarrollo de software ERP en etapas tempranas, que resultan más precisos que los obtenidos con redes neuronales artificiales y SVR.

Redes Neuronales Artificiales (RNA)

Según (Finnie *et al.*, 1997), las Redes Neuronales Artificiales (RNA) son reconocidas por sus habilidades para proveer buenos resultados cuando están lidiando con problemas donde existen complejas interacciones entre las entradas y las salidas y donde los datos de entrada son distorsionados por altos niveles de ruido, y el potencial para predecir con precisión es muy bueno. El modelo está diseñado para capturar las relaciones causales entre la variable dependiente y las variables independientes. En el caso de estimación que se plantea en este trabajo, entre la variable estimación del esfuerzo de un proyecto de software y las variables de entrada que han sido determinadas como características de los proyectos. Las RNA pueden conducir a desarrollar modelos que pueden ser precisos a través de un ejemplo dado, pero fallan cuando las condiciones cambian. Carecen de la capacidad de explicación o justificación de la estimación realizada y no proveen un ambiente para una adaptación directa por el usuario de los datos, es decir una recalibración del modelo. Para poder incorporar un nuevo caso al conjunto de casos válidos del propio modelo, necesitan ser reentrenadas, generando un modelo nuevo.

Existen varias investigaciones que han abordado esta técnica, desarrollando diversas arquitecturas de RNA para estimar el esfuerzo de desarrollo de proyectos de software como en (Finnie *et al.*, 1997), (Briand and Weiczorek, 2002), (Marza and Teshnehlab, 2009) entre otros. En (Briand and Weiczorek, 2002), se hace un estudio donde se evalúa el funcionamiento de las RNA con otros modelos como los de regresión, inducción de reglas y RBC. Aquí se evidencia que las RNA presentan una exactitud superior a los modelos de regresión y una exactitud muy cercana a la de los modelos basados en RBC. Son muy buenas cuando existen valores atípicos en los datos. Se tomaron como variables de entrada: el número de reportes, el número de pantallas, el número de creaciones, el número de actualizaciones, el ambiente de programación, las líneas de código, 14 características generales del sistema y las variables utilizadas en los modelos de análisis por PF. La variable de salida fue la estimación del esfuerzo.

El estudio realizado en (Briand and Weiczorek, 2002), se basó en un conjunto de 148 proyectos de software terminados. Se realizó una selección de las variables de entrada partiendo de un conjunto de 36 variables, que se disminuyó a siete luego de un análisis de frecuencia y un análisis de regresión estadística. Se propuso el análisis a tres topologías distintas de RNA y se evaluó la precisión de estos modelos con modelos clásicos de regresión y el juicio de un experto, arrojando como resultado que el modelo de siete variables fue el que mejor se comportó. En este estudio se evidencia que la precisión de la estimación realizada por un modelo basado en RNA depende mucho de las variables de entrada. Otro enfoque que hace uso de una red neuronal para calcular valores de parámetros de desarrollo de software (tamaño o esfuerzo), se puede encontrar en (Pendharkar, 2010). En este caso, la red neuronal es probabilística.

Una combinación novedosa aparece en (Attarzadeh *et al.*, 2012), donde se incorpora una RNA al clásico modelo constructivo de costo (COCOMO), para presentar un modelo ANN-COCOMO II que proporcione mayor precisión en las estimaciones de software desde etapas tempranas del desarrollo. En este modelo se utiliza la habilidad de aprendizaje de la red neuronal para calibrar los atributos del software, manteniéndose los méritos del modelo COCOMO. Su evaluación en 156 conjuntos de datos de proyectos de dos repositorios, arrojó un 8.36% de mejora en la precisión de la estimación, cuando fue comparado con el original COCOMO II.

Los modelos de RNA también se han combinado con enfoques de la lógica difusa para la estimación del esfuerzo en proyectos de software, donde el conocimiento se describe mediante reglas. En (Marza and Teshnehlab, 2009), se presenta esta combinación, donde las RNA tienen dificultades en el uso de reglas del conocimiento y por tener una

estructura de caja negra, dificultan la extracción del mismo, recurriéndose a los Sistemas de Inferencia Borrosos (SIB) tratados en el próximo epígrafe, que pueden incorporar reglas base y permiten una fácil interpretación e implementación, aunque no pueden aprender el conocimiento lingüístico, estos sistemas se abordan en el apartado siguiente.

Como se evidencia en múltiples estudios ((Sikka et al, 2010), (Barcelos *et al.*, 2006), (Zeng and Rine, 2004), (Jun and Lee, 2001)), los modelos basados en RNA son capaces de dar un adecuado modelo de estimación de esfuerzo. Su comportamiento en gran medida depende de los datos que se empleen en su entrenamiento y el grado de disponibilidad de los datos adecuados del proyecto, determinarán el grado en el que el modelo de estimación pueda ser desarrollado. Más recientemente, en (García et al, 2011) se ha propuesto una metodología de optimización de modelos de RNA para la estimación de esfuerzo del desarrollo de software, que guía en la búsqueda del mejor modelo neuronal para el problema estudiado, a fin de mejorar el rendimiento, tanto en tiempo como en precisión. Esta metodología, aplicada en un inicio a una red neuronal de tres capas, se continúa perfeccionando, aplicándose a un conjunto más amplio de arquitecturas de redes neuronales.

Sistemas de Inferencia Borrosos (SIB) y Lógica Difusa (LD)

Los SIB hacen uso de la lógica difusa, que según (Raynor, 1999) es un sistema basado en la manipulación de conjuntos borrosos. En estos sistemas, el conocimiento se describe mediante reglas y los términos lingüísticos se modelan mediante conjuntos borrosos, utilizándose modelos de inferencias como los de Mandani, Sugeno y Tsukamoto. Proporcionan una técnica para tratar la imprecisión, la vaguedad y el desconocimiento de los datos en un dominio determinado (Bello *et al.*, 2002).

Según (Cuauhtemoc, 2011), dos consideraciones podrían justificar la decisión de implementar un modelo *fuzzy* en la estimación del esfuerzo de desarrollo de un software: primero, es imposible desarrollar un modelo matemático preciso del dominio y segundo, las métricas sólo producen estimaciones de la complejidad real. De acuerdo con estos criterios, la formulación de un pequeño conjunto de reglas naturales describiendo la interrelación entre las métricas del software y la estimación del esfuerzo, podría fácilmente mostrar su intrínseca y amplia correlación. En ese trabajo se crea un modelo de estimación partiendo de la correlación que existe entre el código nuevo y cambiado para la implementación de un software, y el esfuerzo. Se tiene en cuenta también la correlación entre el reuso de código y el esfuerzo, llegándose a un conjunto de dos reglas. Las funciones de membresía consideradas en el modelo fueron

triangulares debido a que se ha demostrado que proveen mayor exactitud por encima de otras como la Gaussiana y la Trapezoidal. En el estudio se demuestra que no existen diferencias significativas entre la estimación arrojada por el modelo borroso y un modelo de regresión lineal aplicado al mismo conjunto de datos.

Una desarrollo similar se realiza en (García *et al.,* 2013) donde se utilizan los modelos de Mamdani y Takagi-Sugeno para comparar la precisión de la estimación con un modelo de regresión lineal. En este caso los resultados mostraron que el sistema *fuzzy* Takagi-Sugeno fue más preciso que el sistema Mamdani y el modelo de regresión lineal para la estimación del esfuerzo de desarrollo de software, en proyectos donde el esfuerzo fue mayor o igual que 100 horas-hombre.

Un inconveniente de los modelos borrosos es que necesitan el ajuste por parte del experto en algunos estados de su heurística de generación. Sin embargo, los modelos estáticos presentan una generación con un procedimiento bien definido. No obstante, la combinación de ambos puede generar resultados superiores a cada uno por separado. Un ejemplo de esta combinación para la estimación del esfuerzo de desarrollo de un software se puede ver en (Marza and Seyyedi, 2009) donde se combinan modelos de regresión con conceptos borrosos para el manejo de datos discretos, creándose un modelo borroso de regresión.

Otros estudios han mostrado que los modelos basados en la lógica difusa consiguen buenos resultados, siendo solamente superados por los modelos de redes neuronales con un número mayor de variables de entrada. Un casamiento entre estas dos técnicas es llamado Neuro-Fuzzy, término introducido en la estimación del costo en (Hodgkinson and Garratt, 1999). Estos sistemas pueden tomar los atributos lingüísticos de los sistemas difusos y combinarlos con los atributos de aprendizaje y modelado de las redes neuronales, para producir sistemas transparentes y adaptativos.

El uso de estos sistemas Neuro-Fuzzy en la estimación, puede encontrarse en (Sandhu *et al.,* 2008) donde un sistema de este tipo es utilizado para aproximar la función no lineal del esfuerzo con mayor precisión, sobre otros modelos algorítmicos conocidos como Halstead, Walston-Felix, Bailey-Basili and Doty. Los resultados muestran que el sistema Neuro-Fuzzy tiene mejores resultados, afirmándose que el modelo propuesto puede ser utilizado para la estimación del esfuerzo en cualquier tipo de proyecto.

La lógica difusa también ha sido ampliamente usada con otras técnicas de aprendizaje automático para mejorar el rendimiento de un modelo, pre-procesando las entradas de dichos modelos (Wen *et al.*, 2012, Attarzadeh and Hock, 2010), donde a partir de un modelo desarrollado en tres pasos: "fuzzificación" de las variables de entrada propuestas en COCOMO II (tamaño, multiplicadores de costo y factores de escala), aplicación de un SIB y "defuzzificación" de la variable de salida, se obtiene como resultado el esfuerzo.

Un enfoque similar es el que aparece en (Huang *et al.*, 2007) donde se utiliza una RNA combinada con la lógica difusa para mejorar el desempeño del modelo COCOMO, aprovechando las características ventajosas del enfoque Neuro-Fuzzy de habilidad de aprendizaje y mejor interpretación de la solución. Otro enfoque, en este caso aplicando las características de COCOMO'81, se puede localizar en (Idri *et al.*, 2000).

Árboles de Decisión (AD)

AD es una técnica que construye un modelo en forma de árbol, dividiendo recursivamente el conjunto de datos hasta que un criterio de parada es satisfecho. Esta división se realiza con el propósito de alcanzar la máxima homogeneidad posible, relativa a la variable de salida o dependiente, entre los ejemplos que alcanzan el nodo. Todos los nodos en el árbol, menos los terminales (también llamados hojas), especifican una condición basada en una de las variables que tienen influencia en la variable dependiente. Luego de que el árbol es generado, el mismo se puede usar para realizar predicciones siguiendo un camino a través del árbol de acuerdo con los valores específicos de las variables del nuevo caso (Idri and Elyassami, 2011). Los AD han demostrado su superioridad en términos de exactitud de la predicción en varios campos. Los algoritmos más usados para la construcción de AD son ID3, C4.5 y Árbol de Clasificación y Regresión (Clasification And Regression Tree – CART). Entre las ventajas fundamentales del uso de la estimación mediante AD se encuentran:

- Pueden ser considerados como "cajas blancas", debido a que su funcionamiento es fácil de explicar.
- Realizan selección de rasgos, lo que evita la utilización de otras técnicas para determinar los rasgos que verdaderamente influyen en el cálculo del esfuerzo en los modelos de estimación.

Los Árboles de Regresión (AR) son un tipo especial de AD desarrollado para tareas de regresión. En este tipo específico de AD la elección de cada nodo está usualmente guiada por el criterio del error cuadrado mínimo (Braga *et al.*, 2007).

De igual manera que los AR son un tipo especial de AD, los *model trees* son un tipo especial de AR. No obstante la principal diferencia entre AR y model trees es que las hojas en los AR presentan un valor numérico mientras que las hojas en los *model trees* tienen una función lineal (Braga *et al.*, 2007).

Los AR constituyen una de las técnicas que más han sido usadas para la estimación de esfuerzo de desarrollo en proyectos de software. En el estudio que se muestra en (Briand *et al.*, 1999) se examinan los resultados obtenidos con varias técnicas como son: Regresión Ordinaria de Mínimos Cuadrados (Ordinary Least-Squares Regression -OLS), Análisis de Varianza progresivo (Analysis of Variance - ANOVA) para bases de datos no balanceadas, RBC, AR (algoritmo CART) y la combinación de CART con regresión OLS y RBC. Luego de comparar los resultados obtenidos se determinó que las técnicas OLS y ANOVA brindan resultados significativamente superiores a los obtenidos con las demás técnicas. También se resalta la facilidad de interpretación y uso de CART para la construcción de un modelo de estimación.

En (Briand and Wust, 2001) se utiliza la regresión Poisson y AR, así como su combinación, para construir modelos de predicción de esfuerzo a partir de medidas de tamaño y diseño. En este estudio los resultados arrojan que el uso de modelos híbridos combinando la regresión Poisson y árboles de regresión CART claramente superan la exactitud de modelos que usan solamente la regresión Poisson.

Por su parte, el estudio publicado en (Jeffery *et al.*, 2001) analiza la precisión de diferentes técnicas de estimación y examina su comportamiento basado en bases de proyectos tanto de distintas compañías como de una misma compañía. Entre las técnicas que se usaron en esa investigación, una versión de árbol CART fue de las que mejores resultados brindó cuando se trabajó con datos de una misma compañía. Sin embargo, al utilizar los datos de varias compañías, fue la técnica que peores resultados ofreció.

Recientemente, los Árboles de Regresión Aditivos Múltiples (Multiple Additive Regression Trees - MART) han sido propuestos como un novedoso avance que extiende y mejora el modelo de los árboles basados en el algoritmo CART usando el gradiente estocástico fortalecido (*stochastic gradient boosting*). En (Elish, 2009) se evalúa el potencial de este tipo de árbol para la estimación de esfuerzo de desarrollo de proyectos de software comparado con otros modelos publicados anteriormente, en términos de precisión. Luego de analizar los resultados logrados usando MART, con los que se referencian en otras publicaciones en las que se hace uso de SVR, Regresión Lineal y Redes Neuronales RBF,

se llega a la conclusión de que el modelo obtenido con el uso de MART mejora la exactitud de la estimación de los otros modelos. Estos resultados se obtuvieron sobre la base de proyectos de la NASA.

También en (Oliveira *et al.*, 2010) se hace uso de AD para la estimación de esfuerzo en específico mediante el *model tree* M5P. Aquí se combinan varias técnicas con AG. En (Idri and Elyassami, 2011) se investiga el uso del algoritmo ID3 borroso para la estimación de esfuerzo. Este AD es diseñado integrando los principios de los árboles ID3 y la teoría de los conjuntos borrosos, permitiendo al modelo manejar datos dudosos e imprecisos cuando se describen los proyectos de software. La principal característica de los ID3 borrosos es que un ejemplo pertenece a un nodo con un grado de certidumbre. Los autores analizan dos modelos ID3 borrosos para la estimación de esfuerzo, cada uno con una forma distinta de calcular la entropía borrosa, dependiendo del operador de conjunción usado. Los resultados del estudio muestran que el uso de un nivel de significación óptimo y una adecuada fórmula de calcular la entropía borrosa mejoran la precisión de los estimados. Al compararse con la versión precisa del ID3 se nota una gran mejora en la exactitud de la estimación.

Discusión

En este trabajo se ha presentado un conjunto de técnicas de Inteligencia Artificial utilizadas en la estimación del esfuerzo de proyectos de software, entre las que se encuentran: el Razonamiento Basado en Casos (RBC), los Algoritmos Genéticos (AG), la Programación Genética (PG), Support Vector Regression (SVR), Redes Neuronales Artificiales (RNA), Sistemas de Inferencia Borrosos (SIB) y Lógica Difusa (LD), Árboles de Decisión (AD), lo cual demuestra la tendencia actual de aplicaciones de estos modelos en la Ingeniería de Software. Se evidencia un aumento del uso de técnicas como los algoritmos genéticos y la programación genética, además de la técnica support vector regression en los últimos años.

Otro elemento a destacar es que se ha generalizado el uso de enfoques híbridos en la estimación, donde técnicas de aprendizaje automático se combinan entre sí. Esta combinación ha sido liderada por los algoritmos genéticos y la lógica difusa.

En los estudios realizados se comprueba que es aplicada la técnica de IA, combinada o no, sobre repositorios de proyectos reales que han sido orientados al tipo de problema que el investigador enfrenta. Sin embargo se aprecia la existencia de un amplio conjunto de repositorios de proyectos con las más disímiles características, donde no existe

un consenso en las variables que se almacenan. Esto hace muy difícil la selección del mejor modelo de estimación, ya que las pruebas no son realizadas sobre los mismos datos.

Las diferentes métricas empleadas para la evaluación de los modelos aplicados en la estimación también varían entre los investigadores de la temática. Se prefiere el uso de la magnitud del error medio relativo aunque pudiera ampliarse la evaluación con el uso de residuales absolutos y técnicas estadísticas.

Conclusiones

Existe una amplia variedad de técnicas de Inteligencia Artificial que han sido empleadas para la estimación del esfuerzo en proyectos de software. Entre las más usadas se encuentran el razonamiento basado en casos, las redes neuronales artificiales y los árboles de decisión. Juntas abarcan el 80% de los estudios aplicados a la estimación del esfuerzo desde los años 90 hasta la actualidad, aunque se evidencia un aumento del uso de otras técnicas de IA, o la combinación de técnicas en lo que se refiere como enfoque híbrido, en los últimos años.

La existencia de un amplio conjunto de repositorios de proyectos con características disímiles, donde no existe un consenso en las variables que se almacenan, hace difícil la selección del mejor modelo de estimación, ya que las pruebas no son realizadas sobre los mismos datos.

Se considera que el uso de técnicas de Inteligencia Artificial en la estimación del esfuerzo en proyectos de software constituye un área de investigación en desarrollo y de interés para la naciente industria cubana de software.

Referencias

- ADEKILE, O. Object-oriented software development effort prediction using design patterns from object interaction analysis. University of Georgia, 2008.
- AFZAL, W. & TORKAR, R. On the application of genetic programming for software engineering predictive modeling: A systematic review. Expert Systems with Applications, 2011.
- ATTARZADEH, I. & HOCK OW, S. A Novel Algorithmic Cost Estimation Model Based on Soft Computing Technique. Journal on Computer Science, 2010, 6: 117-125.

- ATTARZADEH, I., MEHRANZADEH, A. & BARATI, A. Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation. In: Fourth International Conference on Computational Intelligence, Communication Systems and Networks, 2012.
- BARCELOS TRONTO, I. F., SIMOES DA SILVA, J. D. & SANT'ANNA, N. An investigation of artificial neural networks based prediction systems in software project management. The Journal of Systems and Software 2008, 81: 356–367.
- BARCELOS TRONTO, I. F., SIMÕES DA SILVA, J. D. & SANT'ANNA, N. Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation. Brasil, Brazilian National Institute for Space Research - INPE, 2006.
- BELLO, R. P., GARCÍA, M. M., MORELL, C., RODRÍGUEZ, Y. & GÓMEZ, Y. Modelos de recuperación de casos con rasgos borrosos. Centro de Estudios de Informática. Facultad de Matemática, Física y Computación. Universidad Central de Las Villas. CUBA, 2002.
- BRAGA, P. L., OLIVEIRA, A. L. I., RIBEIRO, G. H. T. & MEIRA, S. R. L. Bagging Predictors for Estimation of Software Project Effort. In: International Joint Conference on Neural Networks, 2007 Orlando, Florida.
- BRIAND, L. C., BASILI, V. R. & THOMAS, W. M. A pattern recognition approach for software engineering data analysis. IEEE Transactions on Software Engineering, 1992, 18: 931-942.
- BRIAND, L., LANGLEY, T. & WIECZOREK, I. A replicated Assessment and Comparison of Common Software Cost Modeling Techniques. International Software Engineering Network Technical Report ISERN, 1999, 15.
- BRIAND, L. C. & WUST, J. Modeling Development Effort in Object-Oriented Systems Using Design Properties. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2001, 27.
- BRIAND, L. C. & WEICZOREK, I. Resource Estimation in Software Engineering. Encyclopedia of Software Engineering, 2002.
- BOEHM, B. Safe and Simple Software Cost Analysis. IEEE SOFTWARE, September/October 2000.
- BURGESS, C. & LEFLEY, M. Can genetic programming improve software effort estimation? A comparative evaluation. Information and Software Technology, 2001, 43: 863-873.
- CORAZZA, A., DI MARTINO, S., FERRUCCI, F., GRAVINO, C. & MENDES, E. Investigating the use of Support Vector Regression for web effort estimation. Empir Software Eng, 2011a, 16: 211–243.
- CORAZZA, A., DI MARTINO, S., FERRUCCI, F., GRAVINO, C., MENDES, E. & SARRO, F. Using tabu search to configure support vector regression for effort estimation. Empir Software Eng, 2011b.

- CORAZZA, A., DI MARTINO, S., FERRUCCI, F., GRAVINO, C. & MENDES, E. Applying Support Vector Regression for Web Effort Estimation using a Cross-Company Dataset. In: Third International Symposium on Empirical Software Engineering and Measurement, 2009c.
- CRESPO YÁÑEZ, F. J. Un modelo paramétrico difuso para la estimación del esfuerzo de desarrollo del software. Departamento de Ciencias de la Computación. Alcalá, Universidad de Alcalá, 2003.
- CUAUHTEMOC, L.-M. A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. Applied Soft Computing, 2011, 11: 724-732.
- DESHARNAIS, J. M. & AL., E. Adjustment Model for Function Points Scope Factors-A Statistical Study. IFPUG Spring Conference. Florida, 1990.
- ELISH, M. O. Improved estimation of software project effort using multiple additive regression trees. Expert Systems with Applications, 2009, 36: 10774–10778.
- FINNIE, G. R., WITTIG, G. E. & DESHARNAIS, J.-M. A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. Journal Systems Software, 1997, 39: 281-289.
- GARCÍA, A., GONZÁLEZ, R., COLOMO, J.L. & RUÍZ, B. Methodology for Software Development Estimation Optimization Based on Neural Networks. IEEE Latin America Transactions, 2011, 9: 384-398.
- GARCÍA, N., LÓPEZ, C. & CHAVOYA, A. A comparative study of two fuzzy logic models for software development effort estimation. Procedia Technology, 2013, 7: 305-314.
- HUANG, S.-J. & CHIU, N.-H. Optimization of analogy weights by genetic algorithm for software effort estimation. Information and Software Technology, 2006, 48: 1034-1045.
- HUANG, X., HO, D., REN, J. & CAPRETZ, L. F. Improving the COCOMO model using a neuro-fuzzy approach. Applied Soft Computing, 2007, 7: 29-40.
- HUANG, S.-J., CHIU, N.-H. & CHEN, L.-W. Integration of the grey relational analysis with genetic algorithm for software effort estimation. European Journal of Operational Research, 2008, 188: 898–909.
- IDRI, A., ABRAN, A. & LAILA, K. COCOMO Cost Model Using Fuzzy Logic. In: 7th International Conference on Fuzzy Theory and Technology. Atlantic City, New Jersey, 2000.
- IDRI, A. & ELYASSAMI, S. Applying Fuzzy ID3 Decision Tree for Software Effort Estimation. International Journal of Computer Science Issues, 2011, 8.
- JEFFERY, R., RUHE, M. & WIECZOREK, I. Using Public Domain Metrics to Estimate Software Development Effort. In: 7th International Software Metrics Symposium, Londres, Reino Unido, 2001.

- JØRGENSEN, M., INDAHL, U. & SJØBERG, D. Software effort estimation by analogy and ‘‘regression toward the mean’’. Journal of Systems and Software, 2003, 68: 253-262.
- JØRGENSEN, M., SHEPPERD, M. A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering, 2007, 33: 33-53.
- JUI-SHENG, C., MIN-YUAN, C., YU-WEI, W. & CHENG-CHIEH, W. Forecasting enterprise resource planning software effort using evolutionary support vector machine inference model. International Journal of Project Management, 2012, 30: 967–977.
- JUN, E. S. & LEE, J. K. Quasi-optimal case-selective neural network model for software effort estimation. Expert Systems with Applications, 2001, 21: 1-14.
- KEMERER, C. F. An Empirical Validation of Software Cost Estimation Models. Communications of the ACM, 1987, 30: 416-429.
- MACDONELL, S. G. & SHEPPERD, M. J. Combining techniques to optimize effort predictions in software project management. The Journal of Systems and Software, 2003, 66: 91-98.
- MARZA, V. & SEYYEDI, M. A. Fuzzy Multiple Regression Model for Estimating Software Development Time. International Journal of Engineering Business Management, 2009, 1.
- MARZA, V. & TESHNEHLAB, M. Estimating Development Time and Effort of Software Projects by using a Neuro_Fuzzy Approach. Advanced Technologies, 2009.
- MUKHOPADHYAY, T., VICINANZA, S. S. & PRIETULA, M. J. Examining the Feasibility of a Case Based Reasoning Model for Software Effort Estimation. MIS Quarterly. 1992
- OLIVEIRA, A. L. I. Estimation of software project effort with support vector regression. Neurocomputing, 2006, 69: 1749–1753.
- OLIVEIRA, A. L. I., BRAGA, P. L., LIMA, R. M. F. & CORNÉLIO, M. L. GA-based method for featur selection and parameters optimization for machine learning regression applied to software effort estimation. Information and Software Technology, 2010, 52: 1155-1166.
- PATNAIK, K. S., MALHOTRA, S. & SAHOO, B. Software Development Effort Estimation using CBR: A Review. 2004.
- PENDHARKAR, P. Probabilistic estimation of software size and effort. Expert Systems with Applications, 2010, 37: 4435–4440.
- RAYNOR, W. J., JR. The International Dictionary of Artificial Intelligence, Glenlake Publishing Company, Ltd., 1999.

- SALVETTO DE LEÓN, P. F. Modelos automatizables de estimación muy temprana del tiempo y esfuerzo de desarrollo de sistemas de información. Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software. Madrid, Universidad Politécnica de Madrid, 2006.
- SANDHU, P. S., BASSI, P. & SINGH BRAR, A. Software Effort Estimation Using Soft Computing Techniques. World Academy of Science, Engineering and Technology, 2008, 488-491.
- SHEPPERD, M. & SCHOFIELD, C. Estimating software project effort using analogies. IEEE Transactions on Software Engineering, 1997, 23: 736-743.
- SIKKA, G., KAUR, A. & UDDIN, M. Estimating Function Points: Using Machine Learning and Regression Models. In: 2nd International Conference on Education Technology and Computer (ICETC), 2010.
- SMOLA, A. J. & SCHOLKOPF, B. A Tutorial on Support Vector Regression. Statistics and Computing, 2004, 14: 199-222.
- SRINIVASAN, K. & FISHER, D. Machine Learning Approaches to Estimating Software Development Effort. IEEE Transactions on Software Engineering, 1995, 21: 126-137.
- SUN-JEN, H. & NAN-HSING, C. Optimization of analogy weights by genetic algorithm for software effort estimation. Information and Software Technology, 2006, 48: 1034-1045.
- SUN-JEN, H., NAN-HSING, C. & LI-WEI, C. Integration of the grey relational analysis with genetic algorithm for software effort estimation. European Journal of Operational Research 2008, 188: 898–909.
- WALKERDEN, F. & JEFFERY, R. An empirical study of analogy-based software effort estimation. Empirical Software Engineering, 1999, 4: 35-158.
- WEN, J., LI, S., LIN, Z., HU, Y. & HUANG, C. Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology, 2012, 54: 41–59.
- WU, D., LI, J. & LIANG, Y. Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. The Journal of Supercomputing, 2010.
- ZENG, H. & RINE, D. Estimation of Software Defects Fix Effort Using Neural Networks. 28th Annual International Computer Software and Applications Conference (COMPSAC'04), 2004.