La enseñanza de métricas de software

Edgar Danilo Domínguez Vera*
José Luis Martínez Flores*

Resumen

Dentro de la Ingeniería del Software existe un interés especial en tratar de evaluar la calidad del software, ya sea paquete de aplicación o lenguaje de programación.

En el terreno académico universitario, donde se prepara a profesionistas que se van a dedicar al desarrollo de software, existe la preocupación por saber cuáles son las bases teóricas que le permitirán al egresado generar código fuente de calidad.

Atendiendo a la anterior preocupación, se desarrolló una investigación cuyo objetivo general fue determinar si impartir un curso de métricas de software, a los estudiantes de sistemas computacionales, contribuye al desarrollo de código fuente de mejor calidad.

Los resultados obtenidos son contundentes a favor de la hipótesis anteriormente planteada. En el presente artículo se detalla el marco teórico y se destacan las pruebas estadísticas que apoyan la anterior afirmación.

1. INTRODUCCIÓN.

Dado que una compañía que desarrolla software puede gastar hasta el 70% de su esfuerzo en dar mantenimiento al software existente, es necesario que se desarrolle de buena calidad, esto es, que permita su facilidad de mantenimiento; entendiendo esto último como la facilidad de comprender, corregir y/o mejorar el software .[1]

La medición del software es importante ya que permite a los administradores y desarrolladores a entender mejor el proceso de desarrollo, así como la calidad del software que se produce.

Ahora bien, la medición de la calidad del software es un punto particularmente controversial. Sin embargo, al investigar el marco teórico, se encontró que el utilizar la Ciencia del Software de Halstead[2] para medir dicho parámetro ha tenido una amplia aceptación.[1]

La Ciencia del Software de Halstead, es una forma de medir el software cuantitativamente, y se ha demostrado que los estudiantes que han recibido una enseñanza en éstas y otras métricas, producen programas que exhiben menos complejidad, requieren menos tiempo de codificación y prueba, y además, es más fácil de darle mantenimiento.[3]

En este artículo se discutirá en el punto 2 el marco teórico que sirvió de referencia a la investigación; en el

* Programa Doctoral en Ingeniería de Sistemas de la FIME - UANL
. edoming@gama.fime.uanl.mx
jlmartin@ccr.dsi.uanl.mx

punto 3, se detallará la metodología; en el punto 4, se hablarán de las limitaciones del estudio; en el punto 5, se discutirá el análisis de los resultados y finalmente, en el punto 6, se darán las conclusiones y recomendaciones.

2. MARCO TEÓRICO.

Las métricas de Halstead[2] consideran que un programa está formado por una serie de partículas, las cuales pueden ser consideradas como operadores u operandos.

Los operandos son definidos como las variables o constantes que se utilizan en la implementación, mientras que los operadores son los símbolos que afectan el valor u orden del operando.

Las métricas básicas en la Ciencia del Software son:

- n1 = número de operadores distintos
- n2 = número de operandos distintos
- N1 = número total de operadores
- N2 = número total de operandos

A partir de estas métricas básicas se definen un conjunto de métricas para las características de un programa tales como

- n = Vocabulario
- N = Longitud del Programa
- V = Volumen
- V* = Volumen Potencial
- L = Nivel del Programa
- D = Dificultad del Programa
- E = Esfuerzo de Programación
- T = Tiempo de Programación
- I = Contenido de Inteligencia
- λ = Nivel del Lenguaje

Ahora bien, como se dijo anteriormente, las métricas de Halstead ayudan a medir la calidad del código fuente.

Por otro lado, en una investigación anterior [4] se desarrolló un analizador de código fuente para lenguaje FoxPro2, mismo que arroja como resultados las métricas de Halstead para un programa hecho en dicho lenguaje. Este analizador fue utilizado después de verificar que funciona para FoxPro2.6 para Windows.[5]

El diseño del experimento cae en la categoría de un experimento verdadero de tipo explicativo[6] donde se formaron un grupo experimental y uno de control. Para lograr que los grupos fueran equivalentes se recurrió al emparejamiento. Se plantearon hipótesis de causalidad bivariada de diferencia de medias entre los grupos con un cierto sentido de entendimiento, y finalmente se hicieron pruebas Z a las muestras tomadas.

3. METODOLOGÍA.

En esta investigación se formó un grupo experimental y uno de control, con seis estudiantes cada uno.

También, se detectaron una variable independiente y una dependiente. La independiente es el curso de métricas del software que se le impartió al grupo experimental y la dependiente es la calidad de los programas, que se va a medir en ambos grupos.

Posteriormente, el grupo experimental se expuso a la presencia de la variable independiente, y finalmente, se buscó diferencias en los resultados de la calidad de los programas.

Así, este estudio fue hecho con las siguientes características:

1. Se eligió al lenguaje FOXPRO2.6 para Windows porque en 1997 fue el lenguaje más solicitado por las empresas regiomontanas en sus anuncios periodísticos y en la bolsa de trabajo de la FIME-UANL y, porque se imparte actualmente en la materia de Programación III de la FIME-UANL.[7]
2. Se hicieron un grupo experimental y uno de control constituidos por seis estudiantes cada uno. Los estudiantes eran alumnos de la carrera de Ingeniero Administrador de Sistemas en la FIME-UANL, del quinto semestre y que cursaron, en el semestre febrero-agosto de 1998, la materia programación III: FOXPRO2.6 para Windows.
3. Los grupos se hicieron equivalentes a través del emparejamiento con respecto a la calificación que obtuvieron en la materia [Tabla 1]. El maestro que les impartió la materia fue el mismo.
4. Cada estudiante hizo diez programas en FOXPRO2.6 para Windows, los programas fueron los mismos para ambos grupos.
5. Antes que los alumnos del grupo experimental hicieran los programas del punto anterior, se les impartió, en un lapso de 6 horas, un curso de métricas de software.
6. Se utilizó un analizador de código fuente realizado en una investigación anterior, mismo que calcula las métricas de Halstead para cada programa.[4]
7. Finalmente, se hicieron pruebas Z para comparar medias entre los grupos. La comparación se hizo en cuanto el volumen de los programas, el nivel de los programas, el contenido de inteligencia en los programas, el esfuerzo de programación, el tiempo estimado de programación y el nivel del lenguaje.

Elegir el nivel de significancia α para hacer las pruebas "Z" fue particularmente interesante, ya que en ingeniería el valor más utilizado es α=0.05, sin embargo, en ingeniería se hacen mediciones para elemento físicos como pudieran ser tornillos, tuercas, clavos etc. pero en este caso particular, estamos midiendo software que es un elemento lógico derivado de procesos complejos de la mente humana, por lo que se considera que encontrar una significancia mayor al 80%, es decir un α=0.20, es suficientemente significativa dada la naturaleza del problema.

El curso de métricas de software que se impartió al grupo experimental cubre los capítulos 1, 2 3, 20 del libro de Pressman,[1] además el tema de la Ciencia del Software[1,2] y, un programa hecho en dos versiones al que se sometió al analizador de código. De esta manera, los estudiantes vieron que un mismo programa, con diferente estructura de codificación, arroja valores diferentes para las métricas de calidad de Halstead.

Grupo Experimental		Grupo de Control	
Alumno	Calificación	Alumno	Calificación
A	70	A	70
B	73	B	81
C	85	C	81
D	85	D	85
E	88	E	89
F	93	F	90
Suma	494	Suma	496
Promedio	82.33	Promedio	82.67

Tabla 1 Grupos Equivalentes

4. LIMITACIONES DEL ESTUDIO

Se puede clasificar a los programas por su longitud (***N***) como pequeños (***N*<=100**), medianos (***N*>100** y ***N*<=500**) , grandes (***N*>500** y ***N*<=1500**) y muy

grandes (***N*>1500**), entendiéndose como longitud de un programa (***N=N1+N2***) a la suma de los operadores y operandos que se utilizan en la codificación de un algoritmo.

Por lo tanto, una de las limitaciones de este estudio es que la dispersión de los programas, en cuanto a la longitud (***N***), está restringida a programas que son medianos. Así, la dispersión para cada grupo quedó como se muestra en la Tabla 2:

	Grupo Experimental	Grupo de Control
Pequeños	3.33%	1.66%
Medianos	88.33%	88.33%
Grandes	8.33%	10.00%
Muy grandes	0%	0%

Tabla 2 Dispersión del tamaño de los programas

Se cree que si los programas tienen una mejor dispersión, en cuanto a ***N***, y si tienen rangos más grandes se puede hacer un mejor análisis.[4]

Otra limitación es que la función de todos los programas es para hacer reportes, es decir, lista que siempre presenta los datos con el mismo formato.[8] Así, en esta investigación no se incluyen programas de actualizaciones, consultas, administración de bases de datos, funciones matemáticas, menús y otros. Se comenta lo anterior porque se ha comprobado que la función de un programa influye en la correlación entre ***N*** y su estimador (***N^),***[1] y en el nivel del lenguaje.[4,8]

También podemos agregar dentro de las limitaciones, que si bien la muestra en cuanto a número de programas se considera, estadísticamente hablando, grande; la muestra, en cuanto a número de estudiantes en cada grupo (el experimental y el de control), es pequeña (seis estudiantes por cada grupo).

5. ANÁLISIS DE RESULTADOS.

En Resumen, los resultados encontrados fueron:

1. Se encontró que en la hipótesis que establece que la media del volumen de los programas hechos por el grupo experimental, es menor que la media del volumen de los programas hechos por el grupo de control; hay una tendencia a favor del 61.83%, esto con un nivel de confianza del 80%. Sin embargo, no hay suficiente prueba estadística que permita aceptar dicha hipótesis.
2. Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel de los programas hechos por el grupo experimental, es mayor a la media del nivel de los programas hechos por el grupo de control.
3. Se encontró que en la hipótesis que establece que la media del contenido de inteligencia de los programas hechos por el grupo experimental, es mayor a la media del contenido de inteligencia de los programas hechos por el grupo de control, hay una tendencia a favor del 34.48%, esto con una confiabilidad del 80%. Sin embargo, no hay suficiente prueba estadística que permita aceptar dicha hipótesis.
4. Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del esfuerzo de programación de los programas hechos por el grupo experimental, es menor a la media del esfuerzo de programación de los programas hechos por el grupo de control.
5. Hay suficiente prueba estadística, con una confiabilidad del 80%, que permite aceptar que la media del tiempo estimado de programación de los programas hechos por el grupo experimental, es

menor a la media del tiempo estimado de programación de los programas hechos por el grupo de control.

6. Hay suficiente prueba estadística, con una confiabilidad del 90%, que permite aceptar que la media del nivel del lenguaje de los programas hechos por el grupo experimental, es mayor a la media del nivel del lenguaje de los programas hechos por el grupo de control.

Los resultados estadísticos se pueden apreciar en la tabla 3

Resultados de Prueba Z					
M1=Media del Grupo Experimental M2=Media del Grupo de Control					
	Z	Z alfa	Hipótesis	alfa	Resultado
Volumen	-0.5204	-0.8416	M1 < M2	0.20	Rechazo
Nivel Prog.	1.5580	1.2815	M1 > M2	0.10	Aceptado
Cont. Int.	0.2902	0.8416	M1 > M2	0.20	Rechazo
Esfuerzo	-0.9928	-0.8416	M1 < M2	0.20	Aceptado
Tpo. Prog.	-0.9928	-0.8416	M1 < M2	0.20	Aceptado
Nivel Leng.	1.2981	1.2815	M1 > M2	0.10	Aceptado

Tabla 3 Resumen de Resultados.

En conclusión, la enseñanza de métricas de software sí permite, a los estudiantes de sistemas computacionales, generar código fuente de mejor calidad que si no hubieran recibido tal enseñanza.

6. CONCLUSIONES Y RECOMENDACIONES

Se encontró que impartir un curso de métricas a los estudiantes de Sistemas Computacionales sí contribuye para que generen software con buena calidad.

La buena calidad del software se puede tener cuando los programas tienen menos impurezas, tales como: una menor utilización de operandos y operadores, menos reemplazos innecesarios, más expresiones matemáticas factorizadas, menos operandos sinónimos y/o ambiguos (que se llaman igual pero tienen una función diferente dentro de un programa), menos instrucciones innecesarias, etc.

Si un programa cumple con lo anterior, será mas fácil de entender, más fácil de modificar y por lo tanto será más económico en tiempo, dinero y esfuerzo, para darle mantenimiento.

Por otro lado, el tipo de programación que se utilizó en este estudio es la llamada procedural, hoy, la programación que viene empujando fuerte es la orientada a objetos, que es una filosofía muy distinta a la programación procedural. Investigaciones realizadas han encontrado que las métricas de Halstead son válidas para el lenguaje de Programación Orientado a Objetos C++ [9] En una posterior investigación se puede ver si los resultados son válidos para otros lenguajes orientados a objetos como Visual FoxPro.

Otro tema para futuras investigaciones es que sería conveniente ver la medición de la calidad del software a través de los puntos de función [1] Hay un Grupo Internacional de Usuarios de Punto de Función (International Function Point User's Group UFPUG) que ofrecen información al respecto, para contactarlos accesar:

http://www.bannister.com/ifpug/home/docs/comm.html

Otras referencias son:

webmaster@softwaremetrics.com,
http://www.softwaremetrics.com/esp/fivemajor.htm,
http://www.spr.com/library/0funcmet.htm.

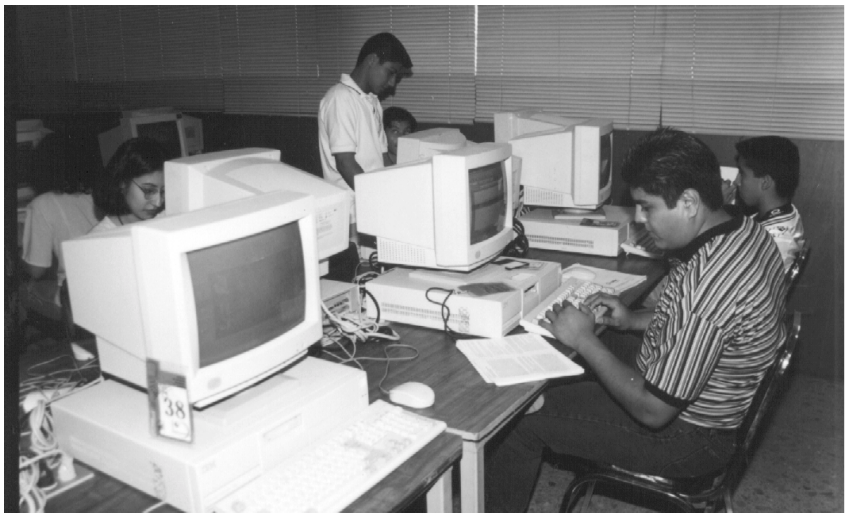
Por otro lado, Warren Harrinson del PSU Center for Software Quality Research de Portland State University y Gene Miluk de Denver Metrics Gruop realizan investigaciones para usar La Ciencia del

Software como una aproximación a los Puntos de Función para Código Fuente Existente. (Using Software Science as a Proxy for Function Points for Existing Code Assets) para más información en:

www.cs.pdx.edu/~warren/Papers/FP_PR.htm

REFERENCIAS.

1. R.S. Pressman, *Ingeniería del Software: Un enfoque práctico*, tercera edición; McGraw-Hill, España, 1993.
2. M.H. Halstead, *Elements of Software Science*, Elsevier North-Holland, 1977.
3. B.J. Bowman, y W.A. Newman W.A. "Software Metric as a Programming Training Tool", *J. Systems Software*, Vol 13, pp. 139-147, 1990.
4. J.L. Martínez Flores, *Métricas de Software en Lenguajes de Cuarta Generación*, Tesis de Maestría en Ciencias de la Administración con Especialidad en Sistemas. UANL-FIME. San Nicolás de los Garza, N.L. México, 1994.
5. J.J. García-Badel, *FoxPro2.5 para DOS y Windows: A su Alcance*, McGraw-Hill, España, 1997.
6. R. Hernández Sampieri , C. Fernández Collado y P. Baptista Lucio, *Metodología de la Investigación*, McGraw-Hill, México, 1995
7. M.G. Gutierrez Alanis, *Demanda y Perfil de Profesionistas Solicitados durante el Año de 1997 de las Carreras Ofrecidas por FIME UANL*, Secretaría de Planeación y Desarrollo, 1998.
8. L.G. Navarro Guerra, J.L. Martínez Flores, A.M. Álvarez Socarrás, "Estimación del Tamaño de un Programa y del Nivel del Lenguaje para el Lenguaje Progress utilizando Métricas de Halstead", *Proceedings CONIELECOMP '97*, Cholula, Puebla, 1997, p.p.213-216.
9. X. Espinosa de los Monteros Anzaldúa, J. L. Martínez Flores, A.M. Álvarez Socarrás, "Utilización de las Métricas de Halstead para la Estimación del Tamaño de un Programa y del Nivel del Lenguaje para el Lenguaje de Programación Orientado a Objetos C++", *Proceedings CONIELECOMP '97*, Cholula, Puebla, 1997, pp. 217-220



Apartado Postal 076-F, Cd. Universitaria, C.P. 66450, San Nicolás de los Garza, N. L. México Tel. (01-8)329-40-20 ext. 5863, Fax (01-8)332-09-04, Cel. (044-8) 183-46-05
e-mail: edoming@gama.fime.uanl.mx, jlmartin@ccr.dsi.uanl.mx

Edgar Danilo Domínguez Vera es Maestro de tiempo completo de la U.A.N.L-F.I.M.E desde 1991. Es Ingeniero Administrador de Sistemas egresado de la F.I.M.E. - U.A.N.L. en 1990. Actualmente es Comisario de la Asociación Nacional de Instituciones de Educación en Informática (A.N.I.E.I.), y realizó esta investigación para obtener el Grado de Maestría en Ciencias de la Administración con especialidad en Sistemas en la F.I.M.E. - U.A.N.L. en donde también se desempeña como Coordinador de Sistemas

José Luis Martínez Flores es Doctor en Ingeniería de Sistemas graduado de la Universidad Autónoma de Nuevo León. Actualmente es profesor investigador y Director Adjunto del Programa Doctoral en la Facultad de Ingeniería Mecánica y Eléctrica de la U.A.N.L. Sus líneas de investigación están orientadas a Optimización en Redes y Métricas de Software.

Edgar Danilo Domínguez Vera, José Luis Martínez Flores
Programa Doctoral en Ingeniería de Sistemas.
Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León,