[54]

Sedano, T. (2016). Code Readability Testing, an Empirical Study. In 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET) (pp. 111–117). IEEE. <https://doi.org/10.1109/CSEET.2016.36>

**KEYWORDS AUTOR:**

**TÍTULO**:

Prueba de legibilidad de código, un estudio empírico

**PARA ANEXAR A DOCUMENTO:**

**\*\* INICIO \*\*\***

**DATASET**:

**DOCUMENTACIÓN:**

<http://sedano.org/toddsedano/2013/02/19/improving-code-readability-turning-comments-into-methods.html>

**LENGUAJE DE PROGRAMACIÓN:**

**TAGS**

LEGIBILIDAD; MEJORES PRACTICAS; CARACTERISTICAS; EDUCACION;

Sedano (2016) [54] realizó un estudio para evaluar la legibilidad del código con 21 estudiantes de maestría en ingeniería de software, del curso “Craft of Software Development” de la universidad de Carnegie Mellon, Estados Unidos, en 2013. La prueba consistió en 4 sesiones donde se reunió un programador, denominado el autor, y un lector, donde el primero dió una breve descripción de la funcionalidad y el lector luego leía y comprendía por sí sólo cómo el código realizaba los procesos [54]. Al finalizar, el lector daba la retroalimentación correspondiente y el programador realizaba los ajustes para la siguiente sesión [54]. Los resultados de la investigación, mostraron que la capacidad de los programadores para escribir código legible mejoró, así como la identificación de las soluciones frecuentes para mejorar la legibilidad del código [54]. Las sugerencias más frecuentes que se obtuvieron al finalizar las pruebas fueron: mejorar los nombres de las variables, mejorar los nombres de los métodos, crear nuevos métodos para reducir la duplicación del código, simplificar las condiciones y estructuras, y simplificar las condiciones del bucle [54].

**CARACTERÍSTICAS**: nombres de las variables, nombres de los métodos, duplicación del código, longitud de línea.

**APLICACIÓN**: Pruebas de legibilidad del software para el aprendizaje.

**\*\* FIN \*\*\***

**RESUMEN COMPLETO**

Sedano (2016) realizó un estudio para evaluar la legibilidad del código con 21 estudiantes de maestría en ingeniería de software, del curso “Craft of Software Development” de la universidad de Carnegie Mellon en 2013. La prueba de legibilidad del código mejoró la capacidad de los programadores para escribir código legible e identificar las soluciones frecuentes necesarias para mejorar la legibilidad del código. El estudio reveló que las sugerencias más frecuentes para aumentar la legibilidad del código son mejorar los nombres de las variables, mejorar los nombres de los métodos, crear nuevos métodos para reducir la duplicación del código, simplificar las condiciones y estructuras, y simplificar las condiciones del bucle.

La Prueba de legibilidad del código, cuyo objetivo es la comprensión del código, revela áreas donde el código no es legible y permite un diálogo entre el codificador y el lector. La retroalimentación es instantánea, ya que el autor ve exactamente cómo el lector interpreta el código.

Funciona con el siguiente proceso:

1. El autor le indica al lector el uso y funcionalidad producido, sin detalles del diseño del código.
2. El autor indica que archivos fueron adicionados o modificados y explica los casos de prueba.
3. El lector lee el código y luego describe en sus palabras el proceso que realiza el código. Puede hacer preguntas al autor.
4. El autor toma notas, de lo que puede mejorar.
5. Al finalizar el lector confirma con el autor que haya comprendido. El autor pregunta al lector sobre su experiencia.
6. El autor y el lector discuten como mejorar el código.

TRABAJOS RELACIONADOS CON LA IMPORTANCIA DE LA LEGIBILIDAD

|  |  |
| --- | --- |
| **AUTOR** | **DESCRIPCIÓN** |
| Andrew Hunt, David Thomas (2000) | En el libro “Pragmatic Programmers” examinaron las herramientas, procesos y trucos que ayudan a los programadores a dominar su oficio. |
| Robert Martin (2008) | En el libro “Clean Code” direccionaron técnicas para ayudar a los desarrolladores a ser mejores programadores. |
| Kent Beck (2006) | En el libro “Implementation Patterns, abordaron buenos patrones de diseño para desarrollar software. |
| Wang (2011) | Examinaron la inserción automática de líneas en blanco en el código para mejorar la legibilidad. |
| Sasaki (2013) | Reordenaron las declaraciones para mejorar la legibilidad, por declarar las variables inmediatamente antes de utilizarlas. |
| Binkley et al (2009)  Butler et al (2010)  Liblit et al (2006)  Relf (2005) | Examinaron la influencia de los nombres de identificadores en la legibilidad del software. |
| Relf (2005) | Herramienta de Relf para apoyar a los desarrolladores en mejorar los nombres de las variables y los métodos. |
|  |  |

Para esta investigación, la legibilidad fue definida como la cantidad de esfuerzo mental requerido para comprender el código, teniendo 4 puntajes desde 1 a 4, donde 1 era “muy dificil” y 4 era “muy fácil de leer”. Los de puntaje 4 requirieron menor tiempo por parte de los participantes.

Resultados: durante las 4 sesiones del experimento, varios programadores mejoraron la legibilidad de sus programas.

Lista de aspectos que surgieron en los experimentos para mejorar la legibilidad (de mayor a menor frecuencia):

mejorar los nombres de las variables\*, mejorar los nombres de los métodos, extraer métodos para reducir duplicación del código, simplificar las condiciones “if”, reducir los “if” anidados, simplificar las condiciones de los bucles \*, reducir las estructuras cíclicas, mejorar los nombres de clases, re secuenciar los argumentos de los métodos, simplificar estructuras de datos.