[9]

Alawad, D., Panta, M., Zibran, M., & Islam, R. (2018). An Empirical Study of the Relationships between Code Readability and Software Complexity.

**TAGS**

LEGIBILIDAD; CONSTRUCCIONES DE CODIGO; METRICAS; EVOLUCION; COMPLEJIDAD; CARACTERISTICAS;

**HERRAMIENTA**

<https://github.com/ipeirotis/ReadabilityMetrics>

**DATASET**

<https://github.com/zibranm/dataset>

**PARA ANEXAR A DOCUMENTO:**

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

Para una mejor calidad en el software, se desea una baja complejidad y una alta legibilidad. Alawad et al. logran demostrar que la complejidad y la legibilidad del software tienen una relación inversamente proporcional, es decir, que cuando una baja la otra sube; Así, si un código fuente presenta una alta complejidad, eso significa que la legibilidad del mismo es baja, pero si la complejidad es baja, la legibilidad es alta.

**FACTORES:**

**CONSTRUCCIONES DE CÓDIGO**

Alawad et al. (2018) evaluaron el efecto que tienen las construcciones de código en la legibilidad de un software, encontrando que la presencia de comentarios, el espaciado adecuado y los nombres significativos contribuyen a una alta legibilidad del código [9]. El bucle while es mejor que el bucle do-while para una mejor legibilidad. El uso de matrices y bucles anidados no ayuda mucho en la legibilidad del programa [9].

**Construcciones de código (Características):**

1. Lines of Code (LOC)

2. Meaningful Names

3. Comment Indents

4. Indents

5. Scope

6. Inheritance

7. Polymorphism

8. Class Distribution

9. Spacing

10. Recursive

11. Fórmulas

12. Consistency

13. Line Length distribution

14. Identifier name Length

15. Identifier frequency

16. IF-else

17. Nested if

18. For Loop

19. While Loop

20. Do-While Loop

21. Nested -loop

22. Switch

23. Array

**Método:**

Se uso machine learning usando la herramienta WEKA. En el contexto de aprendizaje automático, cada programa Java se considera una instancia y cada construcción de código se considera una característica. Para la clasificación de características con WEKA, elegimos la selección de características basada en correlación, porque es una técnica popular para seleccionar los atributos / características más relevantes en un conjunto de datos.

RESULTADO: **La presencia de comentarios, el espaciado adecuado y los nombres significativos contribuyen a una alta legibilidad del código. El bucle while es mejor que el bucle do-while para una mejor legibilidad. El uso de matrices y bucles anidados no ayuda mucho en la legibilidad del programa.**

**MÉTRICAS DE LEGIBILIDAD DEL SOFTWARE**

* El Índice de legibilidad automatizada (ARI):

ARI [28] se basa en la relación entre la dificultad de la oración y la dificultad de la palabra. La dificultad de las oraciones se determina como palabras por oración y la dificultad de las palabras es al calcular letras por palabras. La ecuación para calcular ARI es:

ARI = 4.71 (caracteres) + 0.5 (palabras) - 21.43

El valor numérico de la métrica ARI es calcular el nivel de grado necesario para comprender el texto. Por ejemplo, ARI = 3 significa que los estudiantes de 3er grado (de 8 a 9 años de edad) deberían poder comprender el texto [28].

* La medida simple de Gobbledygook (SMOG):

SMOG es sugerido por G Harry McLaughlin [22] en 1969. Esta métrica evalua el tiempo (en años) requerido por cualquier persona para leer el texto. La ecuación para calcular SMOG es:

SMOG = 3 + Raíz cuadrada de recuento polisillable

El valor métrico SMOG significa un nivel de grado escolar de EE. UU. Que indica que un estudiante promedio en ese nivel de grado puede leer el texto [28]. Por ejemplo, SMOG = 7.4 indica que el texto es entendido por un estudiante promedio en séptimo grado.

* Índice de legibilidad de Flesch-Kincaid (FKI):

El valor del índice de legibilidad de Flesch-Kincaid [24] se calcula utilizando la siguiente fórmula:

206.835− 1.015 \* (𝑡𝑜𝑡𝑎𝑙 𝑤𝑜𝑟𝑑𝑠 / 𝑡𝑜𝑡𝑎𝑙 𝑠𝑒𝑛𝑡𝑒𝑛𝑐𝑒𝑠) - 84.6 \* (𝑡𝑜𝑡𝑎𝑙 𝑠𝑦𝑙𝑙𝑎𝑏𝑙𝑒𝑠 / 𝑡𝑜𝑡𝑎𝑙 𝑤𝑜𝑟𝑑𝑠)

El valor alto del índice Flesh-Kincaid (FKI) indica una alta legibilidad del código, mientras que un valor bajo implica que el código es difícil de leer. El rango de esta métrica es un número de 0 a 100. una puntuación más alta indica una lectura más fácil. Un documento promedio tiene un puntaje FKI entre 6 y 70. Como regla general, un estudiante promedio de 5to grado puede entender puntajes de 90-100. Los estudiantes de octavo y noveno grado pueden entender documentos con un puntaje de 60-70; y los graduados universitarios pueden entender documentos con un puntaje de 0-30.

* El índice de confusión de Gunning (GFI):

Otra métrica de legibilidad utilizada en este estudio es el Índice de confusión de Gunning, originalmente propuesto por Robert Gunning [23]. Utiliza la longitud promedio de las oraciones y el porcentaje de palabras difíciles. La ecuación para calcular el índice de confusión de Gunning es la siguiente:

GFI = 0.4 (ASL + PHW)

donde, ASL es la duración promedio de la oración y PHW es el porcentaje de palabras difíciles. Esta métrica es similar a la escala Flesch en que compara sílabas y longitudes de oraciones. Un puntaje GFI de 5 significa legible, 10 significa duro, 15 significa difícil y 20 indica muy difícil. Según su nombre, las palabras 'Foggy' son palabras que contienen 3 o más sílabas.

* Índice Coleman-Liau (CLI):

Meri Coleman y T. L. Liau [25] definieron otro índice de legibilidad como ARI, que se utiliza en este estudio para determinar el valor de legibilidad del código. Se calcula usando la siguiente fórmula:

CLI = 0.0588L - 0.296S - 15.8

donde, L y S son el número promedio de letras y oraciones respectivamente. Se basa en caracteres en lugar de sílabas por palabra y longitud de la oración. Un valor de CLI calculado utilizando la fórmula mencionada anteriormente significa una calificación de escuela secundaria. Por ejemplo, CLI = 10.6 para un texto dado significa que el texto puede leerse adecuadamente para un estudiante de secundaria de 10-11 grado.

* Puntuación de legibilidad de Buse (BRS):

El puntaje de legibilidad de Buse es la probabilidad de que un determinado código sea altamente legible. Buse y Weimer [14] desarrollaron un clasificador de aprendizaje automático de dos clases (es decir, de alta lectura y baja lectura) basado en una serie de construcciones de programas como características. BRS es la probabilidad de que un código determinado se clasifique en la clase de alta lectura.

**EVOLUCIÓN DE MODELOS Y MÉTRICAS DE LEGIBILIDAD DEL SOFTWARE**

|  |  |
| --- | --- |
| **Autores** | **Descripción** |
| Buse & Weimer (2008) | han propuesto una métrica de legibilidad de código y desarrollado una herramienta de legibilidad que mide automáticamente la métrica de legibilidad propuesta. Precisión de 80%, sobre fragmentos de código Java.  El estudio mostró que la legibilidad está fuertemente correlacionada con algunos atributos de calidad del software, tales como cambios de código, mensajes de registro de defectos e informes automáticos de defectos. |
| Posnett et al (2011) | argumentó que la legibilidad del código es una propiedad subjetiva y no es persuadible generar puntaje de legibilidad utilizando la herramienta de legibilidad automatizada. Además, incluyeron que la legibilidad depende en gran medida de la información contenida en el código fuente y, por lo tanto, el puntaje de legibilidad se puede calcular en función del tamaño y la entropía del código |
| Pahal et al (2017) | realizó una revisión de métricas para la calidad del software. Los autores revisaron varias métricas de legibilidad en la literatura, como la métrica Flesh-Kincaid, la métrica Gunning-Fog, el índice SMOG, el índice de legibilidad automatizada y el índice Coleman-Liau y concluyen que la elección de las métricas de legibilidad depende de diferentes empleos. Mencionan varios elementos de código que mejoran y degradan la legibilidad, por ejemplo, comentarios apropiados y variables mal definidas, respectivamente |

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

**NOTAS**

Los resultados se derivan de un **análisis de 35 programas Java que cubren 23 construcciones** de código distintas. El análisis incluye **seis métricas** de legibilidad y dos métricas de complejidad. Nuestro estudio confirma empíricamente la sabiduría existente de que la legibilidad y la complejidad están correlacionadas negativamente.

TEMAS CLAVE:

* Métricas de legibilidad del software - Readability metrics
* Métricas de Complejidad del software

**LEGIBILIDAD**

La legibilidad del código se define como el juicio humano de determinar si un código fuente es comprensible y fácil de leer. Se considera como una importante métrica de calidad del software, ya que tiene una gran influencia en el mantenimiento del software. Normalmente, la fase de mantenimiento consume del 40% al 80% del costo total del ciclo de vida de un software (Dubey y Rana 2011) [9]. Aggarwal et al (2012) afirmaron que la legibilidad del código fuente y la legibilidad de la documentación son cruciales para la mantenibilidad de un software [9]. Los investigadores Raymond (1991), Deimel (1985) y Rugaber 2000), identificaron a la lectura del código como una actividad clave en el mantenimiento del software, y también reconocieron que es la actividad que consume más tiempo entre todas las actividades de mantenimiento [9].

La legibilidad del código es considera como una "propiedad accidental", no es esencial, ya que no está determinada por el espacio del problema, y puede ser controlada en gran parte por los ingenieros de software. Mientras que las métricas de complejidad del software miden el tamaño de las clases y los métodos, el acoplamiento y las interdependencias entre los módulos, la legibilidad del código considera factores locales y línea por línea, tales como: nombres de identificaciones, indentaciones, espacios y longitud de líneas de código.

COMPLEJIDAD DEL SOFTWARE

Se define en el glosario de estándares de IEEE como "el grado en que un sistema o componente tiene un diseño o implementación que es difícil de entender y verificar" [12]. La complejidad del código puede verse afectada por muchos factores, tales como: líneas de código, número total de operadores y operandos, acoplamiento entre objetos y número de flujos de control [13]. Estos factores son utilizados en las métricas del software para la medición y la cuantificación aproximada de la complejidad del software.

La complejidad del software es considerada como una propiedad "esencial" del software, ya que refleja la complejidad del problema del mundo real con el que se enfrenta un software [6].

La legibilidad del código y la complejidad del software tienen un impacto sustancial en la calidad del software. **Para una mejor calidad, se desean baja complejidad y alta legibilidad**. La complejidad puede afectar la legibilidad del código, mientras que la baja legibilidad también puede resultar en una mayor complejidad percibida. Así, la legibilidad y la complejidad están relacionadas.

PREGUNTAS DE INVESTIGACIÓN

presentar un estudio empírico de las relaciones entre la legibilidad del código y la complejidad del software. Usando una variedad de métricas de legibilidad y complejidad en un nivel de granularidad fino, examinamos las relaciones entre estos dos atributos.

RQ1: ¿Qué tipo de relación existe entre la legibilidad del código y la complejidad del software?

RQ2: ¿Cuáles son las construcciones de código que afectan la legibilidad del código?

Se hace análisis de 35 programas Java que cubren 23 construcciones de programación distintas (por ejemplo, bucle, bucle anidado, condicionales).

Evidencia empírica para confirmar la teoría de que la complejidad y legibilidad del código están inversamente correlacionadas. Aplicando una técnica de aprendizaje automático, también identificamos construcciones de código que afectan sustancialmente la legibilidad del código.

1. TRABAJOS RELACIONADOS DE MÉTRICAS DE LEGIBILIDAD Y MÉTRICAS DE COMPLEJIDAD

1.1. Métricas de software

FACTOR DE LEGIBILIDAD : estructuras de código que las influencian

Métricas de Legibilidad:

* Buse & Weimer [9], [10] han propuesto una métrica de legibilidad de código y desarrollado una herramienta de legibilidad que mide automáticamente la métrica de legibilidad propuesta. Precisión de 80%, sobre fragmentos de código Java.

El estudio mostró que la legibilidad está fuertemente correlacionada con algunos atributos de calidad del software, tales como cambios de código, mensajes de registro de defectos e informes automáticos de defectos.

* Daryl Posnett et al. [16] argumentó que la legibilidad del código es una propiedad subjetiva y no es persuadible generar puntaje de legibilidad utilizando la herramienta de legibilidad automatizada. Además, incluyeron que la legibilidad depende en gran medida de la información contenida en el código fuente y, por lo tanto, el puntaje de legibilidad se puede calcular en función del tamaño y la entropía del código.
* Pahal et al (2017) realizaron una revisión de métricas para la calidad del software en la literatura, como la métrica Flesh-Kincaid, la métrica Gunning-Fog, el índice SMOG, el índice de legibilidad automatizada y el índice Coleman-Liau y concluyen que la elección de las métricas de legibilidad depende de diferentes empleos. Mencionan varios elementos de código que mejoran y degradan la legibilidad, por ejemplo, comentarios apropiados y variables mal definidas, respectivamente.

Métricas de complejidad

Todas las métricas de complejidad propuestas miden la complejidad en función de tres atributos: tamaño del software, flujo de datos y flujo de control. El conjunto de métricas Halstead Complexity Model [18], McCabe [19], Line of Code (LOC) [20] y Chidamber & Kemerer [21] son ejemplos de métricas de complejidad propuestas.

1.2. Análisis de la relación

Puneet et al. [7], la investigación se basó en Ingeniería de Software Basada en Componentes (CBSE). El estudio utiliza una métrica de complejidad que mide la complejidad de la interfaz para los componentes de software. Está fuertemente correlacionada con la legibilidad de los componentes de software. Los resultados indican una correlación negativa entre legibilidad y complejidad.

Buse y Weimer [14] propusieron un enfoque para construir una herramienta de legibilidad. Investigaron la correlación entre el puntaje de legibilidad de su herramienta de legibilidad propuesta y la complejidad ciclomática mediante el uso de la correlación de momento del producto Pearson. Descubrieron que la legibilidad está débilmente correlacionada con la complejidad en sentido absoluto, y efectivamente no está correlacionada con la complejidad en sentido relativo.

**DESCRIPCIÓN DE MÉTRICAS, HERRAMIENTAS Y CONJUNTOS DE DATOS USADOS**

**MÉTRICAS DE LEGIBILIDAD**

* El Índice de legibilidad automatizada (ARI):

ARI [28] se basa en la relación entre la dificultad de la oración y la dificultad de la palabra. La dificultad de las oraciones se determina como palabras por oración y la dificultad de las palabras es al calcular letras por palabras. La ecuación para calcular ARI es:

ARI = 4.71 (caracteres) + 0.5 (palabras) - 21.43

El valor numérico de la métrica ARI es calcular el nivel de grado necesario para comprender el texto. Por ejemplo, ARI = 3 significa que los estudiantes de 3er grado (de 8 a 9 años de edad) deberían poder comprender el texto [28].

* La medida simple de Gobbledygook (SMOG):

SMOG es sugerido por G Harry McLaughlin [22] en 1969. Esta métrica evaluó el tiempo (en años) requerido por cualquier persona para leer el texto. La ecuación para calcular SMOG es:

SMOG = 3 + Raíz cuadrada de recuento polisillable

El valor métrico SMOG significa un nivel de grado escolar de EE. UU. Que indica que un estudiante promedio en ese nivel de grado puede leer el texto [28]. Por ejemplo, SMOG = 7.4 indica que el texto es entendido por un estudiante promedio en séptimo grado.

* Índice de legibilidad de Flesch-Kincaid (FKI):

El valor del índice de legibilidad de Flesch-Kincaid [24] se calcula utilizando la siguiente fórmula:

206.835− 1.015 \* (𝑡𝑜𝑡𝑎𝑙 𝑤𝑜𝑟𝑑𝑠 / 𝑡𝑜𝑡𝑎𝑙 𝑠𝑒𝑛𝑡𝑒𝑛𝑐𝑒𝑠) - 84.6 \* (𝑡𝑜𝑡𝑎𝑙 𝑠𝑦𝑙𝑙𝑎𝑏𝑙𝑒𝑠 / 𝑡𝑜𝑡𝑎𝑙 𝑤𝑜𝑟𝑑𝑠)

El valor alto del índice Flesh-Kincaid (FKI) indica una alta legibilidad del código, mientras que un valor bajo implica que el código es difícil de leer. El rango de esta métrica es un número de 0 a 100. una puntuación más alta indica una lectura más fácil. Un documento promedio tiene un puntaje FKI entre 6 y 70. Como regla general, un estudiante promedio de 5to grado puede entender puntajes de 90-100. Los estudiantes de octavo y noveno grado pueden entender documentos con un puntaje de 60-70; y los graduados universitarios pueden entender documentos con un puntaje de 0-30.

* El índice de confusión de Gunning (GFI):

Otra métrica de legibilidad utilizada en este estudio es el Índice de confusión de Gunning, originalmente propuesto por Robert Gunning [23]. Utiliza la longitud promedio de las oraciones y el porcentaje de palabras difíciles. La ecuación para calcular el índice de confusión de Gunning es la siguiente:

GFI = 0.4 (ASL + PHW)

donde, ASL es la duración promedio de la oración y PHW es el porcentaje de palabras difíciles. Esta métrica es similar a la escala Flesch en que compara sílabas y longitudes de oraciones. Un puntaje GFI de 5 significa legible, 10 significa duro, 15 significa difícil y 20 indica muy difícil. Según su nombre, las palabras 'Foggy' son palabras que contienen 3 o más sílabas.

* Índice Coleman-Liau (CLI):

Meri Coleman y T. L. Liau [25] definieron otro índice de legibilidad como ARI, que se utiliza en este estudio para determinar el valor de legibilidad del código. Se calcula usando la siguiente fórmula:

CLI = 0.0588L - 0.296S - 15.8

donde, L y S son el número promedio de letras y oraciones respectivamente. Se basa en caracteres en lugar de sílabas por palabra y longitud de la oración. Un valor de CLI calculado utilizando la fórmula mencionada anteriormente significa una calificación de escuela secundaria. Por ejemplo, CLI = 10.6 para un texto dado significa que el texto puede leerse adecuadamente para un estudiante de secundaria de 10-11 grado.

* Puntuación de legibilidad de Buse (BRS):

El puntaje de legibilidad de Buse es la probabilidad de que un determinado código sea altamente legible. Buse y Weimer [14] desarrollaron un clasificador de aprendizaje automático de dos clases (es decir, de alta lectura y baja lectura) basado en una serie de construcciones de programas como características. BRS es la probabilidad de que un código determinado se clasifique en la clase de alta lectura.

METRICAS DE COMPLEJIDAD

Halstead Complexity Volume

McCabe’s Cyclomatic Complexity:

HERRAMIENTAS

* La herramienta de legibilidad Buse & Weimer [14]: para calcular el puntaje de legibilidad. Es un ejecutable de línea de comandos de Java.
* La segunda herramienta [29] es una aplicación web de código abierto, básicamente un analizador, para determinar: ARI, SMOG, índice de confusión de Gunning, índice de legibilidad Flesch-Kincaid e índice de Coleman-Liau.
* producto Pearson Coeficiente de correlación de momento [30]: Para determinar la existencia de estadísticamente significativo correlación entre métricas.
* WEKA [26]: herramienta que viene con la implementación de un colección de algoritmos de aprendizaje automático.

DATASET

mismo conjunto de datos que el trabajo de Tashtoush et al. [7] El conjunto de datos contiene 35 programas Java diferentes que cubren 23 construcciones de código distintas.

LEGIBILIDAD Y COMPLEJIDAD

Se analiza la correlación usando el **Coeficiente de correlación producto-momento de Pearson**,la cual permite examinar la relación lineal entre variables.

Dos variables se correlacionan positivamente cuando si una aumenta, la otra también aumenta. La correlación negativa entre variables implica que si una se hace más grande, la otra se hace más pequeña. la correlación va de -1 a +1, donde 0 es que no hay correlación entre las variables.

Respuesta a la pregunta 1: Existe una correlación negativa entre la legibilidad del código y la complejidad del programa.

CONSTRUCCIONES DE CÓDIGO AFECTAN LA LEGIBILIDAD

Construcciones de código:

1. Lines of Code (LOC)

2. Meaningful Names

3. Comment Indents

4. Indents

5. Scope

6. Inheritance

7. Polymorphism

8. Class Distribution

9. Spacing

10. Recursive

11. Fórmulas

12. Consistency

13. Line Length distribution

14. Identifier name Length

15. Identifier frequency

16. IF-else

17. Nested if

18. For Loop

19. While Loop

20. Do-While Loop

21. Nested -loop

22. Switch

23. Array

Se uso machine learning usando la herramienta WEKA. En el contexto de aprendizaje automático, cada programa Java se considera una instancia y cada construcción de código se considera una característica. Para la clasificación de características con WEKA, elegimos la selección de características basada en correlación, porque es una técnica popular para seleccionar los atributos / características más relevantes en un conjunto de datos.

RESULTADO: **La presencia de comentarios, el espaciado adecuado y los nombres significativos contribuyen a una alta legibilidad del código. El bucle while es mejor que el bucle do-while para una mejor legibilidad. El uso de matrices y bucles anidados no ayuda mucho en la legibilidad del programa.**

AMENAZAS A LA VALIDEZ DEL ESTUDIO

Los resultados de este estudio se derivan de un análisis de solo 35 pequeños programas escritos en Java. Por lo tanto, los resultados pueden no ser generalizables a programas escritos en otros lenguajes de programación y grandes sistemas industriales.

se utilizan solo seis métricas de legibilidad para evaluar la legibilidad del código, pueden no ser concluyentes, ya que la legibilidad es algo sujeto a la percepción humana y depende de la experiencia humana.

TRABAJOS FUTUROS

En el futuro, planeamos extender este trabajo incluyendo programas de diferentes tamaños y escritos en diversos lenguajes de programación para obtener un mejor punto de vista sobre la relación.

PROPUESTA: