SOFTWARE APPLIED TO THE MEASUREMENT OF CODING STANDARDS USING A MATHEMATICAL MODEL TO CALCULATE THE COHESION INDEX IN THE WEB DEVELOPMENT PRODUCTS

Cesar Yesid Barahona Rodríguez
Universidad De Cundinamarca
Facatativá, Cundinamarca, Colombia
cbarahona@ucundinamarca.edu.co

Héctor Hernán Castellanos Rodríguez, John Fredy Osorio Franco
Universidad De Cundinamarca
Facatativá, Cundinamarca, Colombia
hhcastellanos@ucundinamarca.edu.co
jfosorio@ucundinamarca.edu.co

***Abstract*— In search of automating processes in the development of quality, it is proposed to create a software product that allows to carry out the process of evaluation of the standards of codification in the files of source code. For this it is investigated on the standards that exist today, its benefits and characteristics, besides this is designed a metric of evaluation based on a mathematical model, in order that this is used by the software to be able to calculate the cohesion index between the developed software and the evaluated code with the objective of being able to determine if the code complies with the norms established by the standard selected in this investigation.**

Keywords*— *Coding standards, Software quality, Maintainability, Code integrity, Software products

I. INTRODUCTION

En la industria del software uno de los aspectos que representa un mayor costo en el proceso de construcción de software, es el de realizar mantenimiento al código fuente, esto puede deberse a varios aspectos, como pueden ser el no utilizar una arquitectura adecuada, el no seguir un proceso de desarrollo adecuado, el no aplicar estándares de codificación, entre otras razones. En este artículo se busca enfatizar en este último aspecto, el no utilizar estándares de codificación, en muchas fábricas de software, no se exige a los desarrolladores que apliquen un estándar en específico, si no se deja la libertad total al desarrollador de codificar de la manera que considere correcta, es por esto que cuando es necesario realizar cambios sobre el mismo se dificulta este proceso debido a la variación que puede existir en cada uno de los ficheros que componen el proyecto, a razón de esto en ocasiones es necesario reescribir el fichero al no entender el mismo lo que duplica el tiempo y costo del desarrollo.

En busca de mitigar este aspecto se realiza un proceso investigativo y de desarrollo, con la cual se quiere resaltar la importancia de usar un estándar de codificación y a través de esta investigación encontrar un estándar que sea utilizado en la actualidad, esto con el fin de diseñar un modelo matemático que permita calcular un índice de cohesión del software, para proseguir con el desarrollo de un software que permita calcular este índice en los ficheros de código fuente escritos por los desarrolladores y así poder emitir conclusiones sobre el mismo para poder proceder a la aprobación o rechazo del mismo.

II. IMPORTANCE OF PROGRAMMING STANDARDS IN QUALITY SOFTWARE

Los documentos de estándares de programación definen el proceso de codificación como prácticas de matrices hacia una clasificación lo cual se debe adherir los estudiantes al desarrollar código. Existe una norma ISO para los lenguajes de programación en C que permite la implementación por parte de los compiladores y así mismo no todos los compiladores cumplen los estándares de programación, al enseñar a los desarrolladores a diseñar código a un cierto estándar, se promoverán los conceptos de confiabilidad de código, facilidad de mantenimiento y portabilidad, ya que el estudiante tendrá la capacidad de estar preparado para un ambiente de negocios.

"Un estudio demostró empíricamente que la violación de las normas de los estándares, es impacto negativo en un software de fiabilidad"[1].

Durante el paso de los años los estándares de codificación se han vuelto populares como un medio para asegurar la calidad de software durante el proceso de desarrollo de un software, se aseguran un estilo común de programación, lo que aumenta la capacidad de mantenimiento y evitan el uso de construcciones potencialmente generadoras de problemas, aumentando así la confiabilidad. Las reglas en tales estándares se basan principalmente en la opinión experta y ganada por los años con una cierta lengua en varios contextos.

Los autores Zhenmin Li y Lin Tan en su investigación "Un estudio empírico de las características de los errores en el software de código abierto moderno" argumentan que los primeros controles automatizados han contribuido a la fuerte disminución de los errores de memoria presentes en el software. Sin embargo, la disponibilidad de normas y herramientas adecuadas, existen varias cuestiones hacen obstáculo, Por ejemplo, el 30% de las líneas de uno de los proyectos utilizados en este estudio contenían tal violación. Las infracciones pueden ser subproductos del análisis estático, que no siempre puede determinar si el código viola o no una determinada comprobación.[2]

"Cualquier modificación del software tiene una probabilidad no nula de introducir un nuevo fallo, y si esta probabilidad excede la reducción obtenida al fijar la infracción, el resultado neto es una mayor probabilidad de fallas en el software"[3].

El análisis de requisitos para el diseño se captura en el campo de comentario como lo hace la noción de gestión de configuración y plan de prueba de verificación proporcionando al estilo de diseño de código un estándar para la legibilidad, comprensión y facilidad de mantenimiento. Para evaluar el conocimiento de los aprendices sobre el análisis de los requisitos a través del diseño, el uso y la corrección de los comentarios son fundamentales. Esto se ajusta a las mejores prácticas, Las revisiones de código se utilizan como proceso para reforzar la capacidad de mantenimiento y la conformidad.[1]

Final mente los aspectos más representativos de un estándar de codificación hace limitar al lenguaje a un conjunto seguro para evitar el uso de instrucciones inseguras, reduciendo eficazmente los defectos del código y mejorando la calidad de software, una de las normas de codificación ampliamente aceptadas para el software es MISRA-C.

MISRA-C es un conjunto de directrices para el uso de programación en este caso en C, primeramente, publicado en 1998 y realizando así una nueva versión en 2004, contiene 141 reglas de estándares de programación conveniente para el desarrollo crítico. Aborda directamente un diseño inadecuado para mejorar la capacidad del sistema, ofreciendo un servicio confiable y logrando de esta manera ser aceptado por industrias y aplicaciones como lo son: militar, automóvil, aeroespacial, medico etc.

Para la comprobación de reglas relacionadas con las variables MISRA-C desensamblada el código fuente siendo implícito lo que hace que logre representar las variables de maneras operadas en código de desensamble, como resultado puede colocarse en un registro general o en una pila y referenciada por su dirección, siendo total mente diferente al código fuente. Una variable de código fuente hace desaparecer el código de desensamble debido a las optimizaciones del compilador, todo esto significa que algunas reglas de estándar de codificación con las variables no pueden ser verificadas.[4]

III. MODULE FOR THE EVALUATION OF CODING STANDARDS UNDER THE SOFTWARE QUALITY METHODOLOGY

Algunas razones por la cual se espera que los desarrolladores de la universidad de Cundinamarca adopten y sigan los estándares de codificación no es solo la familiarización con el proceso de desarrollo, si no que se promueva la legibilidad, transparencia, organización y portabilidad del código fuente, con el propósito que el código pueda mantenerse, reutilizarse y reducir costos de desarrollo de software. Beneficiando a los grupos de desarrolladores o a una organización más grande para reducir tiempo requerido por las personas para comprender o revisar el trabajo de sus compañeros, centrándose así en el enfoque de los proyectos informáticos orientados a la tecnología web los cuáles deben ser diseñados y construidos con los estándares internacionales de calidad.[5]

Es por eso por lo que el módulo que se presenta a continuación Como apoyo a la actividad investigativa desarrollada en el programa de ingeniería de sistemas, extensión Facatativá es un anexo al software de calidad llamado Calisoft el cual se fundamenta en una plataforma de evaluación para los productos de software que se desarrollan en la universidad que se basan en la realización de software mediante tres (3) sistemas de calificaciones. El primer sistema de calificación es en donde se presenta al evaluador la documentación y modelación del proyecto, el segundo sistema es la parte en donde se evalúa la gestión de pruebas tanto funcionales como prueba de carga y estrés, la tercera herramienta cuenta con la parte administrativa en donde se hacen las configuraciones de acuerdo con los estándares de calidad.

El módulo que se desarrollo es un complemento al sistema de evaluación de gestión de pruebas, lo cual mejorara el sistema de calificación de la plataforma de Calisoft, este complemento dará un gran soporte ante los estándares de programación que se rigen actualmente en el mundo, se busca que con la implementación de este módulo los evaluadores puedan garantizar que se esté respetando el estándar de codificación previamente establecido, mientras que el módulo de proyecto de estandarización determine una calificación hacia el desarrollador, de esta manera se puede constatar en que posibles errores o en qué aspectos el desarrollador está fallando a la hora de codificar, recordando la importancia que tiene este ítem para garantizar un software de calidad.

Así mismo para la garantía de la realización de este módulo se basó en 3 puntos importantes

- Investigación previa con el objetivo de realizar una documentación y seleccionar un estándar de codificación adecuado
- Construcción de una Métrica de evaluación

- Desarrollo del módulo para la evaluación de estándares de codificación

IV. CODING CRITERIA

Antes de empezar primero se realizó una investigación predefiniendo unos requisitos principales para saber la instancia y la magnitud del proyecto los cuáles son

- Generar los parámetros de estándares
- Evaluar declaración de variables, clases, funciones, contantes, indentación, declaración de comentarios y espacios de nombre.
- Creación de la métrica de evaluación con las anteriores directrices para crear un sistema de calificaciones de manera cuantitativa.
- Creación de la interfaz gráfica.
- Visualización del código fuente del desarrollador.
- Realización de informes de calificación del módulo de evaluación de estándares de codificación.

Los desarrollos de los productos de software deben estar soportados por normas de calidad que hayan sido emitidas por organizaciones internacionales como la ISO teniendo en cuenta que la evaluación generada cumpla con las normas internacionales como la ISO/IEEE 25010, buscando establecer las pautas mínimas que debe cumplir un software de calidad. La norma ISO/IEEE 25010 establece un modelo de calidad de software, un sistema de calificación de los productos, determinando las características de calidad que se tienen en cuenta a la hora de evaluar las propiedades de un producto de software determinado.

El modelo de calidad del producto definido por la ISO/IEC 25010 se encuentra compuesto por las ocho características de calidad.

- Adecuación función
- Eficiencia de desempeño
- Compatibilidad
- Usabilidad
- Fiabilidad
- Seguridad
- Mantenibilidad
- Portabilidad

El cual los productos de software están ligados por la parte de la mantenibilidad el cual representa su modificación efectiva y eficiente debido a las necesidades evolutivas o perspectivas.[6]

Por último, se escogió un estándar de codificación base que por el cual los estudiantes de la universidad de Cundinamarca deben realizar su respectiva lectura, análisis y seguimiento para poder realizar una evaluación correspondiente de sus productos de software correspondientes a la codificación son las recomendaciones del estándar de PHP (PHP Standars Recommendations) Es un acrónimo de la recomendación de normas de PHP lo cual contienen dentro de su formato PSR-1, PSR-2, PSR-3 y PSR-4 mediante el cual son las recomendaciones de PHP-FIG. Sus nombres comienzan con PSR- y terminan con un número. Cada recomendación de PHP-FIG resuelve un problema específico que con frecuencia se encuentra en la mayoría de los frameworks de PHP[7].

- PSR-1: Estilo básico de código
- PSR-2: Estilo de código estricto
- PSR-3: interfaz Logger
- PSR-4: Autoloaders.

V. EVALUATION METRIC

Observando lo anteriormente expuesto sobre estándares de codificación y los beneficios que brinda el utilizar estándares al momento de realizar desarrollos de software, es importante analizar cada una de las directrices que propone el estándar explicado como lo es el PSR. Este estándar propone una serie de normas que aplicarlas al momento de codificar nuestro software permite que este sea más limpio y pueda ser fácilmente entendido por otro programador; Analizando las directrices que expone el estándar, se busca identificar cuáles de estas son de mayor utilidad y tienen mayor peso para lograr el objetivo de obtener un código de calidad y que pueda ser fácilmente mantenido.

Partiendo de los supuestos anteriores después de identificar que directrices son de mayor importancia, se busca elaborar la rúbrica de evaluación teniendo como categorías o elementos de evaluación cada una de las normas del estándar especificado y asignándole un porcentaje de calificación de acuerdo a la importancia o peso considerado, posteriormente se definirá los niveles de calificación para cada una de las categorías, estos deben ser cuantificables y medibles, con esto se busca generar una calificación la cual pueda estar basada en normas y estándares usados en la industria de software actual.

Con el diseño de esta métrica se busca poder evaluar al programador para verificar si está aplicando el estándar de la forma correcta, además el poder ser calificado permite al programador poder identificar en qué aspectos está presentando falencias al momento de codificar, con esto se busca lograr que el software entregado sea mucho mejor elaborado.

Para la construcción de la métrica de evaluación las categorías que se van a evaluar son algunas de las directrices propuestas por el estándar, las cuales son:

- Declaración de variables.
- Declaración de métodos.
- Declaración de clases.
- Indentación de las estructuras de control.

- Uso de comentarios en el código fuente.
- Declaración de constantes.
- Uso de espacios de nombre.

A cada una de estas directrices se le asigna una prioridad de acuerdo con su grado de relevancia dentro del estándar propuesto, con esto se busca que las directrices que son usadas con mayor frecuencia y que pueden ser de mayor importancia para lograr el correcto uso del estándar tengan un mayor porcentaje al momento de obtener la calificación final.

Table 1. Priority of each guideline.

Directrices	Prioridad		
	Alta	Media	Baja
Variables	X		
Métodos	X		
Clases	X		
Indentación		X	
Comentarios		X	
Constantes			X
Espacios de nombre			X

Observando la tabla 1. Se pueden identificar tres niveles de prioridades alta, media y baja. Cada una de estas tendrá un valor numérico que estará en el rango de 1 a 5 siendo las directrices de prioridad alta las que posean un mayor valor.

La forma de realizar el cálculo de la calificación final se basará en obtener la nota individual de cada una de las normas, para luego consolidarlas en una única ecuación y obtener la nota correspondiente, para hacer este cálculo se utiliza una ecuación para cada una de las directrices anteriormente mencionadas, las cuales se presentan a continuación:

$$Nv = \frac{\sum_{N=1}^{NTA} \frac{NVC}{NVT}}{NTA} \quad (1)$$

NTA: Número total de archivos de código de fuente donde se hayan declarado variables.

NVC: Numero de variables correctamente declaradas.

NVT: Numero de variables totales declaradas por archivo de código fuente.

Nv: Calificación para la directriz declaración de variables.

$$Nm = \frac{\sum_{N=1}^{NTA} \frac{NMC}{NMT}}{NTA} \quad (2)$$

NTA: Número total de archivos de código de fuente donde se hayan declarado métodos.

NMC: Numero de métodos correctamente declarados.

NMT: Numero de métodos totales declarados por archivo de código fuente.

Nm: Calificación para la directriz declaración de métodos.

$$Nc = \frac{\sum_{N=1}^{NTA} \frac{NCC}{NCT}}{NTA} \quad (3)$$

NTA: Número total de archivos de código de fuente donde se hayan declarado clases.

NCC: Numero de clases correctamente declarados.

NCT: Numero de clases totales declaradas por archivo de código fuente.

Nv: Calificación para la directriz declaración de clases.

$$Ni = \frac{\sum_{N=1}^{NTA} \frac{NIC}{NIT}}{NTA} \quad (4)$$

NTA: Número total de archivos de código de fuente.

NIC: Numero de estructuras de control correctamente indentadas.

NIT: Numero de estructuras de control utilizadas por archivo de código fuente.

Ni: Calificación para la directriz de indentación de estructuras de control.

$$Ncm = \frac{\sum_{N=1}^{NTA} \frac{CMC}{CMT}}{NTA} \quad (5)$$

NTA: Número total de archivos de código de fuente.

CMC: Numero de estructuras de código comentadas.

CMT: Numero de estructuras utilizadas por archivo de código fuente.

Ncm: Calificación para la directriz uso de comentarios en el código fuente.

$$Ncs = \frac{\sum_{N=1}^{NTA} \frac{CSC}{CST}}{NTA} \quad (6)$$

NTA: Número total de archivos de código de fuente donde se hayan declarado constantes.

CSC: Numero de constantes correctamente declarados.

CST: Numero de constantes totales declaradas por archivo de código fuente.

Ncs: Calificación para la directriz declaración de constantes.

$$Ne = \frac{\sum_{N=1}^{NTA} \frac{NSC}{NST}}{NTA} \quad (7)$$

NTA: Número total de archivos de código de fuente donde se hayan declarado espacios de nombre.

NSC: Numero de espacios de nombre correctamente declarados.

NST: Numero de espacios de nombre declarados por archivo de código fuente.

Ne: Calificación para la directriz de uso de espacios de nombre.

El cálculo de la nota final se realizará aplicando una única ecuación en la cual se consolidarán cada una de las notas obtenidas para cada directriz y estas tendrán un peso dentro de la ecuación de acuerdo con su a nivel de prioridad.

$$\frac{Vp(Nv) + Vp(Nm) + Vp(Nc) + Vp(Ni) + Vp(Ncm) + Vp(Ncs) + Vp(Ne)}{Vp(Npa) + Vp(Npm) + Vp(Npb)} \quad (8)$$

Vp: Valor de la prioridad de cada directriz.

Npa: Número de directrices de prioridad alta.

Npm: Número de directrices de prioridad media.

Npb: Número de directrices de prioridad baja.

La ecuación (8) se utiliza para poder calcular el índice de cohesión, para esto se necesita el valor obtenido para cada una de las directrices anteriormente explicadas, por ello cada uno de estos valores se multiplica por su respectiva prioridad y se sumaran, para luego ser divididos por el total de las directrices utilizadas en la codificación. El índice de cohesión busca medir a que nivel se está utilizando el estándar al momento de codificar, este valor puede ser representado como un porcentaje y así poder evaluar al programador sobre la aplicación de esta norma de calidad.

VI. DEVELOPMENT AND OPERATION OF THE MODULE

En la etapa final de este trabajo de investigación se realizará el desarrollo de una herramienta que tome como base el estándar que se seleccionó anteriormente y evalué de acuerdo con la métrica expuesta en el presente artículo, los archivos de código fuente que componen el proyecto que se esté desarrollando; permitiendo calcular el índice de cohesión, para poder emitir conclusiones de si el desarrollo cumple con las medidas de calidad deseadas para el producto software.

El funcionamiento de esta herramienta se enfoca en el análisis de código fuente para determinar si el estándar fue aplicado en un nivel, que sea aceptable para las normas de calidad deseadas para el producto. Es por esto por lo que esta herramienta tendrá una arquitectura lineal la cual incluirá varios componentes, estos a su vez tendrán como salida objetos que serán la entrada para el siguiente componente, esto con el fin de procesar el código y poder calcular el índice de cohesión para el proyecto.

Los componentes en los cuales está dividida la herramienta buscan filtrar y clasificar los elementos del código para procesarlos individualmente y evaluarlos de acuerdo con el estándar.

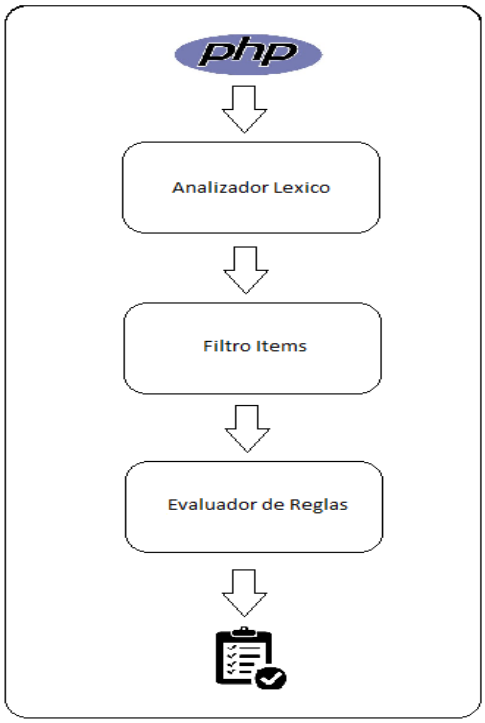

Fig. 1. Module Architecture

Como se puede observar en la Fig. 1. La herramienta está constituida por tres componentes, cada uno de estos recibe un elemento de entrada y a su vez entregará otro que servirá de entrada para el componente siguiente. Cada uno de estos componentes tiene un funcionamiento y objetivo diferente los cuales se presentan a continuación:

A. Lexical Analyzer.

Este componente recibe como entrada el código fuente que está escrito en lenguaje PHP. El proceso que realiza es reconocer las palabras y elementos que hacen parte del léxico del lenguaje, a su vez se encarga de clasificarlos en grupos, según la función que cumplen dentro del lenguaje. Cada vez que el analizador lee un elemento nuevo, crea un token el cual es un objeto que contiene información acerca de ese elemento como el grupo en que se clasifico, valor y demás información que será necesaria para los procesos siguientes, una vez que el token ha sido creado se anexa a lista de tokens.
Una vez que el analizador termina de realizar la lectura del fichero, procede a entregar la lista de tokens creada, al siguiente componente el cual se encargara de continuar el proceso.

B. Items Filter

Este componente recibe como entrada la lista de tokens entregada por el componente anterior. Este se encarga de filtrarlos de acuerdo con el ítem o directriz de evaluación especificados en la métrica de evaluación. A medida que filtra cada uno de los tokens los anexa a una estructura ordenada y elimina los demás elementos que no son necesarios para la evaluación. Una vez termina de recorrer la lista, este componente se encarga de entregar al siguiente una estructura reducida y ordenada para que este realice el proceso final de evaluación.

C. Rules Evaluation

El proceso de este componente inicia recibiendo del anterior, una estructura ordenada de acuerdo con los ítems de evaluación establecidos. Esta toma cada uno de los elementos de la estructura, verifica a que directriz de evaluación pertenece y realiza el proceso de evaluación según sean las normas establecidas para esa directriz, si este componente aprueba las normas de evaluación se procede a computarlo según se estableció en la métrica de evaluación. Una vez este termina la validar cada uno de los elementos, genera un resultado de evaluación y lo entrega como salida del módulo.
Este proceso se repite para cada uno de los ficheros que se van a evaluar, una vez que se han evaluado todos, se realizara el cálculo del índice de cohesión el cual es el objetivo final de todo este proceso.

VII. CONCLUSION

Se pudo identificar qué beneficios trae el utilizar estándares de codificación como el incremento en la fiabilidad del software, la disminución en el tiempo de ejecución, entre otros. A su vez se pudo seleccionar un estándar de codificación como lo es el PSR para continuar el proceso de investigación.

La construcción de un modelo matemático que permita calcular el índice de cohesión se pudo realizar implementando un esquema basado en directrices jerarquizadas en prioridades, basándose en su frecuencia de uso e importancia dentro del estándar.

El desarrollo del módulo se pudo realizar implementado una arquitectura lineal y utilizando herramientas de software libre. Un siguiente paso en este proceso será realizar pruebas al mismo sometiéndolo a diferentes escenarios de prueba para determinar el nivel de certeza que este pueda brindar.

REFERENCES

[1] D. V. MacKellar, “Injection of Business Coding Standards Practices to Embedded Software Courses Donald,” in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016, pp. 1–8.

[2] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, “Have things changed now?,” *Proc. 1st Work. Archit. Syst. Support Improv. Softw. dependability - ASID ’06*, pp. 25–33, 2006.

[3] C. Boogerd and L. Moonen, “Using software history to guide deployment of coding standards,” *Trader Reliab. high-volume Consum. Prod.*, pp. 39–52, 2009.

[4] Z. Dai, X. Mao, D. Wang, D. Liu, and J. Zhang, “Checking compliance to coding standards for x86 executables,” *Proc. - Symp. Work. Ubiquitous, Auton. Trust. Comput. Conjunction with UIC 2010 ATC 2010 Conf. UIC-ATC 2010*, no. November 2010, pp. 449–455, 2010.

[5] I. Acosta, E. Nieto, and C. Barahona, “Metodología para la evaluación de calidad de los productos software de la Universidad de Cundinamarca,” vol. 3, no. 2.

[6] P. Roa, C. Morales, and P. Gutierrez, “Norma ISO / IEC 25000,” *Univ. Dist. Fr. Jose Caldas*, vol. 3, no. 2, 2015.

[7] J. Lockhart, *Modern PHP New Features and Good Practices*. 2015.