Caracterización de estándares de codificación aplicando una métrica de evaluación para medir el nivel de cohesión en un software de calidad

*Ing.Cesar Yesid Barahona Rodríguez, **Hector Hernan Castellanos,***John Fredy Osorio
*Facultad de ingeniería, Universidad de Cundinamarca ,Programa Ingeniería de sistemas
Facatativá,Cundinamarca,Colombia
cbarahona@ucundinamarca.edu.co
**Facultad de ingeniería, Universidad de Cundinamarca, Programa Ingeniería de sistemas
Facatativá,Cundinamarca,Colombia
hhcastellanos@ucundinamarca.edu.co
***Facultad de ingeniería, Universidad de Cundinamarca, Programa Ingeniería de sistemas
Facatativá,Cundinamarca,Colombia
jfosorio@ucundinamarca.edu.co

Resumen

PHP Standars Recommendations (PSR) es un estandar de codificación que es ampliamente utilizado en la industria de software; al implementar un estándar como esté, permite obtener un código organizado y mejor desarrollado al cual se le pueda realizar mantenimiento de una forma sencilla y rápida, este estandar se puede caracterizar con el objetivo de encontrar directrices para evaluar el grado de cohesion entre el código impletandado y la asimilacion de estandares de codificacion, con la finalidad de obtener un producto de software con caractersiticas de calidad. Es por esto que se busca diseñar una métrica de evaluación que permita identificar cuales de estas directrices representan una mayor importancia, con el objeto de generar una valoración de la aplicación del estándar.

Abstract

PHP standards Recommendations (PSR) is an encoding standard that is widely used in the software industry; When implementing a standard like this, Allows you to obtain an organized and better developed code that can be easily and quickly maintained, This standard can be characterized with the objective of finding guidelines to evaluate the degree of cohesion between the implemented code and the assimilation of coding standards, With the purpose of obtaining a software product with quality characteristics. This is why it is sought to design an evaluation metric that allows identifying which of these guidelines represent a greater importance, in order to generate a valuation of the application of the standard.

1. Introducción

Hoy en día una de las áreas que presenta mayor crecimiento y constante cambio es la industria tecnológica, cada día nacen más empresas que deciden tomar como actividad económica el desarrollo de software o actividades relacionadas con el sector tecnológico, es por esto que buscan dar un valor agregado a sus productos y/o servicios, para lograr esto muchas de estas deciden utilizar estándares de calidad como lo es la norma ISO/IEC 25000 conocida como requisitos y evaluación de calidad de productos de software, una de estas características es la mantenibilidad que se puede explicar cómo la capacidad que tiene un sistema para poder realizar cambios sobre sus componentes de una manera efectiva.[1] Cabe resaltar que muchos de estos cambios no lo realiza la persona que creó el software en un principio; es por esto que las empresas deciden aplicar estándares de codificación en sus trabajos para que estos den la impresión que han sido hechos por un solo programador y puedan ser comprendidos en un futuro por otros programadores.

Por lo tanto, se propone desarrollar una herramienta que evalué el estándar de codificación previamente establecido por la institución en los proyectos de desarrollo de software, además que esta pueda generar una calificación en base a una métrica de evaluación previamente diseñada, así se busca lograr que los productos informáticos desarrollados en la industria del software cumplan con criterios de alta calidad orientados a estándares de codificación.

2. Estandares de programacion

Los estándares de programación son una parte integral de un software, según MacKellar intenta introducir este concepto a través de la aplicación y la educación mediante el documento de normas de programación. Se proporciona en medio de conferencias integradas con ejemplos adecuadamente, los estándares definidos en el documento y las asignaciones mejoradas para el proceso de desarrollo llamado "código de revisión por pares", el cual se centra en la capacidad que tiene el estudiante para innovar dentro de un conjunto de restricciones definidas externamente que tienen como objetivo simular un ambiente de negocio[2].

Al concepto de estándares de programación se considera importante está línea de seguimiento ya que se opta por especificar que estándares de desarrollo son adecuados para el modulo que se quiere realizar, no obstante, se hace una pregunta importante.

¿Porque es importante tener Buenas practicas al escribir código?

La realimentación hacia una evaluación continua y automática a través de las prácticas de programación proporciona información inmediata al desarrollador de los errores cometidos en codificación, permitiendo que puedan ser corregidas y reevaluadas repetidas veces, estableciendo un marco de evaluación donde la persona pueda aumentar su eficiencia en el momento de estar sentado escribiendo código. Según J. C. Rodríguez, estos son los siguientes errores por el cual son susceptibles para ser identificados inmediatamente

- Incumplimiento de interfaz
- Errores de estilo
- Diseño de codificación
- Errores de funcionalidad
- Errores graves de ejecución
- Insuficiencias en las pruebas[3]

Un ejemplo de la utilización de estándares de programación entorno a las empresas es asegurar que los programadores profesionales requieran normas de codificación más seguras y confiables, convirtiéndose en una situación que es muy frecuentemente uno de los problemas más importantes y que a menudo son descuidados en las empresas, así mismo cuando se tiene malos hábitos de escribir mal el código, por ejemplo, no comentar y sin embargo no conocer ningún tipo de estándares de programación, no se podrá entender de manera correcta lo que un desarrollador desea plasmar en su creación de software, tal es el caso que puede haber requisitos de codificación dependientes del lenguaje que requieran atención especial para evitar errores en la funcionalidad un programa.

Para empezar con las normas y reglas de estandarización del lenguaje de PHP primero se realiza una pregunta:

¿Qué es un PSR?

PSR (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.

A. PSR-1: Estilo básico de código

Está sección de la norma considera los elementos de codificación estándar proporciona pautas sencillas que son fáciles de implementar con un mínimo esfuerzo, debe cumplir estos requisitos para ser compatible con PSR-1:

- Etiquetas PHP: Lo cual sugiere redondear el código escrito en PHP con las etiquetas <? Php?> O <? =?>
- Codificación: Todos los archivos de PHP deben codificarse con el conjunto de caracteres UTF-8 que, por tal caso, un editor de texto más estructurado y completo lo hace automáticamente.
- Objetivo: Un solo archivo PHP puede definir símbolos (una clase, rasgo, función, constante, etc.) o realizar una acción que tiene efectos secundarios (por ejemplo, crear salida o manipular datos). Un archivo PHP no debería hacer ambas cosas. Esta es una tarea sencilla y requiere sólo un poco de previsión y planificación de su parte.
- Nombres de clases: el formato para el nombre de clases es *CamelCase*, refiriéndose al estilo de escritura que se aplica a frases o palabras compuestas, esto significa que el primer carácter del nombre del método está en minúsculas y la primera letra de cada palabra subsiguiente en el nombre del método es mayúscula.
- Nombres de las constantes: las constantes de PHP deben utilizar todos los caracteres en mayúsculas y utilizar subrayados para separar palabras.
- Nombres de métodos: de igual manera que los nombres de clase, los métodos deben tener el formato *CamelCase*.

B. PSR-2: Estilo de código estricto

Después de implementar PSR-1, el siguiente paso es implementar PSR-2. El estándar PSR-2 define el estilo de código PHP con pautas más estrictas.Un estilo de código estricto común permite a los desarrolladores escribir código que es fácilmente y rápidamente entendido por otros contribuyentes. A diferencia del PSR-1, la recomendación del PSR-2 contiene directrices más estrictas.

- Cada línea de código no debe exceder los 80 caracteres.
- Cada línea no debe tener espacio en blanco.
- Palabras claves en minúscula.
- Los espacios de nombres deben ir seguidos de una línea en blanco.
- **Clases:** La recomendación PSR-2 indica que el soporte de apertura de una definición de clase debe residir en una nueva línea inmediatamente después del nombre de definición de clase
- **Métodos:** La colocación de corchetes de definición de método es la misma que la colocación de corchetes de definición de clase.
- **Vistas:** Debe declarar una visibilidad para cada propiedad de clase y método. Una visibilidad es pública, protegida o privada; La visibilidad determina cómo una propiedad o método es accesible dentro y fuera de su clase.
- **Estructuras de control:** todas las palabras clave de la estructura de control deben ir seguidas de un solo carácter de espacio. Una palabra clave de estructura de control es **if, elseif, else, switch, case, while, do while, for, foreach, try o catch.** Si la palabra clave de estructura de control requiere un conjunto de paréntesis, asegúrese de que el primer paréntesis no esté seguido por un carácter de espacio y asegúrese de que el último paréntesis no esté precedido por un carácter de espacio.

C. PSR-3: interfaz Logger

PSR-3 es una recomendación y prescribe métodos que pueden ser implementados por los componentes del registrador de PHP, describiendo una interfaz común para registrar bibliotecas y escribir registros de una manera simple y universal.

Un componente logger de PHP compatible con la recomendación del PSR-3 debe incluir:

- Una clase PHP que implemente la interfaz denominada.
- Un componente Monolog PHP que implementa completamente la interfaz PSR-3, y se amplía fácilmente con formateadores de mensajes personalizados y controladores.

- Un manejador de mensajes de Monolog que permite enviar mensajes de registro a archivos de texto, syslog, correo electrónico, HipChat, Slack, servidores en red, APIs remotas, bases de datos y prácticamente cualquier otro lugar que pueda imaginar.
- En el evento muy improbable Monolog no proporciona un controlador para el destino de salida deseado, lo cual es super fácil escribir e integrar su propio controlador de mensajes Monolog.

D. PSR-4: Autoloaders

¿Qué pasa si necesita incluir un millar de scripts PHP? o ¿Qué pasa si necesita incluir un centenar de scripts PHP? Las funciones require () e include () no escalan bien y esta es la razón por la cual los autoloaders de PHP son importantes. Un autoloader es una estrategia para encontrar una clase, interfaz o rasgo de PHP y cargarlo en el intérprete de PHP bajo demanda en tiempo de ejecución, sin incluir explícitamente los archivos describiendo una estrategia de autocargador estandarizada.

Los componentes y frameworks de PHP que soportan el estándar del autocargador PSR-4 pueden ser localizados y cargados en el intérprete de PHP con un solo autocargador [4].

3. Metrica de evaluacion

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 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 metrica 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 a 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.

Tabla1: Prioridad de cada directriz.

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

Observando la tabla anterior 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.

3.1 Ecuaciones

La forma de obtener el índice de cohesion final se basará en obtener la calificación 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:

$$Nv = \frac{\sum_{N=0}^{NTA} \frac{NVC}{NVT}}{NTA} \quad (1)$$

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.

$$Nm = \frac{\sum_{N=0}^{NTA} \frac{NMC}{NMT}}{NTA} \quad (2)$$

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.

$$Nc = \frac{\sum_{N=0}^{NTA} \frac{NCC}{NCT}}{NTA} \quad (3)$$

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.

$$Ni = \frac{\sum_{N=0}^{NTA} \frac{NIC}{NIT}}{NTA} \tag{4}$$

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.

$$Ncm = \frac{\sum_{N=0}^{NTA} \frac{CMC}{CMT}}{NTA} \tag{5}$$

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.

$$Ncs = \frac{\sum_{N=0}^{NTA} \frac{CSC}{CST}}{NTA} \tag{6}$$

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.

$$Ne = \frac{\sum_{N=0}^{NTA} \frac{NSC}{NST}}{NTA} \tag{7}$$

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 del indice final se realizara aplicando una única ecuación en la cual se consolidaran cada una de las calificaciones obtenidas para cada directriz y estas tendrán un peso dentro de la ecuación de acuerdo a su a nivel de prioridad.

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

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.
Ic: Indice de cohesion.

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.

4. Importancia de los estandares de programacion en un software de calidad

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”[2].

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[5].

Tomando otro ejemplo como referencia, encontramos que una sola regla es responsable del 83% de todas las violaciones en uno de los proyectos analizados, lo cual es poco probable que solo señale los verdaderos problemas.

“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”[6].

El término plagio en el software es difícil de distinguir, Sin embargo, a través del uso de esta norma específicamente en las secciones que tratan con comentarios e identificadores, se hace evidente y más fácil distinguir entre el diseño de problemas

de grupo que comparte (que es el aprendizaje) y la copia de código. Además, el uso de los exámenes y revisiones de código de compañeros resaltará a los estudiantes que no intentan aprender.

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[2].

Finalmente 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 con fiable 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 totalmente 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[7].

5. Conclusiones

A través del escrito hecho, se puede llegar a dos interrogantes importantes, que concluyen la escritura de este artículo.

¿Qué reglas usar para el estándar?

Un estándar puede ser simplemente ampliamente adoptado, pero todavía contienen reglas que, a simple vista no parecen las más adecuadas para un proyecto, además la simple existencia de diferentes estándares de codificación, se evalúa explícitamente las reglas individuales que pueden ayudar a valorar el estándar correcto.

¿Cómo priorizar una lista de violaciones?

Aunque un estándar puede ser cuidadosamente elaborado y personalizado para un proyecto, los desarrolladores se enfrentan a demasiadas violaciones fijas, dado el tiempo limitado. Para manejar este problema de la manera más eficiente, se hace énfasis en una lista de violaciones, para definir un umbral para determinar cuáles deben ser abordados, y cuáles pueden omitirse.[8].

¿Por qué es importante implementar un estándar de codificación como lo es el PSR?

Hoy en día se hace relativamente difícil en seguir correctamente los estándares de programación especialmente en extensos proyectos de software; el cumplimiento de una norma de codificación se trata en este caso una prueba de aprobación o rechazo, sin embargo, es posible que se convierta en un enfoque diferente donde se mide el nivel de cumplimiento, esto permitirá correlacionar con las mediciones de los productos de software. Contar el número de violaciones de tales reglas en un producto de Software parece estar bien fundamentado e intuitivamente corresponde a una medida de su calidad interna.[9]

Por tal motivo en los trabajos de desarrollo en los que se ha participado, se identifica que, al realizar una colaboración con una mayor cantidad de miembros en el equipo de trabajo, cada uno de estos tiene un estilo de codificación totalmente diferente al de los demás, esto logra dificultar y entorpecer la compresión y análisis del código, por lo tanto una recomendación que logra mitigar este tipo de inconvenientes es implementar normas de programación actuales, como lo

es el PSR siendo un estándar actual utilizado por PHP y frameworks como laravel, para lograr un estilo uniforme en cada uno de los scripts que componen el proyecto, dando la impresión que han sido escritos por un solo desarrollador , con esto se busca que el mantenimiento de un producto de software sea totalmente entendible para el grupo de desarrollo que trabajara sobre este en futuro.

Referencias

[1] P. Roa, C. Morales, and P. Gutierrez, "Norma ISO / IEC 25000," vol. 3, no. 2, 2015.

[2] 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.

[3] J. C. Rodríguez, D. Pino, M. D. Roca, Z. H. Figueroa, and D. González Domínguez, "Hacia la Evaluación Continua Automática de Prácticas de Programación."

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

[5] 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.

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

[7] 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.

[8] C. Boogerd and L. Moonen, "Evaluating the relation between coding standard violations and faults within and across software versions," *Proc. 2009 6th IEEE Int. Work. Conf. Min. Softw. Repos. MSR 2009*, pp. 41–50, 2009.

[9] W. Basalaj Co- and F. Van Den Beuken, "CORRELATION BETWEEN CODING STANDARDS COMPLIANCE AND SOFTWARE QUALITY."