Sistema Web para la Evaluación de Métricas de Software

Trabajo Terminal No. ____-___

Alumnos: Cuevas Olvera Ian Axel, Ortega Estrada Ismael
Directores: Lorena Chavarría Báez, Nancy Ocotitla Rojas
e-mail: icuevas1801@alumno.ipn.mx, iortega1800@alumno.ipn.mx

Resumen – Las métricas de software son una medida cualitativa de atributos del software, están relacionadas con la calidad de este ya que cuantifican una serie de propiedades y atributos en los programas. Aunque las ventajas de utilizar métricas de software son significativas, a saber: selección de mejores alternativas, ahorro de recursos y mantenibilidad del proyecto, no se aplican con la frecuencia que deberían debido a distintos factores, entre los que destacan el tiempo y el esfuerzo que requieren y la falta de conocimiento sobre las mismas. Por lo tanto, la propuesta aquí planteada tiene como objetivo realizar un sistema capaz de examinar programas, y desplegar un análisis de sus características, utilizando como base algunas métricas de software para código fuente estructurado y orientado a objetos.

Palabras clave – Departamento de Ingeniería en Sistemas Computacionales, análisis, código, métricas, programación estructurada, programación orientada a objetos.

1. Introducción

La medición es una herramienta fundamental en cualquier tipo de ingeniería y la ingeniería de software no está exenta de ello. Para el ingeniero de software es indispensable poder evaluar el código que produce con el propósito de tomar mejores decisiones y aportar mayor valor al funcionamiento y desempeño del programa.

Las métricas de software se definen como una medida cualitativa de la forma en que un programa o sistema posee determinados atributos. Si se desea tener calidad en el software es esencial recurrir a ellas para que sirvan de guía en su evaluación, mejoramiento y clasificación [1] y así lograr dicho objetivo.

De manera general, la finalidad de las métricas de software es proporcionar un conjunto de indicadores que describan, detalladamente, cómo está escrito y cómo funciona un programa. Dentro de sus características, se puede decir que son [2]:

- Cuantificables, ya que están basadas en hechos, no en opiniones.
- Independientes, porque no pueden ser alteradas por las personas que las apliquen.
- Explicables, debido a que ellas mismas y su uso deben estar documentadas.
- Precisas, puesto que no se debe perder información.

La importancia de las métricas del software radica en que, a través de ellas, los desarrolladores pueden estimar los recursos necesarios en la realización de programas, el tiempo aproximado de entrega, el costo que tendrá el software, el mantenimiento de este y, lo más relevante, determinar si el programa es apto para una etapa de producción [2].

Aunque el desarrollo de programas se puede beneficiar de la medición, como se mencionó anteriormente, esta práctica ha perdido relevancia en los últimos años debido a que la mayoría de los desarrolladores, incluso desde que son estudiantes de ingeniería, no la toman en cuenta como una herramienta fundamental para el desarrollo de software. Un estudio interno realizado a 60 participantes de la industria del software en el año 2014 en Sri Lanka [5], país localizado al sur de Asia, reveló que las personas que desarrollan proyectos justifican que no hacen uso de las métricas de software debido a que es tedioso, no tienen los conocimientos necesarios o requieren mucho tiempo y esfuerzo. Según el estudio, el 62% de todos los participantes han realizado proyectos de software sin hacer uso de métricas; sin embargo, el 76% conoce las ventajas de aplicarlas durante el desarrollo de los mismos. Esto indica que, si bien un porcentaje alto de los participantes no han utilizado métricas de software en sus proyectos, un porcentaje aún mayor es consciente de las ventajas que implica su uso, por lo cual es valioso hacer énfasis en ponerlas en práctica.

En el mercado actual, existen pocas herramientas que realizan un análisis de métricas de programación estructurada y orientada a objetos, además, estos sistemas no son aptos para todos los usuarios debido a que sólo despliegan valores y no su significado, por lo cual sólo serán útiles para los usuarios que tengan conocimientos previos de las métricas y su interpretación. Por lo tanto, este es el problema que se abordará en este trabajo: la falta de herramientas computacionales para evaluar métricas de software para código fuente estructurado y orientado a objetos.

En la Tabla 1 se presentan distintas herramientas que fueron implementadas con el fin de evaluar métricas de software.

Tabla 1 Resumen de productos similares

SOFTWARE	CARACTERÍSTCAS	PRECIO EN EL MERCADO
SONARQUBE [6]	Herramienta de gestión de la calidad del código fuente. Permite recopilar, analizar y visualizar métricas del código fuente. Está formado por *Programming Mistake Detector* (PMD), *Check Style*, *Findbugs*, *Clover* y Cobertura. Se usa principalmente con Java, pero da soporte a otros lenguajes. Encuentra vulnerabilidades de seguridad de código, está actualizado y cuenta con soporte a diferentes lenguajes. No ofrece retroalimentación de los resultados, no cuenta con soporte para programación estructurada.	Software gratuito
GOOGLE CODEPRO ANALYTIX [7]	Es una herramienta de gestión de la calidad del software. Ofrece un entorno para evaluación de código, métricas, análisis de dependencias, cobertura de código, generación de pruebas unitarias, etc.	Ya no ofrece soporte (2010)
METRICS PLUGIN [8]	Es un *plugin* para el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés *Integrated Development Environment*) de Eclipse. Es necesario tener instalada la versión de Eclipse 3.1 (2004). El *plugin* analiza métricas de software orientado a objetos. Sin embargo, sólo ofrece los resultados de las métricas y no da una explicación de estos. La ventaja que tiene este *plugin* es que ofrece un diagrama de dependencias.	Ya no ofrece soporte (2004)
JARCHITECT [9]	Es una herramienta que brinda un análisis de métricas de software orientado a objetos para el lenguaje JAVA. Ofrece una interpretación de los resultados en forma de gráficos, pero es una herramienta de paga.	Software de paga
JAVA MEASUREMENT TOOL (JMT) [10]	JMT es una herramienta que analiza las clases Java y las relaciones entre ellas. Utiliza métricas de software orientadas a objetos.	Ya no está disponible (2002)
JAVA NON COMMENTING SOURCE STATEMENTS (JavaNCSS) [11]	Es una herramienta que permite realizar mediciones sobre el código fuente Java, obteniendo los valores de dichas mediciones agrupados a nivel global, clase y función. Utiliza métricas de software orientado a objetos. Su documentación aún se encuentra disponible, sin embargo, los enlaces para su descarga no funcionan.	Ya no está disponible (2019)
Solución propuesta	Sistema web para la evaluación de métricas de código fuente en programas que siguen el paradigma estructurado u orientado a objetos. Otorgará los resultados e interpretación de cada métrica. Se encontrará al alcance de cualquier usuario y, principalmente, cualquier estudiante de ingeniería en la rama de la computación debido a que será un sistema web gratuito.	Software gratuito

Como puede observarse, la gran mayoría de herramientas presentadas, si bien son gratuitas, no ofrecen una interpretación de las distintas métricas aplicadas. Debido a esto, el software propuesto proporcionará una descripción de la medición en los programas que se analicen con la finalidad de que se tenga un mayor

entendimiento de las características que describe cada una de ellas. Lo anterior tiene como propósito incentivar el uso de buenas prácticas en la comunidad de desarrolladores de software.

2. Objetivo

A continuación, se enuncia el objetivo general de este Trabajo Terminal.

Implementar un sistema de software para evaluar programas orientados a objetos y estructurados, escritos en los lenguajes de programación Java y C, respectivamente, usando métricas de software para código fuente.

Objetivos específicos

Una vez establecido el objetivo general, es necesario fragmentarlo en objetivos específicos para tener claro los puntos que se deben desarrollar para alcanzarlo. A continuación, se listan los objetivos específicos de este trabajo.

1. Implementar un subsistema gestor de usuarios.
2. Desarrollar un subsistema gestor de archivos.
3. Implementar un subsistema de análisis de métricas de programación orientada a objetos.
4. Implementar un subsistema de análisis de métricas de programación estructurada.
5. Desarrollar un subsistema de despliegue y comparación de resultados.

3. Justificación

El desarrollo de software a través de la aplicación de métricas es una práctica muy importante en el ámbito de la ingeniería, tan es así que existe una diferencia abismal entre un software que es desarrollado haciendo uso de dichas métricas y uno que no lo es. En primera instancia, un software que está desarrollado aplicando métricas permite tener un mejor panorama del programa que se pretende implementar. Su uso permite tener un estimado de los recursos que serán necesarios, debido a que, al tener parámetros de las características del código, se puede aproximar el costo y la eficiencia [12]. El uso de las métricas hace que el desarrollo de software se realice de manera más organizada ya que permite hacer un cálculo del tiempo en el que se planea culminar un proyecto, así como también, de la manera en que se reparten las tareas entre todos los miembros de un equipo de trabajo. Otro aspecto es la mantenibilidad, que hace referencia a la capacidad de que un programa pueda ser modificado, y que, gracias a las métricas, se ve beneficiado al tener un código limpio y ordenado. Las métricas de software deberían ser aplicadas a todos los proyectos, pues estas pueden ser la clave para la realización de un software que sea apto para salir al mercado y tener éxito. Sin embargo, en la actualidad, las métricas de software no son aplicadas a la gran mayoría de proyectos ya que existen falsas creencias alrededor de su uso, tales como asumir que puede resultar tedioso al representar una tarea extra cuando se desarrolla software, lo que trae consigo un mayor tiempo de desarrollo. Otra de las razones surge de la falta de conocimientos acerca de las buenas prácticas de programación.

Con este proyecto se pretende que los usuarios puedan tener una visión más clara de las características en sus programas escritos en lenguaje C o Java, mediante una evaluación de métricas estructuradas y orientadas a objetos. La intención es incentivar y recalcar las ventajas que tienen en el desarrollo de programas para que la comunidad de programadores, principalmente estudiantes de ingeniería, las tengan presentes cuando desarrollen proyectos, logrando así que el producto de software generado sea costeable, mantenible y, sobre todo, de calidad.

Para la realización de este Trabajo Terminal se pondrán en práctica los conocimientos adquiridos en las distintas unidades de aprendizaje cursadas en la carrera de Ingeniería en Sistemas Computacionales, entre las que destacan: Algoritmia y Programación Estructurada, Teoría Computacional, Análisis y Diseño Orientado a Objetos, Compiladores y Análisis de Algoritmos. Finalmente, se hará uso de los conocimientos de Ingeniería de Software para la organización del proyecto.

4. Productos o Resultados esperados

Este trabajo tiene como objetivo desarrollar un sistema Web que permita realizar el análisis de un programa, escrito en lenguaje C o Java, a partir de métricas de software para código fuente para programación estructurada y orientada a objetos. Para explicar cómo se pretende realizar el sistema, es necesario conocer qué entradas son requeridas para ser transformadas en las salidas esperadas. Esto se puede ver en el diagrama de la figura 1:

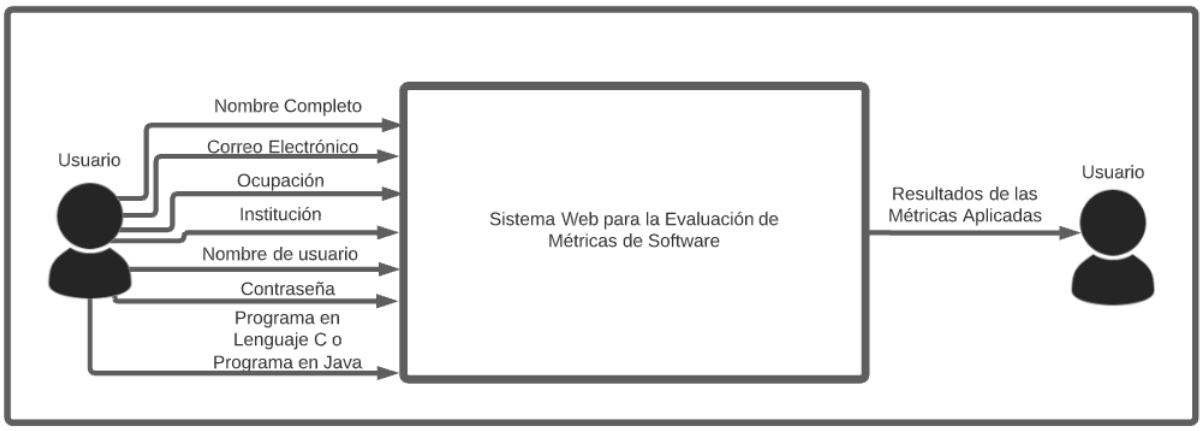

Figura 1 Diagrama de contexto

Como se observa en la Figura 1, la arquitectura tiene siete entradas, a saber: el nombre completo del usuario, correo electrónico, ocupación, institución donde trabaja o estudia, nombre de usuario, contraseña y el programa escrito en lenguaje C o Java. Las primeras seis entradas se usan para realizar el registro del usuario en el sistema. Una vez dado de alta, el usuario deberá proporcionar su nombre de usuario y contraseña para iniciar sesión. Después, el usuario ingresará al sistema el programa escrito en Lenguaje C o JAVA que desee evaluar con las métricas de software y, como salida, tendrá el despliegue de los resultados de las métricas aplicadas.

El sistema Web para la evaluación de métricas de software a su vez se compone de subsistemas, cada uno de los cuales tiene una tarea específica e interactúan entre ellos. La Figura 2 muestra la arquitectura del sistema y los subsistemas que se requieren para obtener los resultados de las métricas aplicadas y las conexiones entre los subsistemas.

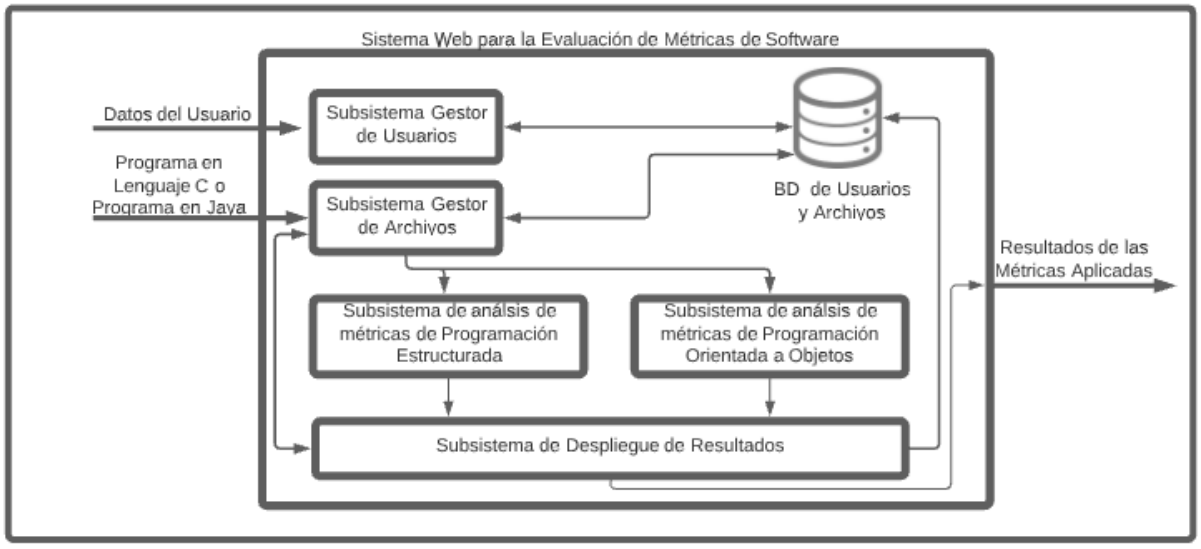

Figura 2 Arquitectura del Sistema

Como se mencionó anteriormente, el sistema consta de siete entradas. Para efectos de hacer más legible el diagrama, las primeras seis se agruparon en la categoría de Datos del Usuario. El Subsistema gestor de usuarios podrá insertar nuevos registros de usuario. Si el usuario ya está registrado, el subsistema verificará que los datos introducidos de nombre de usuario y contraseña coincidan con alguno de los registros en la base de datos, de no ser así se rechazará el acceso al sistema.

El Subsistema gestor de archivos será el encargado de identificar en qué lenguaje está escrito el programa proporcionado por el usuario y se lo pasará al compilador del lenguaje detectado para revisar que el programa no tenga errores de sintaxis. Se debe tomar en cuenta que el sistema sólo verifica que el programa no tenga errores sintácticos y no valida si funciona, es decir, no determina si hace lo que tiene que hacer. Si el programa

no tiene errores, entonces el subsistema guardará el programa en la base de datos y se lo pasará al subsistema de análisis de métricas correspondiente.

El Subsistema de análisis de métricas del lenguaje detectado evaluará cada una de las métricas del siguiente listado [13]:

Programación estructurada:

- Líneas de Código (LOC)
- Complejidad Ciclomática
- Métricas de Halstead

Programación orientada a objetos:

- De acoplamiento
  - Acoplamiento entre objetos (CBO)
  - Proporción de acoplamiento (CF)
- De cohesión
  - Falta de cohesión en los métodos (LCOM)
- De complejidad
  - Complejidad Ciclomática
  - Respuesta para una clase (RFC)
- De herencia
  - Índice de especialización por clase (SIX)
  - Profundidad del árbol de herencia (DIT)
  - Proporción de métodos heredados (MIF)
  - Proporción de atributos heredados (AIF)
- De encapsulamiento
  - Proporción de métodos ocultos (MHF)
  - Proporción de atributos ocultos (AHF)
- De reutilización
  - Número de hijos (NOC)
- De tamaño
  - Líneas de código (LOC)

Cuando este subsistema termine, los resultados del valor de cada métrica se pasarán al Subsistema de despliegue de resultados para que haga una interpretación de los valores, es decir, se explicarán los valores obtenidos en relación con algunos ya establecidos en la literatura y/o por un experto. De esta manera, si el usuario no tiene conocimiento de lo que significan los valores de cada métrica pueda entender mejor con la interpretación ofrecida. Al finalizar, los resultados se guardarán en la base de datos para su posterior visualización.

Una vez explicada la forma en que se realizará el sistema, se pueden listar los productos que se esperan obtener:

1. Sistema web para la evaluación de métricas de software.
2. Manual de usuario.
3. Manual técnico.

5. Metodología

Para desarrollar este Trabajo Terminal se hará uso de la metodología SCRUM, ya que es una metodología ágil para el desarrollo del software o para la gestión de proyectos. Se basa en aspectos como la flexibilidad para la adopción de cambios y nuevos requisitos, la comunicación constante entre los integrantes y, debido a su desarrollo iterativo, se pueden asegurar buenos resultados [15].

SCRUM es la mejor opción para el desarrollo del proyecto, ya que el sistema se puede dividir en subsistemas, cada uno de los cuales tendrá una cantidad determinada de *sprints*, para la etapa de desarrollo, implementación, revisión y pruebas. Con esto se espera una entrega en tiempo y forma del proyecto.

Como se puede ver en la Figura 3, dentro de SCRUM existen dos roles importantes, el *Product Owner* que, para este proyecto será Ian Axel Cuevas Olvera, cuyo trabajo es transformar y elegir las ideas propuestas del equipo para realizar el *Product Backlog* (Lista del Producto), y, el *Scrum Master*, del cual se encargará Ismael Ortega Estrada, quien es el responsable de que las técnicas de scrum sean aplicadas durante la organización del

proyecto y realizará el *Sprint Backlog* que contiene la lista de tareas en el *Product Backlog* que se realizarán en cada *sprint* [16].

Como se observa en la Figura 3, el ciclo de SCRUM se compone de 4 eventos principales, que son:

- **El *Sprint*:** En el cual se puede tomar un tiempo entre 2 a 4 semanas para desarrollar la lista de tareas en el *Sprint Backlog*.
- ***Daily Scrum* (Reunión Scrum diaria):** Se realiza una reunión de no más de 15 minutos para hablar de los avances del día anterior y sobre algún problema que se tenga.
- ***Sprint review* (Revisión del sprint):** Se lleva a cabo 1 vez por semana y no dura más de 1 hora para analizar las tareas terminadas del *sprint*.
- ***Sprint retrospective* (Retrospectiva del Sprint):** Se lleva a cabo cuando un *sprint* dura más de 4 semanas y la intención es ver que cambios se pueden realizar para mejorar la productividad y tiempo. Este bloque no puede durar más de 3 horas.

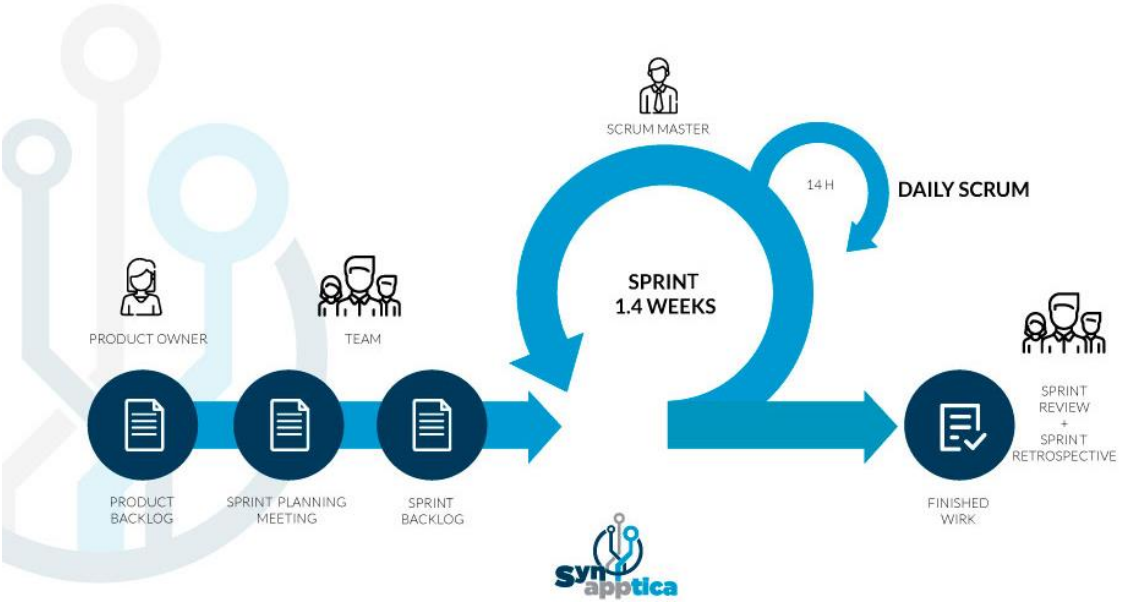

Figura 3 Diagrama de la metodología SCRUM [15]

SCRUM consta de cinco fases para el desarrollo de un proyecto. A continuación, se detallan cada una de ellas aplicadas a el sistema de evaluación de métricas de software [17]:

- **Inicio:** Se investigará y se analizará cada una de las métricas de software propuestas a profundidad.
- **Planificación y estimación:** Se asignará el número exacto de *sprints* para cada subsistema en la arquitectura del sistema, tomando en cuenta un aproximado de 3 a 4 *sprints* por subsistema.
- **Implementación**: Se desarrollarán los subsistemas uno a uno y se crearán los entregables.
- **Revisión y retrospectiva**: Se realizarán juntas para hacer una evaluación de los entregables.
- **Lanzamiento**: El *Product Manager* tendrá que aprobar los entregables y los enviará.

6. Cronograma

Nombre del alumno: Ortega Estrada Ismael

Tabla 2 Cronograma de actividades Ismael Ortega Estrada

Etapa	Actividad	FEB	MAR	ABR	MAY	JUN	AGO	SEP	OCT	NOV	DIC
	Documentación										
1	Investigación de Métricas de Programación Estructurada										
2	Análisis de los requirimientos de la Base de Datos										
3	Diseño y Creación de la Base de Datos										
4	Diseño y Creación del Sistema Gestor de Usuarios										
	Evaluación TT1										
5	Diseño y Creación del Sistema Gestor de Archivos										
6	Conexión de la Base de Datos										
7	Pruebas de la Base de Datos										
8	Reajuste y Pruebas										
9	Manual técnico										
	Evaluación TT2										

Nombre del alumno: Cuevas Olvera Ian Axel

Tabla 3 Cronograma de actividades Ian Axel Cuevas Olvera

Etapa	Actividad	FEB	MAR	ABR	MAY	JUN	AGO	SEP	OCT	NOV	DIC
	Documentación										
1	Investigación de Métricas de Programación Orientada a Objetos										
2	Análsis de los requirimientos de la interfaz										
3	Diseño y Creación del Módulo de Detección del Lenguaje de Programación										
4	Diseño y Creación del Sistema de Análisis de Métricas de Programación Orientada a Objetos										
	Evaluación TT1										
5	Diseño y Creación del Sistema de Análisis de Métricas de Programación Estructurada										
6	Diseño y Creación del Sistema de Despliegue y Comparación de Resultados										
7	Desarrollo de la Interfaz Web										
8	Reajuste y Pruebas										
9	Manual de usuario										
	Evaluación TT2										

7. Referencias

[1] S. Mirseidova and L. Atymtayeva, “Definition of software metrics for software project development by using fuzzy sets and logic,” in The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems, 2012, pp. 272–276.

[2] D. Ince, H. Sharp, and Mark, Introduction to software project management and quality assurance. London, England: McGraw-Hill Publishing, 1993.

[3] D. Rodríguez y Rachel Harrison, “MEDICIÓN EN LA ORIENTACIÓN A OBJETOS,” Uah.es. [Online]. Available: http://www.cc.uah.es/drg/b/RodHarRama00.pdf. [Accessed: 07-Oct-2021].

[4] P. Escudero, M. Moreno, and F. J. García, “MÉTRICAS ORIENTADAS A OBJETOS,”, Universidad de Salamanca (España), nov. 2001.

[5] W. K. S. Fernando, D. G. S. Wijayarathne, J. S. D. Fernando, M. P. L. Mendis, y I. Guruge, “The Importance of Software Metrics: Perspective of a software development projects in Sri Lanka”, SAITM Research Symposium on Engineering Advancements, pp. 91–95, abr. 2014.

[6] “SonarQube Documentation,” Sonarqube.org. [Online]. Available: https://docs.sonarqube.org/latest/. [Accessed: 07-Oct-2021].

[7] L. Gracia, “Google CodePro Analytix”, Unpocodejava.com, 20-sep-2010. [En línea]. Disponible en: https://unpocodejava.com/2010/09/20/google-codepro-analytix/. [Consultado: 17-oct-2021].

[8] “Metrics 1.3.6”, Sourceforge.net. [En línea]. Disponible en: http://metrics.sourceforge.net/. [Consultado: 17-oct-2021].

[9] CoderGears, https://www. codergears.com/home, “JArchitect :: Achieve Higher Java code quality,” Jarchitect.com. [Online]. Available: https://www.jarchitect.com/. [Accessed: 19-Oct-2021].

[10] C. Kolbe, “JMT - Java Measurement Tool”, Hu-berlin.de. [En línea]. Disponible en: https://www2.informatik.hu-berlin.de/swt/intkoop/jcse/tools/jmt.html. [Consultado: 17-oct-2021].

[11] “JavaNCSS - A Source Measurement Suite for Java”, Github.io. [En línea]. Disponible en: https://javancss.github.io/. [Consultado: 17-oct-2021].

[12] T. J. Mccabe, “A Complexity Measure,” IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. SE-2, no. 4, pp. 308–320, Dec. 1976.

[13] C. Blanco, “Ingeniería del Sofware II,” Unican.es. [Online]. Available: https://ocw.unican.es/pluginfile.php/1408/course/section/1803/tema1-pruebasSistemasSoftware.pdf. [Accessed: 07-Oct-2021].

[14] T. Honglei, S. Wei y Z. Yanan, "La investigación sobre métricas de software y métricas de complejidad del software", *Foro internacional de 2009 sobre tecnología y aplicaciones informáticas*, 2009, págs. 131-136, doi: 10.1109 / IFCSTA.2009.39

[15] “Primeros pasos scrum”, Synapptica.net. [En línea]. Disponible en: https://synapptica.net/metodologia-scrum.html. [Consultado: 17-oct-2021].

[16] M. Trigas, “Metodología SCRUM.” [Online]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

[17] A. Salazar, “Procesos de Scrum”, Prozessgroup.com, 16-oct-2016. [En línea]. Disponible en: http://www.prozessgroup.com/procesos-de-scrum/. [Consultado: 17-oct-2021].

8. Alumnos y Directores

Ian Axel Cuevas Olvera. – Alumno de la carrera de Ing. en Sistemas Computacionales en ESCOM, Especialidad Sistemas, Boleta: 2019630489. Tel. 5548837265, correo: icuevas1801@alumno.ipn.mx.

Firma: 

Ismael Ortega Estrada. – Alumno de la carrera de Ing. en Sistemas Computacionales en ESCOM, Especialidad Sistemas, Boleta: 2019630035. Tel. 5540692606, correo: iortega1800@alumno.ipn.mx.

Firma: 

Lorena Chavarría Báez. – Doctora en Ciencias en Ingeniería Eléctrica opción Computación por el Centro de Investigación y de Estudios Avanzados del IPN (CINVESTAV-IPN). Profesora de la Escuela Superior de Cómputo. Intereses de investigación: sistemas de información, bases de datos. Correo: lorena_chavarria@yahoo.com.mx.

Firma: 

Nancy Ocotitla Rojas. – C. Maestría en Ciencias de la Computación por el CIC. Ingeniería en Sistemas Computacionales por la ESCOM. Profesor de la ESCOM desde el 2004. Áreas de interés: Ingeniería de Software, Bases de Datos, Desarrollo de Aplicaciones para la Web. Email: nanwen1@gmail.com.

Firma: