Revista Digital de Tecnologías Informáticas y Sistemas
https://doi.org/10.61530/redtis.vol8.n1.2024

PARADIGMA DE LA PROGRAMACION ORIENTADA A OBJETOS (POO)

Lucio Guadalupe Quirino Rodríguez[1], Eduardo Alfonso Huerta Mora[2], Asia Cecilia Carrasco Valenzuela[3], Héctor Luis López López[4]

[1]Universidad Autónoma de Sinaloa, Facultad de Informática Mazatlán (MÉXICO)
[2]Universidad Autónoma de Sinaloa, Facultad de Ingeniería y Tecnología de Mazatlán (MÉXICO)
[3]Universidad Autónoma de Sinaloa, Preparatoria Rubén Jaramillo (MÉXICO)

Resumen

El estudio “paradigma de programación orientada a objetos”, tuvo como objetivo principal el detectar el nivel de aprendizaje de los alumnos que cursan la materia de paradigma orientada a objetos POO, se utilizó la metodología hibrida, cuantitativa-cualitativa para analizar los datos recabados de una muestra de 65 estudiantes de la carrera de ingeniería de software de la universidad autónoma de occidente, unidad regional sur. Se aplicó un cuestionario de 10 ítems de tipo linkert, para evaluar cada una de las características de la POO, abstracción, polimorfismo, encapsulación y herencia. También se entrevistó por medio de un cuestionar de preguntas abiertas a cada docente sobre la experiencia que tiene sobre los recursos didáctico utilizados para la instrucción de la materia. Los resultados que se obtuvieron indican que más del 50% de los estudiantes no entienden el paradigma de la POO, solo el 28% si lo entiende y lo aplica para solucionar problemas reales y el 22% entiende los conceptos, pero no los saben aplicar a cada situación problemática que se les presenta. Se recomiendan algunas estrategias didácticas de aprendizaje tales como aprendizaje basado en proyectos, descomposición de problemas, etc. Lo cual reducirá el número de reprobados en esta materia y entenderán el paradigma orientado a objetos.

Palabras clave: Aprendizaje, educación, enseñanza, objeto, paradigma, programación.

Abstract

The study “Object-Oriented Programming Paradigm” aims to detect the level of learning among students enrolled in the Object-Oriented Programming (OOP) course. A hybrid quantitative-qualitative methodology is used to analyze data collected from a sample of 65 software engineering students from the Autonomous University of the West, Southern Regional Unit. A questionnaire with 10 items was administered to evaluate each of the characteristics of OOP: abstraction, polymorphism, encapsulation, and inheritance. Additionally, each instructor was interviewed using an open-ended questionnaire about their experiences with the educational resources used to teach this subject. The results indicate that more than 50% of the students do not understand the OOP paradigm, only 28% understand and apply it to solve real-world problems, and 22% understand the concepts but do not know how to apply them to different problem situations they encounter. Several educational strategies are recommended, such as project-based learning, problem decomposition, etc., which will help reduce the number of students failing this course and improve their understanding of the object-oriented programming paradigm.

Keywords: Learning, education, teaching, object, paradigm, programming.

1 INTRODUCCIÓN

Durante la formación profesional de los estudiantes relacionados con los sistemas informáticos, es de suma importación el manejar diferentes estrategias de enseñanza para que lo aprendido sea de útil para la solución de problemas. No debe ser la excepción en la enseñanza de una metodología de programación. Es por ello, que se aborda este estudio en una de las materias cursadas por los estudiantes,” programación orientada a objetos”, es una asignatura que enseña al alumno el uso de un nuevo paradigma de programación de mucha utilidad para el desarrollo de aplicaciones de escritorio, en internet y para

dispositivos móviles. Es por ello es necesario que los docentes desarrollen habilidades, disposición, capacidades y competencias para un aprendizaje más significativo para los estudiantes.

Por ello, la autoridad encargada de las carreras de informáticas tenga especial cuidado en la planeación de actividades relacionadas con un aprendizaje de las nuevas tecnologías de hoy en día, esto atreves de cursos, talleres, actualización de contenidos, prácticas y actividades extra clase para fortalecer lo visto en clases.

Una de las teorías que abordaremos en este estudio es el constructivismo; esta teoría nos da conocer una nueva cultura educativa que se centra en cómo las personas construyen su conocimiento [1]. Además de incluir aspectos teóricos de la programación orientada a objetos, la cual es el objeto de estudio de la investigación llevada a cabo.

Jean Piaget es considerado como uno de los expositores más ideológicos de la psicología del siglo XX. Su tesis está dirigida no solo en el área de la psicología sino también por las ciencias pedagogía, etc. Básicamente trabajo dos vertientes: el descubrir y explicar las formas más elementales del pensamiento humano ser humano [2], Trabajando la psicología infantil y el desarrollo intelectual.

La teoría constructivista del aprendizaje tiene varias características clave que la distinguen de otros enfoques educativos. Aquí explicaremos los más importantes de la teoría.

El aprendizaje activo es un método de aprendizaje en que los estudiantes deben de asumir el rol de protagonistas en su etapa de aprendizaje, en este aprendizaje participan de manera reflexiva y activa, no como en la forma tradicional donde solo el maestro es el principal transmisor de conocimiento y solo el alumno escucha sin participar, a diferencia del enfoque activo los alumnos interactúan de manera directa y dinámica con los contenidos, aumentando y aplicando la critica a lo que aprendieron en situaciones prácticas. Este tipo de aprendizaje, sustenta que los alumnos construyen activamente sus conocimientos a través de sus experiencias diarias, escuela, hogares, empleos, etc., esto se logra por medio de la exploración, experimentación y resolviendo problemas.

El postulado del aprendizaje activo propone los enfoques de la participación activa de los estudiantes, aprendizaje basado en el estudiante, colaboración y trabajo en equipo, realizar prácticas del conocimiento realizar retroalimentación continua, evaluación formativa y reflexiva y conexión con experiencias previas mediante las características actuales de nuestro mundo, exige que la formación de los futuros profesionistas requiere un cambio [3].

El segundo factor o etapa de cómo es que es ser humano adquiere el conocimiento es la influencia del medio social. El conocimiento no se construye de manera aislada, sino que se encuentra influenciado por la interacción social, la cultura y el lenguaje Vygotsky establece que los aprendizajes se dan primero en situaciones sociales internalizarse en lo individua. La interacción con otras personas da al estudiante la oportunidad de observar, imitar y participar activamente en el aprendizaje lo que ayuda a conocer nuevos conceptos.

Es indiscutible que la reflexión ofrece al estudiante el analizar experiencias previas y relacionadas con la nueva información. Se puede conceptualizar la reflexión [4] “Consiste en pensar, meditar y evaluar sobre tus comportamientos, pensamientos, actitudes, motivaciones, etc. La reflexión es una estrategia que permite obtener información sobre el proceso de aprendizaje”. Los estudiantes, cuando reflexionan realizan un esfuerzo consciente por hacer conexiones de lo que ya aprendieron con los nuevos aprendizajes, esto facilita las construcciones de nuevos aprendizajes, lo cual corrige malos entendimientos reforzando ideas claves.

Una vez analizada la teoría del constructivista, abordaremos el paradigma de la programación orientada a objetos, definiendo que es un paradigma de programación: “Los paradigmas de programación son modelos para resolver problemas comunes con nuestro código, son caminos, guías, reglas, teorías y fundamentos que agilizan nuestro desarrollo y evitan que reinventemos la rueda” [5], resumiendo la definición anterior, podemos afirmar que un paradigma de programación por lo tanto es un método ya probado y validado para resolver un problema. Existen muchos paradigmas de programación entre los cuales podemos mencionar: Programación estructurada, programación por procedimientos, programación modular, paradigma de la programación orientada a objetos, etc.

https://doi.org/10.61530/redtis.vol8.n1.2024

Hablando de la parte técnica de la programación orientada a objetos (POO), se establece que esta metodología organiza el programa en objetos en lugar de módulos o subrutinas, definiendo un objeto como con una entidad del mundo real que cuenta con atributos y comportamientos [6]. En la POO se definen clases que es como una plantilla para crear objetos tratando de ejemplificar es aseveración es cuando al principio de crear un fraccionamiento se crea una casa modelo y en base a esta se crea todo el fraccionamiento.

Según los autores [7], [8], [9] afirman que una de las características principales de la POO es la de proporcionar una estructura clara, además de la facilidad de ofrecer la reutilización y encapsulación de código fuente lo que mejora el desempeño y fácil lectura de los programas. Es indiscutible que la programación, hoy en día es uno de los métodos más utilizados y adaptable a muchos lenguajes de programación de alto nivel. La POO ofrece muchas ventajas entre las cuales podemos mencionar: La reutilización y extensión del código, crea sistemas más complejos, existe una aproximación al mundo real, puede pude implementar programas visuales, se hace para practico el desarrollo de software, se trabaja en equipo y sobre todo facilita el mantenimiento del software.

La programación orientada a objeto se sustenta en cuatro pilares de la programación: Abstracción, encapsulamiento, herencia y polimorfismos [10]. La abstracción consiste en reducir las dificultades y complejidades del mundo real realizando un modelado solo de lo más importante de cada objeto, no tomando en cuenta detalles innecesarios permitiendo a los desarrolladores de programas dar importancia solo a la parte de mayor interés. La abstracción se dice es la mar importante de los pilares de la POO ya que de aquí parte un buen análisis y diseño orientado a objetos [5].

Encapsulación se refiere a la implementación de métodos dentro de una estructura de un programa, haciendo invisible con los métodos toda la información de mis atributos. El objetivo de la encapsulación es la de garantizar la integridad de mis datos por los medios de acceso a los atributos o métodos, los cuales pueden ser públicos, privados o protegidos. Con esta característica de la programación orientada a objeto, los usuarios de una clase desconocen la implementación de mi código.

La herencia en el paradigma orientada a objeto se refiere al mecanismo por el cual una clase o plantilla hereda los atributos y características de otra clase. Utilizando esta característica podemos decir que nuestros códigos de programación serán más rápidos y eficiente, además nos permite la reutilización de código, logrando así código mucho más limpio y entendible. Es importante mencionar que existe tipos de herencia manejables en los lenguajes de programación de alto nivel: Herencia simple, múltiple, jerárquica e hibrida. Con el manejo de herencia tenemos una gama de posibilidades y maneras de desarrollar eficientemente y estructurada nuestros programas [11].

De manera literal podemos definir el polimorfismo como la virtud de un objeto para adquirir múltiples formas o comportamientos. En términos de la programación se refiere a un mismo nombre a varios métodos, pero con distintos comportamientos. Cuando se utiliza esta característica en el paradigma de la programación orientada a objeto se tiene un sinfín de ventajas, por ejemplo: Permiten que los objetos se comporten de manera diferente, ayuda a ejemplificar la lógica del programa, expandir la funcionalidad del programa, etc.

2 METODOLOGÍA

Para la realización de esta investigación se utilizó una metodología hibrida o con un enfoque tipo mixto. En el enfoque mixto hace una división de enfoques tanto cualitativo como cuantitativo. El enfoque cuantitativo se refiere “a un conjunto de estrategia científicas que se usan en investigación para obtener información expresada en datos numéricos.”. Con esta información se pueden medir los resultados expresados en base a números. Este tipo de enfoque se puede emplear en el área de las ciencias exactas. Sin embargo, el enfoque cualitativo, se refiere a procedimientos que tienen su aplicación en las áreas de sociales, por ejemplo, en el área de psicología, administración, educación, etc. Estos dos enfoques se pueden combinarse para generar el modelo hibrido. Para llevar a cabo enfoque cuantitativo en la investigación, se utilizó un diseño no experimental descriptivo mediante encuesta a los estudiantes con 10 preguntas para cada característica de la programación orientada a objetos - abstracción, herencia, polimorfismo y encapsulación- el cuestionario está diseñado con preguntas de tipos likert donde el estudiante respondería el nivel de entendimiento y dominio (alto, medio y bajo), para cada una de las características de la POO para enfoque cualitativo, se entrevistó de manera verbal a los docentes sobre la importancia del paradigma

https://doi.org/10.61530/redtis.vol8.n1.2024

de la POO en otras materias de lenguaje de programación. Se realizó una categorización y codificación a partir de las respuestas recogidas por el cuestionario. El diseño de la investigación se clasifica como un estudio de casos, realizado en la Universidad Autónoma de Occidente en la carrera de Ingeniería de Software, FIMAZ.

La muestra fue conformada por 65 estudiantes (51 hombres y 14 mujeres) del tercer semestre que cursan la asignatura de programación orientada a objetos de los turnos matutino y vespertino de las carreras de ingeniería de sistemas de información y dos profesores que imparte la misma materia en ambos turnos de la Universidad Autónoma de Occidente unidad regional sur. El grupo de los 65 estudiantes fueron seleccionado por que es el único grupo que actualmente llevan la materia de programación orientada a objetos y de suma importancia conocer su opinión acerca del desarrollo académico de la materia. Por lo tanto, podemos decir que la técnica de muestro fue simple y directa y no probabilística. Para este estudio se utilizó la herramienta SPSS versión 20 para el procesamiento y análisis de la información.

3 RESULTADOS

Los primeros datos encontrados respecto a la claridad de conceptos en los pilares o característica de la programación orientada a objetos expuesto por los estudiantes que por el momento cursan esta materia, se pueden observar en la figura 1, lo cual tememos: El concepto de abstracción (explicar algo) el 21.5% de los estudiantes si tienen un claro entendimiento sobre este tema de este concepto, el 66.2% no lo entienden, sin embargo, el 12.3% si lo entiende, pero no sabe dónde utilizarlo o como aplicarlo en soluciones de problemas; En herencia se observa un 30.8% de los estudiantes si comprenden este concepto, un 40% no entienden el tema y 29.2% desconoce cómo y dónde aplicar esta característica; para la fase de polimorfismo el 21.5% si lo entiende, el 46.2% no entiende y 32.3% de manera parcial lo que significa que lo entiende pero no saben cómo aplicarlo; y para finalizar tenemos que la Encapsulación solo el 38.5% si lo entiende y lo sabe aplicar en situaciones que se les presente, el 46.2% realmente se le hace difícil de entender y para parcialmente tenemos un 15.4%.

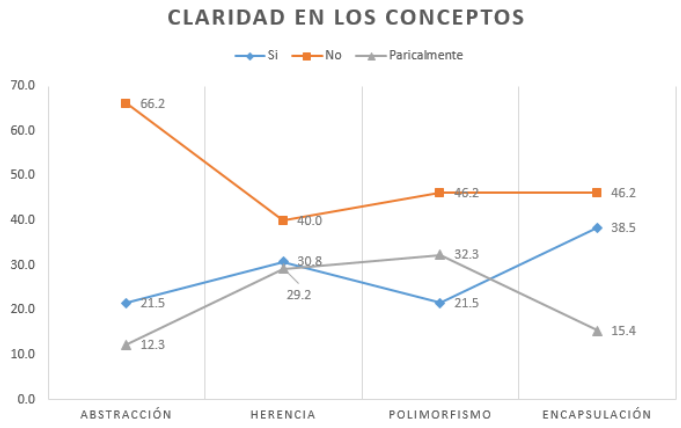

Figura 1. Claridad en los conceptos de POO.

Tabla 1. Claridad en conceptos.

Pilar	Si	No	Parcialmente	Total
Abstracción	14	43	8	65
Herencia	20	26	19	65
Polimorfismo	14	30	21	65
Encapsulación	25	30	10	65
Total	73	129	58	260

https://doi.org/10.61530/redtis.vol8.n1.2024

En la Figura 2, se puede ver claramente que el 50% de los alumnos encuestados no entiende realmente ningún concepto (Abstracción, polimorfismo, herencia y encapsulación) del paradigma orienta a objetos; solo el 28% de los alumnos si lo entiende y aplican estos conceptos para resolver cualquier problema; en la variable parcialmente identificamos solo el 22% de los estudiantes que entienden los conceptos de la POO, sin embargo, no saben aplicarlos a la solución de problemas.

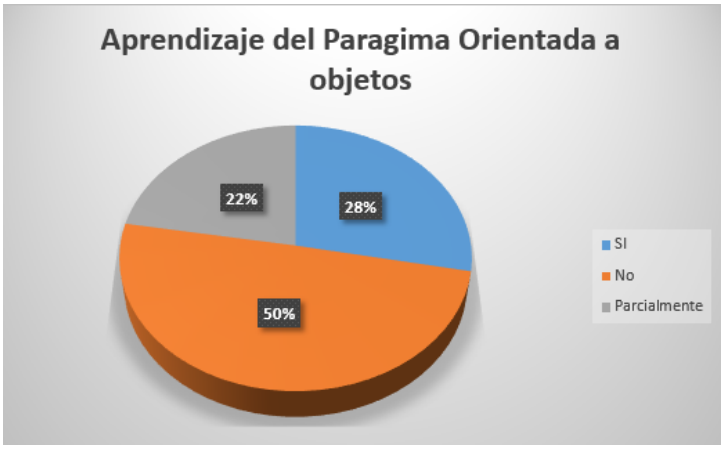

Figura 2. Aprendizaje de la POO.

En la Figura 3, se observa el nivel de aprendizaje de los estudiantes en relación con las características fundamentales o pilares de la programación orientada a objetos (POO), que son: abstracción, herencia, polimorfismo y encapsulación. Los datos muestran que un 33.3% de los estudiantes no comprenden el concepto de abstracción, lo que indica una dificultad significativa para asimilar esta característica esencial de la POO, que permite simplificar problemas complejos al centrarse en los elementos relevantes para el contexto.

Respecto a la herencia, el 32.8% de los estudiantes tienen una comprensión parcial del término. Aunque logran identificar la herencia como un mecanismo que permite que una clase adquiera propiedades y comportamientos de otra, no logran aplicar este concepto de manera efectiva en la resolución de problemas reales, lo que puede deberse a la falta de experiencia práctica o a una comprensión incompleta de su funcionalidad en contextos concretos.

El término de polimorfismo es entendido de manera parcial por el 36.2% de los estudiantes. Si bien algunos logran comprender la idea de que los objetos pueden tomar múltiples formas y que las clases derivadas pueden comportarse de diferentes maneras, aún presentan dificultades al aplicar este principio en escenarios más complejos, lo que limita su capacidad para aprovechar al máximo la flexibilidad que ofrece la POO.

Por último, un 34.2% de los estudiantes manifiestan una comprensión clara y son capaces de poner en práctica el concepto de encapsulación. Este porcentaje refleja una mayor familiaridad con este pilar, que se refiere a la capacidad de ocultar los detalles internos de un objeto y exponer únicamente las interfaces necesarias para interactuar con él, promoviendo así, la modularidad y la seguridad en el código.

Tabla 2. Claridad en conceptos

Pilar	Si	No	Parcialmente	Total
Abstracción	19.2	33.3	13.8	66.3
Herencia	27.4	20.2	32.8	80.3
Polimorfismo	19.2	23.3	36.2	78.6
Encapsulación	34.2	23.3	17.2	74.7

https://doi.org/10.61530/redtis.vol8.n1.2024

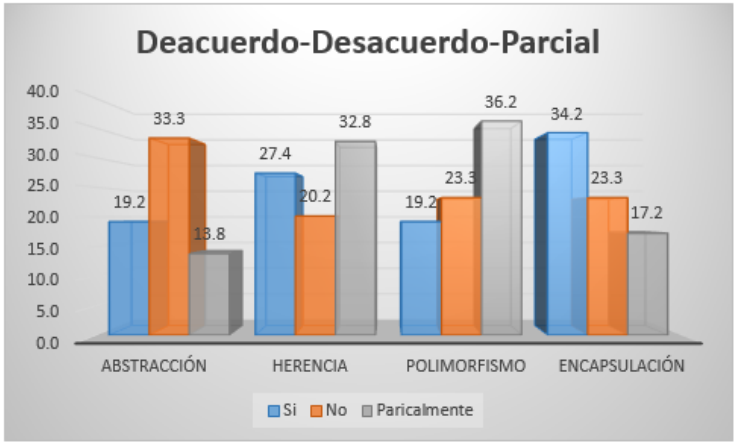

Figura 3. Aprendizaje de POO.

Una limitación o desventaja que pudiera presentarse en este estudio es que los resulados analizados puedan ser dificiles de generalizar debido a los multiples enfoques en el manejo y la implementación de la POO, metodologías de diseño, y los cambios en la evaluación. Además, de los constantes avances acelerados de las TI´C así como los nuevos paradigamas de programación que vuelven obsoletas las nuevas investigaciones.

4 CONCLUSIONES

El paradigma de programación orientada a objetos ha revolucionado la manera en que desarrollamos software, proporcionando un enfoque más intuitivo y alineado con la forma en que percibimos el mundo real. A través de conceptos clave como encapsulamiento, herencia y polimorfismo, OOP permite la creación de sistemas más modulares, reutilizables y mantenibles. Además, la OOP fomenta una mejor organización del código, facilitando la colaboración en equipos de desarrollo y la gestión de proyectos complejos. Sin embargo, también presenta desafíos, como la sobre complicación en el diseño y la curva de aprendizaje asociada a sus principios.

A medida que la tecnología avanza, la OOP sigue evolucionando, integrándose con otros paradigmas y adaptándose a nuevas necesidades. Es esencial que los desarrolladores comprendan tanto sus ventajas como sus limitaciones para aplicar este enfoque de manera efectiva y aprovechar al máximo su potencial en el desarrollo de software.

- Definitivamente el 50% de los estudiantes menciona no entender el paradigma de la programación orientada a objetos.
- Además, se puede concluir que, en la fase de abstracción del paradigma orientado a objetos, más del 66% de los estudiantes no la entiende o les es difícil, seguida por el polimorfismo y el encapsulamiento con un 46.2 y la herencia con un 40%.
- Cabe a aclarar que los profesores que imparten esta asignatura cuenta con el conocimiento y manejo del paradigma orientado a objetos.
- Al finalizar el semestre la gran mayoría de los estudiantes reprueban o no acreditan la materia recursan esta asignatura.
- Se recomienda que los docentes apoyen el curso de POO atreves de estrategia didácticas tales como aprendizaje basado en proyectos, por ejemplo, desarrollar una aplicación para gestionar una biblioteca digital. La aplicación debe permitir registrar libros, organizar diferentes tipos de materiales (libros, revistas, periódicos) y realizar operaciones de préstamo y devolución. Durante el proceso, deberán aplicar los principios de polimorfismo y encapsulación lo cual permite al educando trabajar en actividades concretas y aplicar lo entendido en situaciones reales.
- Utilizar la actividad que involucren el juego, gamificación, en el aprendizaje cotidiano.

- Utilizar la descomposición de problemas complejos en segmentos de fácil manejo desarrolla habilidades de resolución de problemas.
- Apoyarse en la utilización de herramientas como diagrama de flujo, pseudocódigos, esto ayuda al desarrollo de habilidades de resolución de problemas.
- La utilización de enseñanza investida, aquí permite al estudiante leer primero antes de ver el tema con el profesor y durante la clase resolver problemas de manera práctica.

El utilizar estas estrategias didácticas combinadas permitirá que la enseñanza del paradigma orientada a objetos sea más efectiva y fácil de entender cada una de las fases de esta asignatura y resolver cualquier tipo de problema que se les presente durante su formación profesión.

Para dar continuidad a este artículo se propone una segunda fase de estudio de investigación, para indagar sobre las metodologías de enseñanzas en el área de la informática. Se entiende que la informática es un área muy general pero lo que nos interesaría específicamente sobre lenguajes de programación, base de datos, lenguaje de programación, redes, matemáticas, ingeniería de software, etc.

REFERENCIAS

[1] C. V. Bonilla (2024). Didáctica en Entornos Virtuales de Aprendizaje: rumbo a la transformación digital de la formación docente. Revista Saberes Educativos, 2024, no 12, p. 9.

[2] S. L. Balón Romero. Estrategias psicopedagógicas para un niño de 10 años con dificultades en la lectoescritura. MS thesis. La Libertad, Universidad Estatal Península de Santa Elena, 2024, 2024.

[3] D. Verónica y S. E. Gutiérrez-Barreto, «El aprendizaje activo y el desarrollo de habilidades cognitivas en la formación de los profesionales de la salud», FEM: Revista de la Fundación Educación Médica, vol. 24, n.o 6, pp. 283-290, 2021.

[4] T. Parreño, «El constructivismo, según bases teóricas de César Coll», Revista andina de educación, vol. 2, n.o 1, pp. 25-28, 2018.

[5] A. Shirafuji et al., “Exploring the robustness of large language models for solving programming problems”, arXiv [cs.CL], 2023. https://doi.org/10.48550/arXiv.2306.14583 .

[6] O. Allen, X. Downs, E. Varoy, A. Luxton-Reilly, y N. Giacaman, “Block-based object-oriented programming”, IEEE Trans. Learn. Technol., vol. 15, núm. 4, pp. 439–453, 2022. https://doi.org/10.1109/TLT.2022.3190318 .

[7] H. Hourani, H. Wasmi, y T. Alrawashdeh, “A code complexity model of object oriented programming (OOP)”, en 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), 2019. https://doi.org/10.1109/JEEIT.2019.8717448

[8] J. Li, J. Chen, y A. Fong, “Una nueva arquitectura informática que respalde la programación orientada a objetos”, International Review on Computers and Software, vol. 8, pp. 174–179, 2013. https://doi.org/10.15866/IRECOS.V8I1.2762.

[9] L. Wen-Fang, Las características de la programación orientada a objetos en Java. 2005.

[10] P. Amahan y R. Sanqui, Sintaxis a sintaxis: evaluación de la ortogonalidad en el diseño de lenguajes -de programación orientados a objetos utilizando el método de listado de código. Actas de la 4.a Conferencia internacional sobre ingeniería de software y gestión de la. 2021. https://doi.org/10.1145/3451471.3451480.

[11] J. López, y F. Ortin (2013). Soporte eficiente de herencia dinámica para lenguajes basados en clases y prototipos. J. Syst. Softw. , 86, 278-301. https://doi.org/10.1016/j.jss.2012.08.016.