https://doi.org/10.37815/rte.v36n1.1163
Artículos originales

Estudio comparativo de los paradigmas de programación orientada a objetos y programación reactiva en la resolución de Integrales Algebraicas

Comparative study of object-oriented programming and reactive programming paradigms in solving Algebraic Integrals

Pedro Stalyn Aguilar Encarnación[1] https://orcid.org/0009-0005-1664-2280,
Jessenia Paola Castillo[1] https://orcid.org/0009-0009-5649-6862, Harold Jair Carreño[1]
https://orcid.org/0009-0004-9268-1524, Michael Estefania Jativa Brito[1] https://orcid.org/0000-0002-6394-2586

[1]*Escuela Superior Politécnica de Chimborazo*, Orellana, Ecuador
pedro.aguilar@espoch.edu.ec, jessenia.castillo@espoch.edu.ec,
harold.carrenio@espoch.edu.ec, usuario-4@correo.com

Enviado: 2024/04/19
Aceptado: 2024/06/25
Publicado: 2024/06/30

Resumen

Esta investigación evalúa el rendimiento en términos de tiempos de ejecución en la resolución de integrales algebraicas utilizando tanto el paradigma de la programación orientada a objetos (POO) como la programación reactiva (PR). El problema que se aborda es la escasez de evidencia científica que permita determinar qué paradigma ofrece mejores resultados tiempos de ejecución en la resolución de estas integrales. El enfoque del estudio implicó la implementación de dos versiones en Java, cada una construida siguiendo los principios de los paradigmas mencionados. Posteriormente, a través de escenarios experimentales controlados y métodos integrados de Java se midió el tiempo de ejecución de cada aplicación. Los resultados revelaron que la programación reactiva demuestra una mayor eficiencia en términos de rendimiento. Esta investigación se centró en la resolución de integrales algebraicas lineales y polinomiales y señala la necesidad de llevar a cabo investigaciones más exhaustivas en este campo. En conclusión, el estudio muestra que la programación reactiva supera significativamente a la programación orientada a objetos, demostrando tiempos de ejecución notablemente inferiores.

Palabras clave: Integrales algebraicas, Java, Programación orientada a objetos, Programación reactiva, Tiempo de ejecución.

Sumario: Introducción, Metodología, Resultados, Discusión y Conclusiones.

Como citar: Aguilar, P., Castillo, P., Carreño, H. & Jativa, M. (2024). Estudio comparativo de los paradigmas de programación orientada a objetos y programación reactiva en la resolución de Integrales Algebraicas. *Revista Tecnológica - Espol, 36(1), 58-67.*
https://rte.espol.edu.ec/index.php/tecnologica/article/view/1163

Abstract

The article evaluates the performance in terms of execution times in solving algebraic integrals using both the object-oriented programming (OOP) and reactive programming (RP) paradigms. The problem addressed is the lack of scientific evidence that allows us to determine which paradigm offers the best results in terms of execution times in solving these integrals. The approach of the study involves the implementation of two versions in Java, each built following the principles of the aforementioned paradigms. Subsequently, through controlled experimental scenarios and integrated Java methods, the execution time of each application is measured. The results reveal that reactive programming demonstrates greater efficiency in terms of performance. This research focuses on the resolution of linear and polynomial algebraic integrals and points out the need for more extensive research in this field. In conclusion, the study shows that reactive programming significantly outperforms object-oriented programming, demonstrating notably lower execution times.

Keywords: Algebraic integrals, Java, Object-oriented programming, Reactive programming, Runtime evaluation.

Introducción

Los paradigmas de programación juegan un papel crucial en la forma en que los desarrolladores abordan los problemas y diseñan soluciones. Dos de estos paradigmas, la programación reactiva y la programación orientada a objetos, destacan como enfoques distintos con sus propias filosofías y metodologías. La programación reactiva (PR) emerge como una solución dinámica y orientada a eventos, diseñada para abordar las complejidades de las aplicaciones basadas en eventos, mientras que la programación orientada a objetos (POO) se centra en la creación de entidades, que combinan estados y comportamientos relacionados entre sí. Comparar estos dos enfoques ofrece una visión esclarecedora de sus fortalezas, limitaciones y áreas de aplicación, permitiendo a los desarrolladores elegir la herramienta adecuada para cada tarea específica.

La programación orientada a objetos y programación reactiva, han realizado aportes significativos en el área de investigación de la Ingeniería de Software. Estos avances han permitido mejorar la eficiencia y flexibilidad de los recursos empleados para resolver integrales algebraicas, pero aún queda por analizar cómo el rendimiento de las aplicaciones que resuelven esta clase de problemas matemáticos puede variar significativamente en función del paradigma de programación mediante el cual fueron construidos (Galindo et al., 2023).

Ante este antecedente, esta investigación se enfoca en analizar los paradigmas expuestos en el ámbito de las integrales algebraicas, de esta forma, se determina aspectos clave que actúan como un papel crucial en la ingeniería de software y las matemáticas (Budnikova & Bulatov, 2012). En el marco de este estudio, se plantea el problema de investigación en dos preguntas:

1. ¿Qué paradigma de programación ofrece mejores resultados en términos de tiempo de ejecución?
2. ¿Existe una diferencia significativa en el tiempo de ejecución cuando se utilizan los paradigmas de POO y PR para resolver integrales algebraicas en Java?

En ese contexto, se implementó dos aplicaciones de *software* una de forma estructurada organizada y modular, bajo el paradigma orientado a objetos, y otra que contiene la capacidad de manejar eventos y cambios de manera eficiente, basada en programación reactiva (Mosteo, 2020), a fin de evaluar el rendimiento en función del tiempo de ejecución (Maina et al., 2022).

La metodología utilizada en el desarrollo de la investigación fue la Ciencia del Diseño y se empleó técnicas de recolección de datos al realizar pruebas en ambos paradigmas y métodos de análisis específicos como la media aritmética, lo que facilitó la evaluación para determinar el tiempo de ejecución de cada enfoque, permitiendo una comparación detallada de su rendimiento en función del tiempo (Xihui Zhang et al., 2020). Cabe señalar que se integró el uso de métodos propios de Java, para obtener el tiempo de ejecución de las aplicaciones. De esta manera, se espera que los resultados proporcionen evidencia clara sobre las diferencias en el rendimiento entre la programación orientada a objetos y la programación reactiva (Peñuela et al., 2021).

La estructura propuesta para el artículo incluye las siguientes secciones: Introducción: presentación del tema, relevancia y preguntas de investigación; Metodología: diseño del estudio y métodos utilizados para recopilar y analizar los datos: Resultados: presentación y análisis de los resultados; Discusión: comparación de los hallazgos obtenidos con los de otros investigadores; Conclusiones: resumen de los hallazgos clave, implicaciones prácticas y posibles direcciones futuras de investigación.

Metodología

En esta investigación, se implementó el mismo algoritmo para cada paradigma, permitiendo la evaluación comparativa de sus resultados en términos de tiempos de ejecución (Ortin et al., 2023); además, se adoptó la metodología de la Ciencia del Diseño, que abarca el ciclo de relevancia, diseño y rigor, fortaleciendo así la validez y la robustez del estudio.

Ciclo de relevancia

Esta etapa comprende la necesidad de analizar la resolución de integrales algebraicas bajo el paradigma de programación orientada a objetos y la programación reactiva. Para ello, se realizó una exhaustiva revisión de artículos especializados que abordan las temáticas mencionadas. Las variables identificadas para este estudio incluyeron; la variable independiente, representada por el software desarrollado, y la variable dependiente, que se centró en el rendimiento del programa en relación con el tiempo de ejecución. Además, se determinó las preguntas de investigación descritas en el apartado de la introducción, los requisitos de diseño y del artefacto de software a construir.

Ciclo de diseño

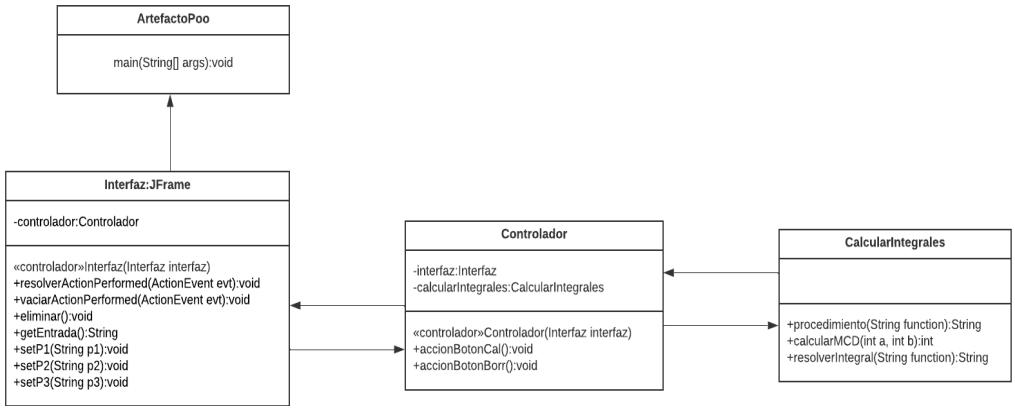
Esta etapa contempló la elaboración de las dos versiones del artefacto de *software.* Para ello, se establecieron los diseños mediante la utilización del lenguaje de modelado unificado, como una herramienta visual para modelar la estructura del sistema, proporcionando una representación clara de las clases.

Para la creación del artefacto de *software* bajo el paradigma de programación orientada a objetos, se inició con el diseño visual de la estructura Modelo-Vista-Controlador (MVC). En esta fase inicial, se detallaron las clases y responsabilidades asociadas a modelos, vistas y controladores, lo que facilitó una creación estructurada y eficiente del *software.* Respecto al diseño del artefacto bajo este enfoque, se elaboró una representación gráfica de la estructura y el comportamiento del *software*, como se muestra en la Figura 1.

La creación del artefacto en programación reactiva se hizo usando el patrón Observer, el cual desempeñó un papel fundamental al establecer una comunicación efectiva entre los diversos elementos del artefacto, permitiendo una sincronización fluida y actualizaciones oportunas.

Figura 1

Diagrama de Lenguaje de Modelado Unificado con enfoque Orientado a Objetos



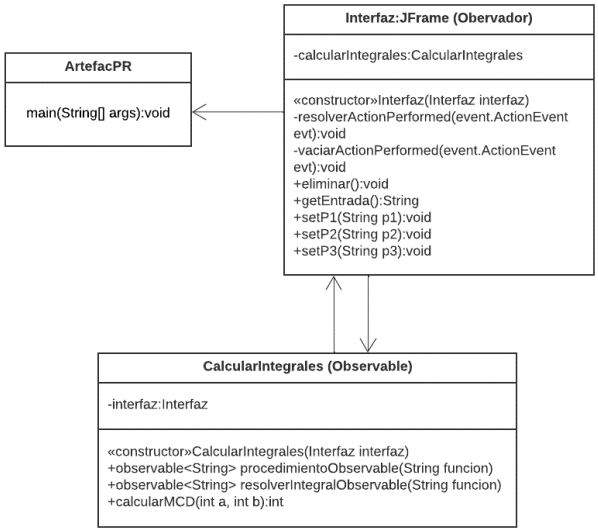
La implementación de estos artefactos empleó tecnologías específicas, optando por Java como lenguaje de programación y utilizando un entorno de desarrollo integrado como NetBeans. La implementación de las dos versiones de *software* se llevó a cabo en los dos paradigmas siguiendo el diseño propuesto.

Los escenarios experimentales se definieron de forma coherente para ambos artefactos, asegurando así la utilización de casos de integrales algebraicas comparables en los artefactos construidos. Esto incluyó la realización de tres pruebas de integrales lineales y otras tres pruebas en integrales polinomiales. Además, se llevaron a cabo pruebas exhaustivas para determinar el tiempo de ejecución. En este proceso, se empleó el método nanotime de Java, conocido por su precisión en la medición del tiempo, lo que garantizó una evaluación detallada y confiable del rendimiento de cada artefacto (Qiu et al., 2016).

La Figura 2 evidencia claramente la aplicación del patrón en mención en la estructura del artefacto. La implementación de estos artefactos empleó tecnologías específicas, optando por Java como lenguaje de programación y utilizando un entorno de desarrollo integrado como NetBeans. La puesta en marcha de las dos versiones de *software* se llevó a cabo en los dos paradigmas siguiendo el diseño propuesto.

Figura 2

Diagrama de Lenguaje de Modelado Unificado con enfoque Reactivo



Después de recopilar los datos, se utilizó la media aritmética como medida central, a fin de facilitar la comprensión del rendimiento promedio en cada enfoque. Estas pruebas, ejecutadas con una herramienta de medición específica, permitieron comparar y determinar qué paradigma mostró mejor rendimiento en la resolución de integrales algebraicas. Cabe mencionar que los tiempos de ejecución se determinaron en nanosegundos; sin embargo, a fin de ofrecer un mejor entendimiento de los resultados fueron convertidos a milisegundos.

Ciclo de rigor

Los datos efectivos recopilados en estas pruebas no solo arrojaron conclusiones sobre qué enfoque resultó más eficiente en la resolución de integrales algebraicas, sino que también contribuyeron al rigor científico y ético del proyecto al proporcionar evidencia empírica respaldada; sin embargo, es importante destacar que una de las limitaciones significativas del estudio fue su enfoque exclusivo en integrales algebraicas lineales y polinomiales, lo que restringió la generalización de los resultados a otros tipos de integrales algebraicas más complejas. A pesar de esta limitación, se establecieron prácticas y protocolos rigurosos desde la fase inicial de revisión de literatura hasta la presentación de resultados, con el fin de facilitar a otros investigadores la replicación exitosa del proyecto. Esta meticulosa atención a los detalles no solo promueve la transparencia y la reproducibilidad en la investigación, sino que también facilita la comparación y validación de los resultados por parte de otros investigadores interesados en el tema, fortaleciendo así la robustez y la confiabilidad del estudio en su conjunto. Se implementaron prácticas y protocolos rigurosos desde la fase inicial de revisión de literatura hasta la presentación de resultados, con el objetivo de facilitar la replicación exitosa de este proyecto en otros estudios, así como la evaluación y validación de los resultados por parte de otros investigadores interesados en el tema.

Background

Programación Orientada a Objetos

La programación orientada a objetos (OOP) constituye una metodología empleada para la modelación y el diseño de sistemas de *software*, en la cual se incorporan los principios fundamentales de encapsulación, abstracción, herencia y polimorfismo. Estos conceptos se rigen por un conjunto de principios establecidos conocidos como principios de diseño. En la programación orientada a objetos, el objetivo es desarrollar fragmentos de código reutilizables y de fácil mantenimiento que puedan llevar a cabo tareas complejas mediante la colaboración entre ellos; sin embargo, la estructura de un programa orientado a objetos difiere significativamente de la de un programa equivalente escrito con otro paradigma de programación. Aunque no existe una definición única y universalmente aceptada de programación orientada a objetos, hay conceptos fundamentales presentes en la mayoría de las definiciones de este paradigma (Zotos, 2007).

La programación orientada a objetos se aplica mediante la creación de clases que representan conceptos como integrales indefinidas, con atributos como la función a integrar y métodos para el cálculo; además, la encapsulación permite ocultar detalles internos, la herencia posibilita la creación de clases especializadas, y el polimorfismo facilita la interoperabilidad para resolver problemas relacionados con cálculos integrales (Singh et al., 2021).

Programación Reactiva

El paradigma de programación reactiva se ha propuesto como una solución idónea para desarrollar aplicaciones centradas en eventos. Aborda los desafíos inherentes a estas aplicaciones al ofrecer abstracciones que permiten expresar programas como reacciones a eventos externos; además, automatiza la gestión del flujo de tiempo y las dependencias de datos y cálculos, liberando a los programadores de la preocupación por el orden de los eventos y las

dependencias de cálculo. Este enfoque se basa en el paradigma de programación de flujo de datos síncrono, introduciendo conceptos como comportamientos para valores continuos y eventos para valores discretos. Además, permite la flexibilidad de la estructura del flujo de datos y admite flujos de datos de orden superior. Las investigaciones en torno a este paradigma de programación se han centrado en la facilitación de la construcción de gráficos y aplicaciones interactivas, y ha dado lugar al desarrollo de numerosas bibliotecas y extensiones para diversos lenguajes de programación; además de estos campos, la programación reactiva también encuentra aplicación en áreas como el modelado y la simulación, la visión por computadora y la iluminación escénica (Bainomugisha et al., 2013).

La programación reactiva implica la utilización de un paradigma de programación que se centra en la propagación automática de cambios. Este enfoque resulta beneficioso al tratar con múltiples variables y funciones interdependientes en el cálculo de integrales. Al adoptar la programación reactiva, se logra una gestión dinámica eficiente de las relaciones algebraicas, permitiendo que los cambios en una variable o función se propaguen automáticamente a través de las dependencias, actualizando de manera automática las demás variables y funciones afectadas. Este enfoque reactivo resulta particularmente útil para mejorar la eficiencia y la flexibilidad en situaciones donde las variables pueden cambiar en tiempo real (Wan & Hudak, 2000).

Integrales Algebraicas

Las integrales algebraicas, al abordar funciones polinomiales y lineales, se enfocan en calcular antiderivadas para expresiones matemáticas que combinan términos mediante operaciones algebraicas básicas (Jeffrey, 2004). La aplicación de técnicas específicas, como la regla de potencias y la regla de linealidad, facilitó significativamente la resolución de estas integrales. Este enfoque es esencial en la teoría de integrales, una rama fundamental del cálculo que proporciona herramientas matemáticas poderosas para comprender y analizar una amplia gama de fenómenos (Cai & Li, 2020); además, las integrales algebraicas tienen aplicaciones prácticas en diversas áreas, como la física, la ingeniería, la economía y la estadística, donde se utilizan para modelar y resolver problemas del mundo real; por otro lado, el estudio de las integrales algebraicas no solo proporciona soluciones numéricas, sino también una comprensión profunda de la relación entre la función original y su derivada, lo que resulta crucial en la comprensión de los procesos de cambio y acumulación en diferentes contextos.

Modelado de Lenguaje Unificado

El modelado Lenguaje de modelado Unificado (UML, por sus siglas en inglés) fue una notación estándar utilizada en el desarrollo de software para diseñar y documentar los artefactos de un sistema. Cuando se trabajaba con un programa en Java y se seguía el paradigma de programación orientada a objetos, el modelado UML era una herramienta valiosa para planificar, diseñar y comunicar la estructura y su comportamiento (Ozkaya & Erata, 2020).

Resultados

Los resultados de esta investigación se realizaron a través de dos fases fundamentales. En la primera etapa, se emprendió la creación de los artefactos de *software*. Posteriormente, en la segunda fase, se llevaron a cabo mediciones para obtener los tiempos de ejecución asociados al artefacto de software en ambos paradigmas.

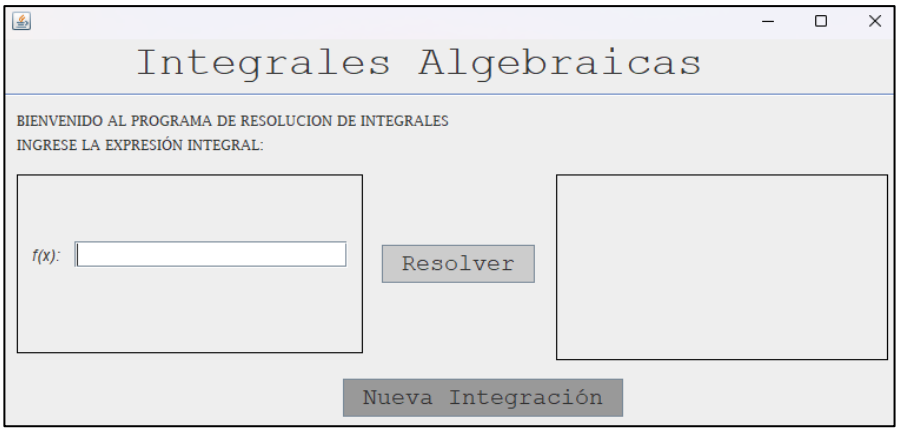
Artefactos de Software

Los resultados de los artefactos de *software* son el producto de un proceso riguroso de diseño, implementación y evaluación. Cabe mencionar que, a fin de no inferir en la obtención de los tiempos de ejecución y ofrecer una visión clara del rendimiento en cada enfoque de

programación, se determinó un interfaz en iguales circunstancias para los dos paradigmas propuestos. La Figura 3 muestra la interfaz utilizada en la construcción de los dos artefactos de *software* construidos.

Figura 3

Interfaz de la aplicación de software



Tiempos de Ejecución

El tiempo de ejecución abarca el lapso completo que un programa o proceso requiere para su finalización, desde el inicio de su ejecución hasta su conclusión, y engloba todas las tareas de inicialización y procesamiento. En este estudio, se realizó una exhaustiva recopilación de los tiempos de ejecución asociados a los artefactos de *software* en ambos paradigmas. Para ello, se emplearon los mismos conjuntos de ejercicios matemáticos en ambos casos, lo que proporcionó una base sólida para la comparación. Los datos recopilados se presentan de manera clara y concisa en la Tabla 1, lo que facilitó su interpretación y permitió un análisis detallado de los dos paradigmas utilizados; además de proporcionar información sobre el rendimiento de los artefactos, esta comparación ofrece una visión más amplia de las fortalezas y debilidades de cada paradigma en términos de tiempo de ejecución.

Tabla 1

Tiempo de Ejecución

NO.	INTEGRAL	TIEMPO EN NANOSEGUNDO PROGRAMACIÓN ORIENTADO A OBJETOS	TIEMPO EN NANOSEGUNDO PROGRAMACIÓN REACTIVA
1	$2x\text{^}3 + 2x - 2$	128400	28400
2	$3x - 2x\text{^}2$	218100	15700
3	$2x\text{^}4 + 7x\text{^}3 - 9$	232400	29700
4	$\frac{3x}{4} + 2$	196600	17800
5	$4x\text{^}5 - 2x\text{^}3 + 6$	207700	18400
6	$\frac{7}{5x} + 3$	159200	24400

El cálculo del promedio mediante la media aritmética se realizó para proporcionar una medida representativa de la POO y PR de los tiempos de ejecución(Tabla 2). Este proceso implicó sumar los tiempos de ejecución individuales de cada prueba para cada paradigma.

Tabla 2

Suma de tiempos de ejecución de acuerdo al número de prueba

	1	2	3	4	5	6	TIEMPO EN NANOSEGUNDOS
POO	128400	218100	232400	196600	207700	159200	**1142400**
PR	28400	15700	29700	17800	18400	24400	**134400**

Luego se dividió la suma total por el número de pruebas realizadas para obtener el promedio y dado que los tiempos de ejecución se registraron en nanosegundos, se realizó una conversión a milisegundos (Tabla 3).

Tabla 3

Promedio Total de tiempos de ejecución por paradigma de programación

PARADIGMA	CÁLCULO DEL PROMEDIO	TOTAL, EN NANOSEGUNDOS	TOTAL, EN MILISEGUNDOS
PROGRAMACIÓN ORIENTADA A OBJETOS	1142400/6	190400	0.1904
PROGRAMACIÓN REACTIVA	134400/6	22400	0.0224

Discusión

En el desarrollo del programa para resolver integrales algebraicas en Java, se compararon la Programación Orientada a Objetos y la Programación Reactiva en términos de tiempo de ejecución. Se encontró que la programación reactiva era más eficiente en tiempo de ejecución que la orientada a objetos, lo cual coincide con la creciente evidencia de su ventaja en la gestión eficiente de eventos y flujos de datos. El hecho de que la programación reactiva haya demostrado ser más eficiente en términos de tiempo de ejecución, una metodología más rigurosa no solo fortalecería la validez de los resultados presentados, sino que también facilitaría la replicación por parte de otros investigadores, fomentando así un avance más sólido en la comprensión de los enfoques de programación en la resolución de problemas matemáticos (Gomez-Gasquet & Diaz-Madronero, 2014).

La PR se ha aplicado en proyectos de construcción para ayudar en la toma de decisiones frente a retrasos, generando nuevos cronogramas y presupuestos asociados. Se han desarrollado métodos dentro de la estrategia de reprogramación reactiva, como la reprogramación total y parcial, que actualizan programas existentes para adaptarse a cambios durante la ejecución del proyecto (Garrido & Carrillo, 2013).

Conclusiones

En el desarrollo de este estudio, se lograron alcanzar los objetivos propuestos, que consistían en evaluar el tiempo de ejecución de dos enfoques para resolver integrales algebraicas mediante programación orientada a objetos y programación reactiva en Java. Los resultados indican que la implementación del paradigma de programación reactiva ofrece un

rendimiento superior, logrando resultados en un tiempo menor en comparación con el enfoque basado en POO.

Los resultados confirman que la programación reactiva es más eficiente que la orientada a objetos; sin embargo, se identifican limitaciones en el enfoque basado en POO, revelando un rendimiento menos eficiente en este contexto específico; también, la investigación exhaustiva de artículos relevantes y la implementación práctica del programa permiten analizar de manera concluyente los tiempos de ejecución, demostrando la superioridad de la programación reactiva en este escenario específico.

La interpretación de los hallazgos sugiere que, al utilizar la programación reactiva, se pudieron lograr mejoras significativas en el rendimiento del programa para la resolución de integrales algebraicas. La capacidad de ejecución inherentes a este enfoque proporciona una ventaja clara en comparación con la programación orientada a objetos en este contexto específico.

Los resultados contribuyen al destacar las diferencias de rendimiento entre estos enfoques en el ámbito específico de la resolución de integrales algebraicas y también aporta para tomar una decisión sobre qué paradigma utilizar para la resolución de integrales algebraicas. Al igual, la literatura respalda la noción de que la programación reactiva, al proporcionar un modelo de ejecución asincrónica y basada en eventos, puede conducir a una mayor capacidad de ejecución y tiempos de ejecución eficientes. En este caso, la evidencia respalda la idea de que adoptar la programación reactiva podía ser beneficiosa en cuanto a eficiencia temporal.

A pesar de los logros alcanzados en este estudio, es crucial reconocer y ser transparente acerca de las limitaciones inherentes al diseño y ejecución de la investigación. Una limitación potencial fue el enfoque en integrales algebraicas lineales y polinomiales, lo cual afectó la generalización de los resultados a otros tipos de integrales algebraicas más complejas. Futuras investigaciones podrían abordar esta limitación ampliando el alcance del estudio a una variedad más amplia de funciones; además, las implicaciones prácticas de este estudio sugieren que la programación reactiva en Java podría ser preferible para aplicaciones que requieren una resolución eficiente de integrales algebraicas. Se recomienda la implementación de este enfoque en entornos similares para mejorar el rendimiento de programas similares, al igual pueden ampliar la comprensión de diferentes enfoques de programación en diversas aplicaciones prácticas.

En conclusión, este estudio contribuye al entendimiento de cómo los paradigmas de programación impactan en el rendimiento de la resolución de integrales algebraicas en Java. Se comprueba que el paradigma de programación reactiva muestra una mayor eficiencia en términos de tiempo de ejecución en comparación con la orientada a objetos. Tras realizar pruebas resolviendo integrales algebraicas lineales y polinomiales, se observa una diferencia significativa en el tiempo de ejecución al utilizar los paradigmas de POO y PR, indicando que el enfoque reactivo es preferible para la resolución de este tipo de problemas en Java. Este trabajo aporta y no se limita solo a la optimización de la resolución de problemas matemáticos, sino que sugiere un impacto más amplio en el desarrollo de soluciones informáticas eficientes, abriendo la puerta a la aplicación de estos enfoques y nuevas perspectivas en problemas más amplios dentro de la programación.

Referencias

Bainomugisha, E., Carreton, A. L., Cutsem, T. van, Mostinckx, S., & Meuter, W. de. (2013). A survey on reactive programming. *ACM Comput. Surv.*, *45*(4). https://doi.org/10.1145/2501654.2501666

Budnikova, O. S., & Bulatov, M. V. (2012). Numerical solution of integral-algebraic equations for multistep methods. *Computational Mathematics and Mathematical Physics*, *52*(5), 691–701. https://doi.org/10.1134/S0965542512050041

Cai, M., & Li, C. (2020). Numerical Approaches to Fractional Integrals and Derivatives: A Review. *Mathematics*, *8*(1). https://doi.org/10.3390/math8010043

Galindo, C., Pérez, S., & Silva, J. (2023). Program slicing of Java programs. *Journal of Logical and Algebraic Methods in Programming*, *130*, 100826. https://doi.org/10.1016/j.jlamp.2022.100826

Garrido, A., & Carrillo, J. (2013). Programación reactiva en la administración de proyectos: aproximación conceptual y aplicaciones prácticas. *Revista EAN*, *74*, 72–85. https://doi.org/10.21158/01208160.n74.2013.737

Gomez-Gasquet, P., & Diaz-Madronero, M. (2014). Algorithms for reactive production scheduling: an application in the ceramic industry. *BOLETIN DE LA SOCIEDAD ESPANOLA DE CERAMICA Y VIDRIO*, *53*(4), I–IV. https://doi.org/10.3989/cyv.2014.v53.i4.1292

Jeffrey, A. (2004). 4 - Indefinite Integrals of Algebraic Functions. In A. Jeffrey (Ed.), *Handbook of Mathematical Formulas and Integrals (Third Edition)* (Third Edition, pp. 145–165). Academic Press. https://doi.org/10.1016/B978-012382256-7/50007-5

Maina, N. K., Muketha, G. M., & Wambugu, G. M. (2022). A Literature Survey of Complexity Metrics for Object-Oriented Programs. *International Journal of Science and Engineering Applications*. https://api.semanticscholar.org/CorpusID:248939582

Mosteo, A. R. (2020). Reactive programming in Ada 2012 with RxAda. *Journal of Systems Architecture*, *110*, 101784. https://doi.org/10.1016/j.sysarc.2020.101784

Ortin, F., Facundo, G., & Garcia, M. (2023). Analyzing syntactic constructs of Java programs with machine learning. *Expert Systems with Applications*, *215*, 119398. https://doi.org/10.1016/j.eswa.2022.119398

Ozkaya, M., & Erata, F. (2020). A survey on the practical use of UML for different software architecture viewpoints. *Information and Software Technology*, *121*, 106275. https://doi.org/10.1016/j.infsof.2020.106275

Peñuela, A., Hutton, C., & Pianosi, F. (2021). An open-source package with interactive Jupyter Notebooks to enhance the accessibility of reservoir operations simulation and optimisation. *Environmental Modelling & Software*, *145*, 105188. https://doi.org/10.1016/j.envsoft.2021.105188

Qiu, D., Li, B., & Leung, H. (2016). Understanding the API usage in Java. *Information and Software Technology*, *73*, 81–100. https://doi.org/10.1016/j.infsof.2016.01.011

Singh, N., Chouhan, S. S., & Verma, K. (2021). Object Oriented Programming: Concepts, Limitations and Application Trends. *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, 1–4. https://doi.org/10.1109/ISCON52037.2021.9702463

Wan, Z., & Hudak, P. (2000). Functional reactive programming from first principles. *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*, 242–252. https://doi.org/10.1145/349299.349331

Xihui Zhang, John D. Crabtree, Mark G. Terwilliger, & Redman, Tyler T. (2020). Assessing Students' Object-Oriented Programming Skills with Java: The "Department-Employee" Project. *Journal of Computer Information Systems*, *60*(3), 274–286. https://doi.org/10.1080/08874417.2018.1467243

Zotos, K. (2007). Object-oriented design principles in mathematics. *Applied Mathematics and Computation*, *188*(2), 1430–1436. https://doi.org/10.1016/j.amc.2006.11.009