Enseñanza de la programación orientada a objetos usando Alice 3D y el patrón MVC

Omaira Isabel Galindo Parra, *Estudiante de Maestría en Software Libre*
Paulo Cesar Ramírez Prada, *M.Sc. en Gestión, Aplicación y Desarrollo de Software*

Facultad de Ingeniería de Sistemas, Programa de Postgrados,
Universidad Autónoma de Bucaramanga, Colombia

ogalindop@unab.edu.co
pramirez206@unab.edu.co

***Resumen*—** El presente artículo describe el proceso realizado en la enseñanza de la programación orientada a objetos, aplicando el patrón de arquitectura Modelo Vista Controlador, a través del software educativo Alice en un ambiente en 3D, con un grupo de estudiantes de la asignatura Algoritmos y Programación de la Escuela de Ingeniería de Sistemas y Computación (EISC) de la Universidad Pedagógica y Tecnológica de Colombia (UPTC) en el primer semestre del año 2015, con el fin de determinar una estrategia alternativa de enseñanza.

Palabras clave: Enseñanza de la programación, patrón de arquitectura MVC, entorno de programación Alice, Programación Orientada a Objetos

***Abstract*—** This article describes the processes realized in teaching object-oriented programming, using the architecture pattern Model View Controller, through Alice educational software in a 3D environment with a group of students of the course Algorithms and Programming School of Computer and Systems Engineering (EISC) of the Pedagogical and Technological University of Colombia (UPTC) in order to determine an alternative teaching strategy.

Palabras clave: Teaching programming, MVC architecture pattern, programming environment Alice, Object Oriented Programming

I. INTRODUCCIÓN

En el ámbito de la educación en ciencias de la computación, y específicamente en el aprendizaje de la programación de computadores se ha identificado que los estudiantes encuentran dentro de este proceso dificultades de diferente índole, sobre todo en los cursos introductorios en donde se enseñan los conceptos básicos de programación y de forma adicional el paradigma de la programación orientada a objetos (POO), debido a que se les dificulta comprender y aplicar dichos conceptos de forma práctica en un lenguaje de programación; por esta razón investigadores y docentes relacionados con esta área de conocimiento han centrado sus investigaciones en la búsqueda de estrategias alternativas en el proceso de enseñanza aprendizaje de la programación.

Con el propósito de mejorar los procesos académicos, el programa de Ingeniería de Sistemas y Computación busca contribuir a la permanencia y retención de los estudiantes, a través de la implementación de estrategias de enseñanza que motiven a los alumnos en el proceso de aprendizaje de la programación de computadores y el paradigma de la Programación Orientada a Objetos, es así como se ha percibido que en otras universidades han optado por usar entornos de programación interactivos de código abierto, para la enseñanza de los conceptos básicos de programación obteniendo resultados positivos, razón por la cual se decide implementar el uso de un entorno interactivo en la enseñanza de un grupo de la asignatura introductoria a la programación y evaluar cuál es el efecto en el proceso de aprendizaje de los estudiantes.

Este artículo se enfoca a describir el proceso llevado a cabo para evaluar el uso de software libre en la enseñanza de la programación y la orientación a objetos, mediante el desarrollo de tres fases planteadas, que a nivel general contemplan la revisión de las herramientas interactivas usadas en la enseñanza de la programación y selección de una de ellas, la revisión de las estrategias usadas para enseñar programación con la herramienta seleccionada, la determinación de una estrategia a usar con un grupo de estudiantes de la UPTC, y finalmente el análisis de los resultados obtenidos.

II. CONTEXTUALIZACIÓN

En la enseñanza de la programación es importante que un estudiante de programas relacionados con las ciencias de la computación, se relacione con una base de conceptos que marcan el punto de partida en el proceso de aprendizaje, independiente del uso de un lenguaje de programación, dado que se manejan de una manera similar entre los diferentes

tipos de lenguajes, estos conceptos básicos se presentan a continuación:

A. Estructuras de programación

Independiente del lenguaje de programación, son consideradas como la base para el desarrollo de un programa, estas están caracterizadas porque definen el orden que siguen las sentencias durante la ejecución de un programa, se comportan de forma externa como una única sentencia, dando la posibilidad de concatenar unas estructuras dentro de otras, dando como resultado el flujo completo de ejecución del programa [1]; existen tres tipos de estructuras de programación, la secuencial, condicional o de selección y las de iteración o repetición.

1) *Estructura Secuencial*. Está conformada por N sentencias que se ejecutan en el orden ya establecido en la codificación del programa, se dice que es la estructura más simple, en el instante de conformar otras estructuras.

2) *Estructura de selección o condicional.* En el proceso de codificación no siempre se puede ejecutar un programa siguiendo los pasos de forma secuencial, que se definieron para resolver el problema, existe una alternativa que consiste en evaluar una sentencia booleana con valores posibles true/false, el programa evalúa la condición y si es verdadero el resultado ejecuta una o un bloque de sentencias, y opcionalmente si es falso ejecuta otras acciones.

3) *Estructuras de repetición.* Se definen como bloques de una serie de instrucciones que se repiten cierto número de veces o mientras que la expresión condicional controladora sea evaluada como verdadera, "dicha expresión se evalúa al comienzo de cada iteración del bucle, y de nuevo antes de cada iteración subsiguiente de la sentencia" [2].

B. Paradigma de la Programación Orientada a Objetos(POO).

Este paradigma de programación toma como concepto principal el objeto, el cual según Eck [1], se concibe como un tipo de módulo que contiene datos y subrutinas, por otra parte se considera como una especie de auto entidad suficiente, compuesta por un estado interno (los datos que contiene) y que puede responder a los mensajes (llamadas a sus subrutinas). Este paradigma ha sido el más utilizado por la mayoría de desarrolladores, porque permite la reutilización de código, mejora la calidad de un programa, gracias a que su desarrollo se realiza de forma modular y desacoplada lo que aporta facilidad en la etapa de mantenimiento.

Este paradigma está basado en la noción de que es importante la comunicación entre objetos, con el fin de simular fenómenos del mundo real; los objetos interactúan entre sí, a través del paso de mensajes, toman ciertos datos, los procesan y los pasan a otro objeto [3].

La programación orientada a objetos tiene un enfoque hacia la ingeniería del software, comenzando por la identificación de los objetos involucrados en un problema y los mensajes que estos objetos deben responder. El programa resultante es una colección de objetos, cada uno con sus propios datos y su propio conjunto de responsabilidades. La interacción entre los objetos se realiza mediante el envío de mensajes entre sí.

Es común que un objeto pertenezca a la misma familia o clasificación de otro objeto, lo que conlleva a decir que los objetos que poseen el mismo tipo de datos y responden a los mensajes de la misma forma, pertenecen a la misma clase, esta clase describe un grupo de objetos en particular que tienen idénticas características (datos) y comportamientos (funcionalidad), aplicando buenas prácticas de programación, para definir el nombre de una clase según Martin [4], se debe iniciar con letra mayúscula, si está compuesta por dos o más palabras, las primeras letras de las siguientes palabras deben iniciar con mayúscula, no se permiten espacios, y al igual que en el nombre de las variables, el nombre debe ser significativo.

C. Patrón de Arquitectura Modelo-Vista-Controlador (MVC).

La premisa mayor de este patrón se basa en la modularidad y la separación de tres aspectos: el modelo de datos, la representación visual de los datos en este caso la vista, y la interfaz entre la vista y el modelo (controlador); el principal objetivo de este patrón es separar los tres componentes de modo que sean tan independientes como sea posible, y que los cambios realizados en uno de ellos no afecten a los otros, de esta manera si se quisiera cambiar la interfaz gráfica de usuario, se realiza sin tener que cambiar el modelo de datos, ni el controlador. En la Figura 1, se muestra la interacción que plantea este patrón entre las tres capas. Una de las grandes ventajas del patrón MVC es la capacidad de reutilizar la lógica de la aplicación (que se implementa en el modelo) en la aplicación de una vista diferente, de esta forma el mantenimiento se realiza de una forma más fácil.

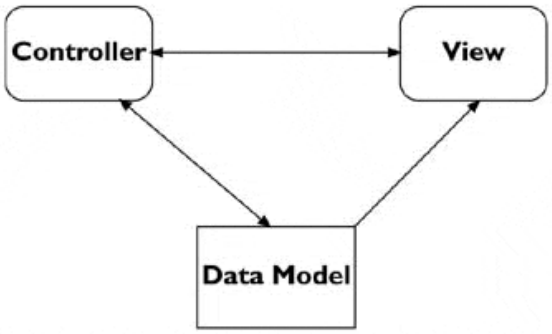

Fig 1. Patrón Modelo Vista Controlador. Fuente: [5]

Los componentes del patrón MVC mostrados en la Figura 1, se describen a continuación:

Modelo: en esta capa se encapsulan los datos específicos de una aplicación y se define la lógica y los cálculos que se realizan para manipular y procesar esos datos. Debido a que los objetos del modelo representan el conocimiento y la experiencia relacionada con un dominio de problema específico, pueden ser reutilizados en dominios de problemas similares [5].

Vista: el objetivo de esta capa es mostrar normalmente a través de la interfaz gráfica de usuario, los datos de los objetos del "modelo" de la aplicación y permitir la edición de los mismos.

III. DISEÑO METODOLÓGICO

El desarrollo del presente proyecto se realiza dentro de una investigación cuasi experimental, que según Campbell y Stanley [6] este tipo de investigación se usa en situaciones en las que no es posible asignar de forma aleatoria los sujetos a evaluar, y en este caso la muestra ya está preestablecida debido a que el grupo experimental es un grupo de estudiantes ya conformado, al igual que los otros grupos de la asignatura Algoritmos y Programación; los cuales se tienen en cuenta para la realización de un cuestionario final, los grupos no son equivalentes debido a que hay estudiantes que están cursando la asignatura por primera, segunda o tercera vez, entre muchos otros que no se pueden controlar.

El proyecto se desarrolló, a través de tres fases mostradas en la Fig 2.

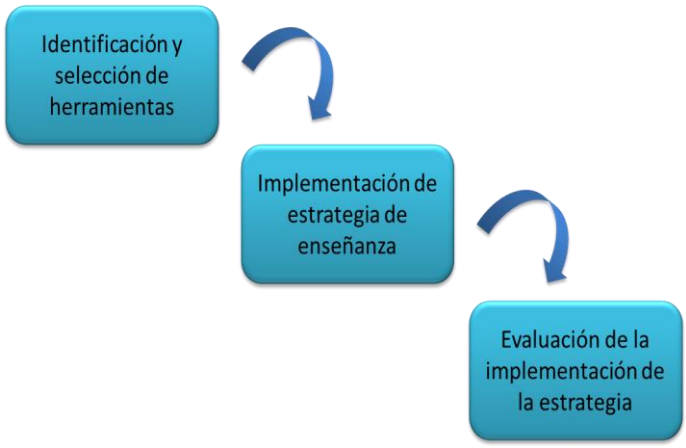

Fig. 2 Fases de desarrollo del proyecto. Fuente: Autores

1) Selección de la población. La Escuela de Ingeniería de Sistemas y Computación en conjunto con el grupo docente del área de programación del cual la autora de la presente investigación hace parte, manifestaron la necesidad de buscar alternativas de enseñanza de la programación que contribuyan en el proceso de adquisición de conocimientos por parte de los estudiantes, es así como en el primer semestre del año 2015, se tomó la decisión de asignar el grupo 2 de la asignatura Algoritmos y Programación para ser orientado por la autora del presente trabajo; esto con el fin de que evaluara el aporte del uso de un entorno de programación interactivo en el proceso de enseñanza de los contenidos programáticos de dicha asignatura.

Para el primer semestre del año 2015, se conformaron 5 grupos de la asignatura Algoritmos y Programación orientados por diferentes docentes, el grupo experimental se denominará AYP-EXP, y los grupos restantes AYP-G1, AYP-G3, AYP-G4, AYP-G5, cada grupo compuesto por 17 estudiantes.

2) Técnicas usadas para la recopilación de datos. Las técnicas que fueron usadas para la recopilación de datos fueron la encuesta y el cuestionario.

La encuesta se le realizó a 15 estudiantes del grupo experimental, y se orientó a evaluar el grado de aporte del entorno interactivo en el proceso de aprendizaje de los conceptos básicos de programación y la orientación a objetos, y por otra parte el grado de aporte de la implementación de la estrategia de enseñanza propuesta en la presente investigación y por último identificar en que aspectos se les presentó un mayor grado de dificultad en el proceso de aprendizaje.

En el caso del cuestionario se diseñaron 24 preguntas, de las cuales se planteó una de emparejamiento, tres de respuesta corta, una de respuesta numérica y 19 de selección múltiple. Las preguntas estuvieron orientadas a evaluar los conceptos de declaración de variables, manejo de tipos de datos, estructura condicional, estructuras de repetición, declaración de clases, métodos, objetos, declaración y manejo de vectores, manejo de errores. Este cuestionario se realizó, a través de la plataforma moodle desde el aula virtual de cada uno de los docentes, en los cinco grupos de la asignatura gracias a la colaboración de los docentes encargados de los otros grupos, quienes pusieron a disposición el tiempo de sus asignaturas para poder aplicar la prueba.

III. RESULTADOS

A. Selección de herramienta interactiva

Teniendo en cuenta la diversidad de entornos de programación, se consideró pertinente realizar la identificación de las características de cada una de estas herramientas, con el fin de seleccionar la herramienta que más se adaptara a los contenidos programáticos y los lineamientos de enseñanza establecidos en la EISC de la UPTC, se definió la metodología

como resultado de la adaptación del método de la suma ponderada y la guía de ponderación de indicadores, características y factores, de la UPTC [7], se realizaron los respectivos estudios y posterior selección. Los pasos generales de la metodología, se pueden observar en la Figura 3.

La metodología de evaluación de herramienta está establecida por cinco pasos como se observa en la Figura 3, partiendo de la definición de los niveles de importancia y el peso correspondiente a cada uno de ellos, como paso dos, se relacionan las características de cada herramienta que serán evaluadas para el desarrollo del estudio comparativo, definiendo su obligatoriedad respecto a los lineamientos de la EISC, en el tercer ya definidas las características a evaluar, se procede a calificar cada una de acuerdo al nivel de importancia y se calcula su valor de ponderación, luego en el cuarto paso se definen unos criterios por cada característica identificada, que serán valorados de acuerdo a su desempeño, y por último se procede a aplicar la evaluación de las características basadas en los criterios específicos de evaluación propios de cada una, obteniendo la tabla de resultados consolidados

A partir de los trabajos realizados por Utting[8] y Losada[9], en dónde se evalúan unas serie de entonos de programación interactivos se decide evaluar a Jeroo, Greenfoot, Scratch, Alice, StarLogoTNG.

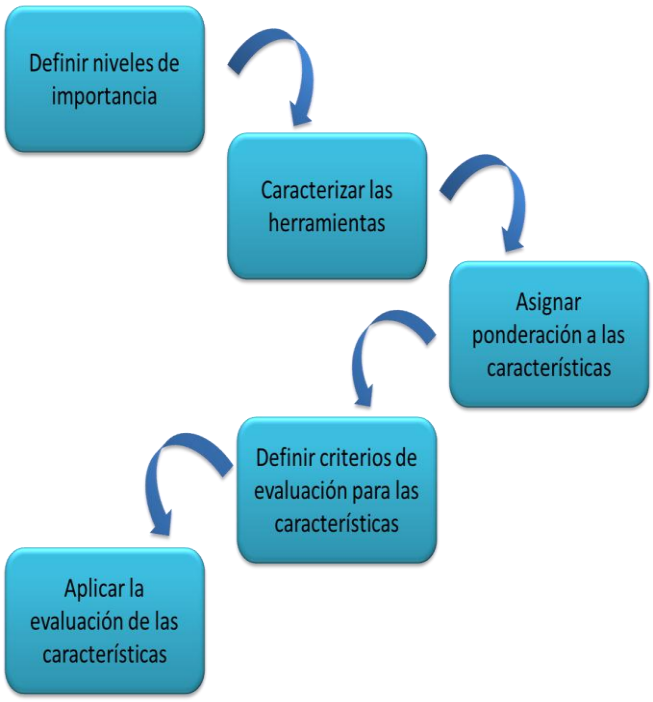

Fig 3. Pasos metodología evaluación de herramientas. Fuente Autores

Como resultado del paso 2, asignar ponderaciones a las características, se obtiene la Tabla 1.

TABLA I
PONDERACIÓN DE CADA CARACTERÍSTICA DE LAS HERRAMIENTAS

CARACTERÍSTICA	OBLIGATORIA	VALOR	PONDERACIÓN
Grupo de Edades	NO	20	3.7
Sistema operativo	NO	20	3.7
Visualización de código Java	SI	70	13.0
Manejo de estructuras de programación	SI	70	13.0
Manejo conceptos Programación Orientada a Objetos	SI	70	13.0
Manejo de tipos de datos	SI	70	13.0
Documentación	SI	70	13.0
Entorno 3D	NO	20	3.7
Entrada de texto Java	NO	20	3.7
Soporte	SI	70	13.0
Interfaz Drag and Drop	NO	20	3.7
Comunidad de Usuarios	NO	10	1.9
Ejecución inmediata	NO	10	1.9
Totales		**540**	**100**

Fuente. Autores

Al implementar la metodología y realizar la evaluación de las herramientas, se obtuvieron los resultados visualizados en la ñFigura 4.

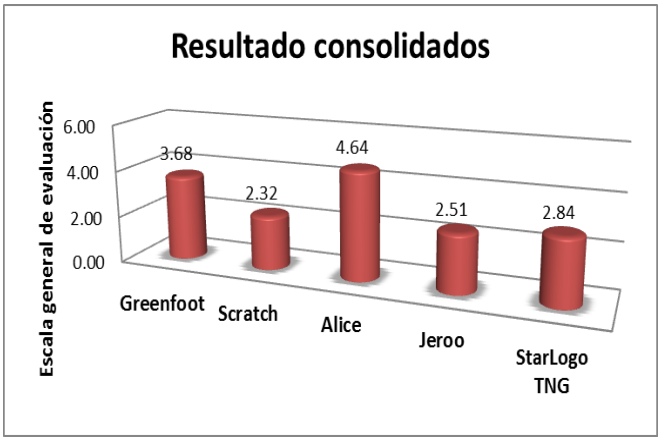

Fig. 4 Resultados consolidados evaluación de herramientas. Fuente: Autores

Los resultados de cada herramienta, corresponden a cada una de las barras de la gráfica; los resultados de este estudio, proveen de la información suficiente para definir las siguientes consideraciones:

- A partir de los criterios, a través de los cuales se realizó la evaluación de las herramientas, Scratch y Jeroo son las herramientas que obtuvieron los resultados más bajos, por tanto son descartadas para ser implementadas en la enseñanza de la programación y la orientación a objetos en el desarrollo del presente proyecto.

- De los resultados obtenidos es posible mencionar que StarLogo TNG obtiene una calificación más alta que las anteriores herramientas, en parte porque provee un entorno 3D y algunas funcionalidades de la programación orientada a objetos, pero no es el entorno que cumple con las características requeridas por la EISC, por consiguiente también es descartada.

- Greenfoot y Alice, son las herramientas que más representan las características solicitadas por la coordinación del área de programación de la EISC, pero existe una diferencia entre ellas, por tal motivo se plantea la **¡Error! No se encuentra el origen de la referencia.**5, correspondientes al análisis de resultados de estas dos herramientas, en que se expone en un gráfico de dispersión, los resultados tanto de Alice como Greenfoot, en cada una de las características determinadas para la evaluación.

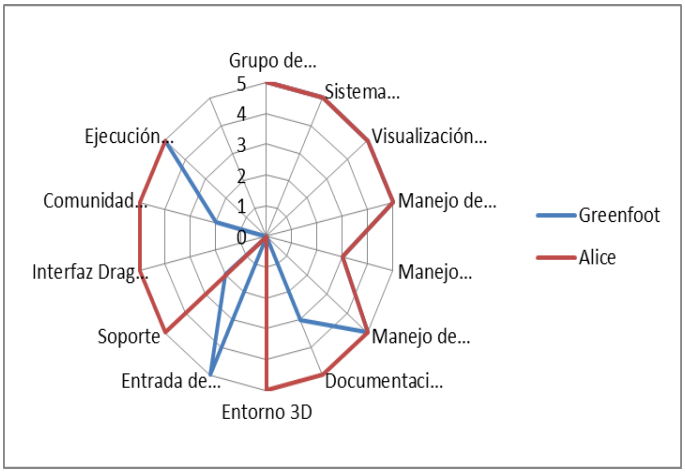

Figura 5. Resultados herramientas Greenfoot y Alice

Tomando como base la Figura 5, se puede inferir que el entorno de programación Alice es la herramienta que cumple en mayor medida con los criterios requeridos para la realización de la presente investigación, presentando un factor diferencial en las características de ofrecer un entorno en 3D, manejo de conceptos de la programación orientada a objetos, interfaz Drag and Drop, Comunidad de Usuarios, Documentación y Soporte, aportando un gran nivel de confianza en el uso de dicho entorno, para la realización de la presente investigación.

B. Implementación de una estrategia de enseñanza con Alice y el patrón de arquitectura MVC

Una vez identificado el entorno de programación Alice, se procede a determinar la estrategia de enseñanza que se implementaría en la EISC, para ello es importante tener en cuenta las estrategias que ya ha sido usadas, de la cuales se resalta la utilizada en la Universidad Autónoma de Bucaramanga, en la que a través de prácticas, y de forma paralela han creado programas en Java que evidencian los conceptos fundamentales de la POO, acordes a los ejercicios planteados en Alice.

Por otra parte, se describe la estrategia de enseñanza que exponen cuatro miembros del equipo de Alice, quienes han aplicado la teoría de educación de transferencia mediada, que consiste en desarrollar una comprensión intuitiva de ambos conceptos la orientación a objetos y los conceptos fundamentales de programación, y luego transferir el programa de Alice directamente a Java, mediante de un entorno interactivo de desarrollo (IDE) basado en texto, esto se realiza utilizando el mismo ejemplo en Alice y en Java, de esta forma se puede mediar en la transferencia del concepto [10]. Lo anterior lo sustentan en las teorías de educación citadas por Salomon y Perkings [11] quienes afirman que el aprendizaje

debe ser transferible, en otras palabras que lo se aprende en cierto contexto debería ser aplicable en otro contexto.

Con base a las estrategias expuestas anteriormente, y a experiencias vividas al interior del grupo de docentes del área de programación, se genera como resultado el desarrollo de una nueva estrategia en la que se tomara como base la estrategia propuesta por el equipo de desarrollo de Alice relacionado con el uso de analogías y la transferencia mediada, junto con la utilizada en la UNAB en la que diseñaron prácticas paralelas en Alice y en Java, adicionalmente se propone utilizar en enfoque basado en la resolución de problemas y la aplicación de buenas prácticas de programación en Java, y la adaptación del patrón de arquitectura Modelo Vista Controlador.

La propuesta de la estrategia se basa en los lineamientos de enseñanza de la EISC, que se orientan de forma general a la aplicación de buenas prácticas y el desarrollo de aplicaciones por capas, el paradigma de programación orientada a objetos en el lenguaje de programación Java.

Para implementar la estrategia con el grupo experimental, se decide realizar seis prácticas, basados en un formato de resolución de problemas, que a nivel general se realiza a través de los pasos mostrados en la Figura 6.

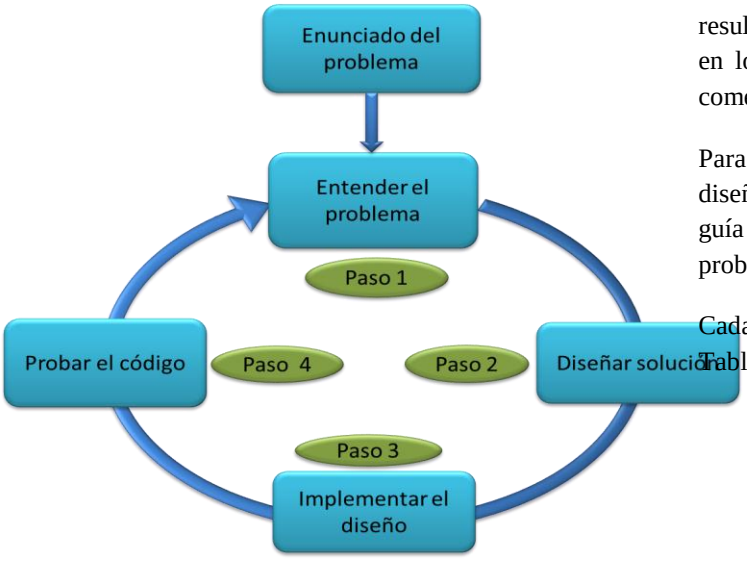

Fig. 6. Pasos generales para resolver problemas. Fuente Autores

El Paso 1, está enfocado a tomar el enunciado del problema, y realizar la identificación de los sustantivos que permitan abstraer los objetos que se requieren para plantear la solución al problema; adicionalmente se propone identificar las acciones asociadas a los verbos que incluye el enunciado, que más adelante darán como resultados los métodos correspondientes a los objetos identificados.

El Paso 2 está orientado a organizar las acciones identificadas en el paso 1, en un diagrama de actividades que establezca el orden de ejecución de las acciones. Por otra parte, se propone ubicar las acciones en el objeto que corresponda, teniendo en cuenta el patrón de arquitectura Modelo Vista Controlador, de modo que los objetos identificados se diseñen en una clase que será ubicada en la capa correspondiente al patrón de arquitectura, según la funcionalidad que cumpla dentro de la solución al problema, lo anterior debe ser diseñado en un diagrama de clases.

Luego de tener el diseño plasmado en el diagrama de actividades y de clases, se procede a realizar el Paso 3, tomando como base las acciones ya ubicadas en las clases respectivas del diagrama de clases, se toma cada acción y se descompone en pasos simples, que permitan cumplir dicha acción, posteriormente y teniendo en cuenta el diseño del paso 2 se procede a realizar el proceso de codificación del algoritmo diseñado en la herramienta correspondiente.

Como último paso, se comprueba el funcionamiento del código implementado, ejecutando el programa con diferentes casos de prueba, y registrando los resultados para luego contrastarlos con los resultados esperados, de no obtenerse los resultados que se esperaban se procede a revisar lo realizado en los pasos anteriores para identificar los errores de lógica cometidos.

Para implementar la estrategia de enseñanza planteada, se diseñaron seis prácticas, las cuales están conformadas por una guía realizada mediante el formato de resolución de problemas, con un ejercicio tanto en Alice como en Java.

Cada práctica se realizó usando la plantilla mostrada en la Tabla II.

TABLA II

PLANTILLA GENERAL DE LAS PRÁCTICAS

Nº DE LA PRÁCTICA:	**NOMBRE DE LA PRÁCTICA:**
OBJETIVOS:	
ENUNCIADO DEL EJERCICIO EN ALICE DESARROLLO MEDIANTE EL FORMATO DE RESOLUCIÓN DE PROBLEMAS	
ENUNCIADO DEL EJERCICIO EN JAVA DESARROLLO MEDIANTE EL FORMATO DE RESOLUCIÓN DE PROBLEMAS	
EVALUACIÓN	**EJERCICIO PARA REALIZAR EN ALICE**
	EJERCICIO PARA REALIZAR EN JAVA

Fuente: Autores

Como se pudo apreciar en el ejemplo de la práctica planteada, tanto los ejercicios de guía, como los de evaluación son ejercicios similares que llevan al estudiante a asociar los conceptos de programación, iniciando con el ejercicio en Alice, a partir del cual se centran en entender el concepto de forma gráfica, sin dejar de lado las etapas de análisis y diseño, que los ayuda a entender mejor el problema que van a resolver y plantear de una forma organizada la solución a dicho problema.

Las prácticas fueron diseñadas e implementadas con el grupo experimental, alrededor de 12 semanas, inicialmente realizando una inducción sobre el uso de Alice, y posteriormente iniciando con la primera práctica diseñada la cual no se llevó a cabo de forma completa, mediante el formato de resolución de problemas debido a que en esta práctica las acciones se desarrollan de forma secuencial en una sola clase dentro del método principal del entorno de programación, bien sea en Alice o en Java.

La aplicación de la estrategia de enseñanza a partir de las prácticas diseñadas se realizó, comenzando con la socialización del ejercicio de guía planteado en Alice, y luego con el ejercicio planteado en Java, donde inicialmente se les orientó sobre el proceso de edición del código fuente en un editor de texto, que les brindara ayuda en la indentación[1] o sangrado del código, y el resaltado de los colores en las palabras reservadas del lenguaje de programación Java. Posteriormente, se les enseñó a compilar y ejecutar el programa desde la consola de comandos, y así mismo a ir identificando los errores de sintaxis o compilación. Después de que se socializaban los ejercicios de guía, se les asignaban los ejercicios incluidos dentro de la práctica, en la última sección de evaluación.

C. Evaluación de la implementación de la estrategia

Al finalizar la implementación de las prácticas, se procede a aplicar una encuesta que busca a nivel general evaluar el grado de aporte de Alice en el proceso de aprendizaje de los conceptos básicos de programación y la orientación a objetos, y por otra parte el grado de aporte de la realización de las prácticas, el uso del formato de resolución de problemas, y la implementación del patrón de arquitectura MVC, en el proceso de resolución de problemas, para ello se procedió a diseñar una encuesta conformada por 15 preguntas, de la que se obtuvieron los siguientes resultados:

Con base a los resultados presentados en la Figura 7, se puede observar que el uso de Alice presentó un Alto grado de aporte en el aprendizaje de los conceptos de algoritmo, objeto, método, estructuras condicionales y estructuras de repetición, con unos porcentajes de 73.3, 53.3, 60 y 53.3 respectivamente, evidenciando que el uso de la herramienta es de gran ayuda en el proceso de aprendizaje de los estudiantes en el nivel introductorio de programación.

[1] Mover una sentencia hacia la derecha insertando espacios, con el fin de mejorar la legibilidad del código fuente.

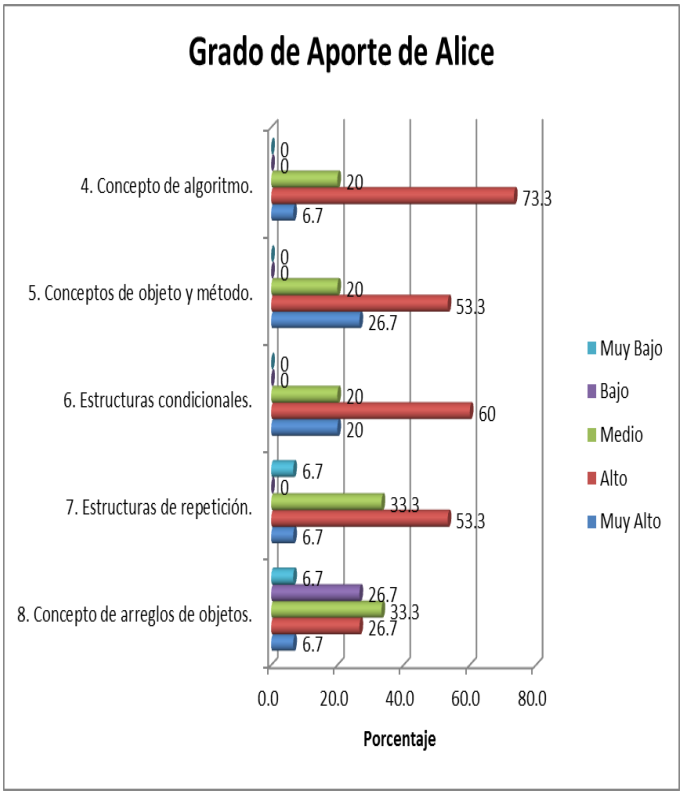

Fig.7. Resultados Grado de aporte de Alice. Fuente: Autores

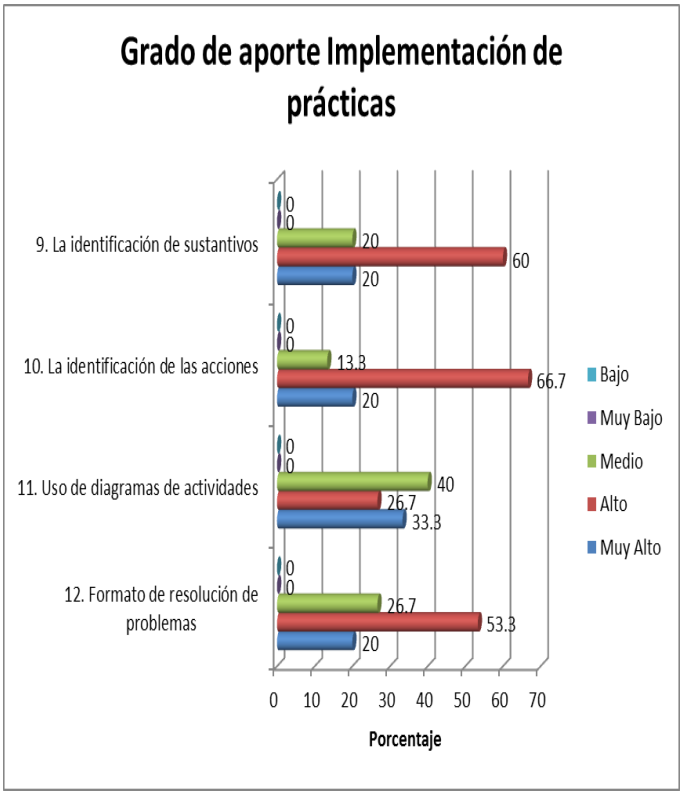

Figura 8. Resultados Grado de aporte de la implementación de las prácticas. Fuente: Autores

En la Figura 10, se observa que los estudiantes consideran, que la implementación de las prácticas, la identificación de los sustantivos y la identificación de las acciones le aportaron en alto grado (en más del 60%) en el proceso del planteamiento de una solución, en cuanto a si el formato de resolución de problemas les sirvió de guía para el desarrollo de un programa, los estudiantes consideran que les sirvió en algo grado, con un porcentaje del 53.3%. La opinión de los alumnos en cuanto al grado de aporte del uso de un diagrama de actividades en el diseño de un algoritmo se presenta en mayor porcentaje, en el grado de aporte medio (40%), seguido del grado de aporte muy alto con un porcentaje de 33.3%.

Es importante considerar que ningún estudiante manifestó que el grado de aporte de la implementación de la estrategia, haya sido bajo, ni muy bajo, lo que evidencia que dicha implementación tuvo un gran aporte en el proceso de aprendizaje de los estudiantes del grupo experimental como lo demuestran los resultados.

En la pregunta 13 se les solicito lo siguiente: *Seleccione en qué grado el uso de un diagrama de clases, le facilitó el proceso de escritura de un programa en el lenguaje de programación Java*, los resultados son los expuestos en la Figura 11, identificando que el 40% consideran que el grado de aporte fue alto y el 40% de muy alto grado, y el 20% restante considera que el grado de aporte medio, y 0% para bajo y muy bajo grado.

En esta pregunta se les solicitó adicionalmente que justificaran su respuesta, por lo que a continuación se muestra una recopilación de las apreciaciones de los estudiantes: *"ya que el diagrama de clases le permite saber que métodos hay que crear y que debe llevar", "muy alto porque con el diagrama de clases en una ayuda muy importante para luego la codificación del programa, además que si se hace el diseño del diagrama de clases uno va entendiendo mucho más el programa, y se ve de cierta manera más ordenado", "es bueno ya que ayuda a hacer la estructura y el orden del código más fácil de entender y el momento de programar ya se tiene claro que se va a hacer y cómo se va a hacer", "mediante el diagrama se establece un orden, el cual me permite idealizar el resultado del programa final", "podemos empezar a desarrollar el programa de una manera un poco más fácil ya*

que estamos siguiendo cierto orden establecido", "este diagrama me daba toda la estructura del programa solo quedaba llenar métodos y organizar el flujo del programa es un gran aporte", "porque el diagrama de clases nos da una gran ayuda ya que en este planteamos desde el principio las clases, métodos, atributos y demás cosas que vamos a necesitar al codificar el programa en Java".

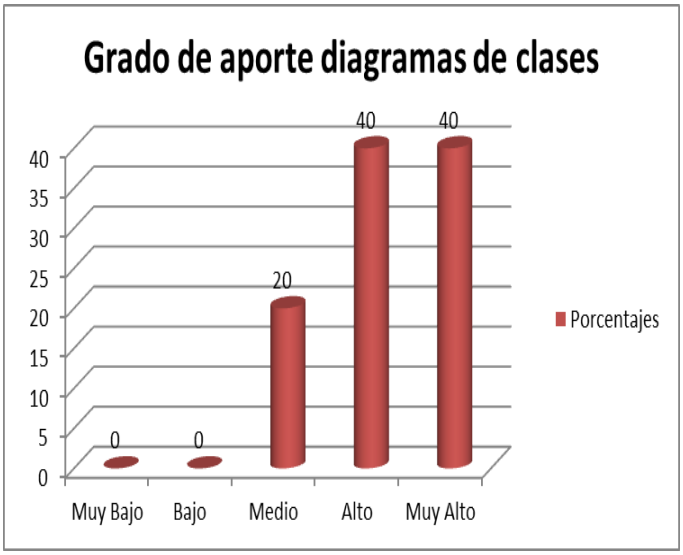

Fig. 9. Resultados pregunta 13, grado de aporte de los diagramas de clases

En la pregunta 14 se les solicito lo siguiente: *Seleccione el grado de satisfacción, que logró al organizar las clases en capas separadas, aplicando el patrón de arquitectura Modelo Vista Controlador*, los resultados son los expuestos en la Figura 12, identificando que el 40% consideran que el grado de aporte fue medio y el 40% de alto grado, y el 20% restante considera que el grado de aporte muy alto, y 0% para bajo y muy bajo grado.

Al igual que en la pregunta 13, se les solicitó que justificaran su respuesta, por lo que a continuación se muestra una recopilación de las apreciaciones de los estudiantes: *"pues debido a que utilizando este patrón de arquitectura nuestro programa se ve de cierta manera más ordenado y jerárquico, porque las acciones que debe realizar el programa se dividen en las distintas capas", "porque el primer semestre cuando trabaje algoritmos no lo trabajamos por capas, sino todo en una sola carpeta y clase", "es muy fácil de organizar el código y depurar los errores por el modo de organización", "este se me dificultó un poco al momento en el que aparecían muchas clases", "es muy alto porque así un programa queda mejor diseñado y más estructurado. También porque así se logró no tener tantos errores en los métodos y de sintaxis", "ya que las tres capas le permiten llevar un orden adecuado del programa o el algoritmo que se esté trabajando y además es más fácil poder identificar los errores".*

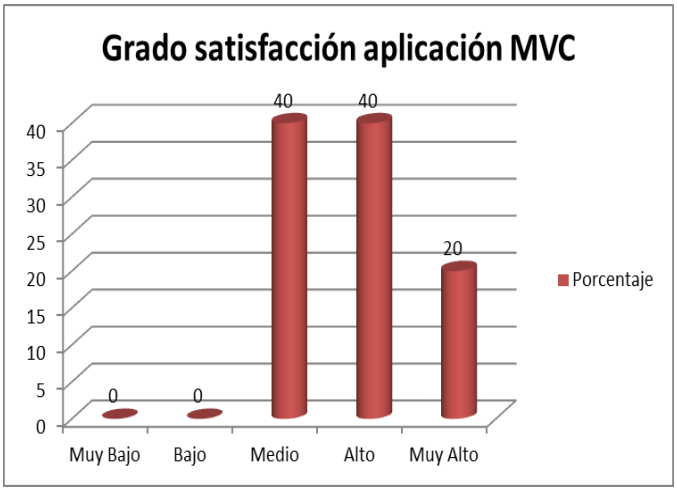

Figura 9. Resultados grado de satisfacción aplicando el patrón MVC. Fuente: Autor

Con el fin de comprobar la hipótesis planteada se diseñó un cuestionario conformado por 24 preguntas, de las cuales se planteó una de emparejamiento, tres de respuesta corta, una de respuesta numérica y 19 de selección múltiple. Las preguntas estuvieron orientadas a evaluar los conceptos de declaración de variables, manejo de tipos de datos, estructura condicional, estructuras de repetición, declaración de clases, métodos, objetos, declaración y manejo de vectores, manejo de errores. Este cuestionario se realizó a través de la plataforma *moodle* desde el aula virtual de cada uno de los docentes, en los cinco grupos de la asignatura gracias a la colaboración de los docentes encargados de los otros grupos, quienes pusieron a disposición el tiempo de sus asignaturas para poder aplicar la prueba.

Se decidió comprobar la hipótesis con un cuestionario unificado, para los cinco grupos de la asignatura incluyendo el grupo experimental, puesto que al finalizar el trabajo es importante a nivel general evaluar el nivel de aprendizaje de los contenidos programáticos de la asignatura, teniendo en cuenta que son los mismos para los diferentes grupos, y que a pesar de las estrategias y metodologías utilizadas por cada docente, estos contenidos debe ser abordados con los estudiantes. De dichos resultados se puede inferir que los promedios obtenidos por los estudiantes se encuentran en el rango de a 3 a 3.5, en la escala de 0 a 5, y que son similares en

todos los grupos, sin embargo se evidencia que los resultados más altos con un promedio de 3.5, de este cuestionario fueron obtenidos por los estudiantes del grupo experimental, evidenciando que se presentó una mejora en el proceso de conceptos fundamentales del paradigma de la POO.

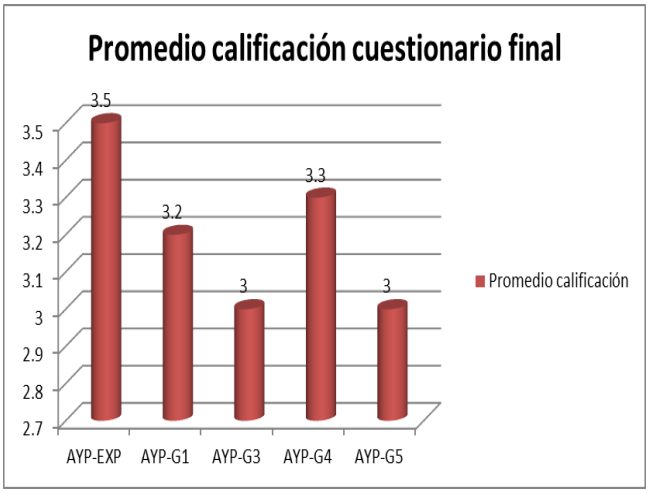

Figura 10. Consolidado resultados cuestionario final. Fuente: Autor

A partir de las notas finales de los estudiantes del grupo experimental, se generó la gráfica expuesta en la Figura 11, donde cada uno de los valores graficados corresponde al porcentaje de estudiantes que obtuvieron una nota clasificada en los rango de 0 a 1, 1 a 2, 2 a 3, 3 a 4 y 4 a 5, en una escala de calificación de 0 a 5. Se observa que el 29% de las notas se ubicaron en el rango de 3 a 4 y el 12% en el rango de 4 a 5, y que los estudiantes que cursaron la asignatura por primera vez hacen parte de esos porcentajes, y los estudiantes que cursaron la asignatura por segunda y tercera vez se ubican en el rango de 2 a 3 y 1 a 2.

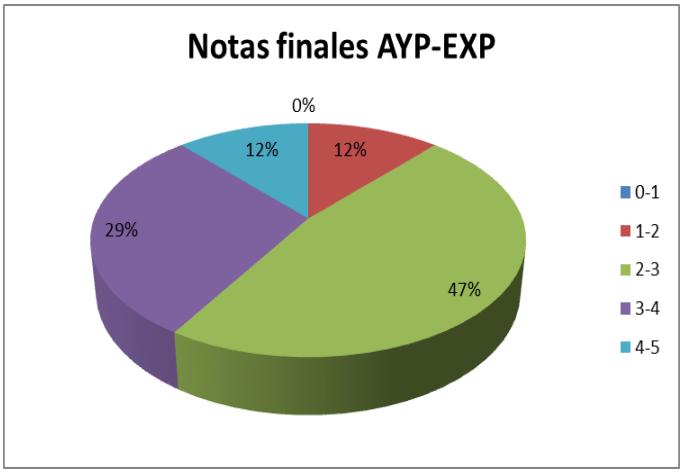

Fig. 11. Resultados notas finales grupo AYP-EXP

IV. Conclusiones

Tomando como base los resultados se puede afirmar que Alice aportó en un alto grado en el aprendizaje de los conceptos básicos de programación y la orientación a objetos, debido a que en el primer acercamiento del estudiante con algún concepto, lo realizó de forma visual y divertida y puede ver de forma inmediata los resultados de lo que va realizando parcialmente, motivando a los alumnos que no tienen experiencia en programación a desarrollar la lógica, al mismo tiempo que van aprendiendo los conceptos básicos de la programación y la POO.

La implementación de la estrategia de enseñanza planteada, permitió que el estudiante comprendiera inicialmente un concepto en específico mediante un ejercicio implementado en Alice, y luego viera el mismo concepto funcionando en un ejercicio similar en Java, de forma paralela, contribuyendo a que se cumpliera la transferencia de conceptos de Alice a Java.

Teniendo en cuenta que los estudiantes en algún momento de su carrera, tienen que codificar en un lenguaje de programación y enfrentarse a la sintaxis del lenguaje y el manejo de errores, se sugiere que el uso de Alice se realice de forma paralela a la codificación en el lenguaje, para que vayan asociando los conceptos, porque debido a una experiencia anterior se percibió que al enseñar los conceptos únicamente con Alice por un periodo determinado, y luego iniciar la enseñanza de los mismos conceptos en el lenguaje de programación, los estudiantes olvidaban lo aprendido con el entorno interactivo, y se tenía que empezar el proceso de ceros en el lenguaje.

Basados en los resultados obtenidos de la aplicación de la encuesta, se puede inferir que al inculcarle al estudiante la aplicación de una serie de pasos para la resolución de problemas, partiendo desde el análisis hasta la fase de pruebas, se le provee al estudiante un modelo que lo llevará a ser organizado en el desarrollo de los programas, independiente del lenguaje de programación o entorno que utilice.

Los estudiantes que estaban cursando la asignatura por primera vez, obtuvieron mejores resultados debido a que apenas están iniciando sus estudios de educación superior, y son más receptivos a los nuevos conocimientos que se les desea transmitir.

El uso del entorno de programación interactivo Alice, junto con la implementación de la estrategia del uso del formato de resolución de problemas, mejoró el nivel de aprendizaje de los conceptos básicos de la programación y la orientación a objetos, en un grupo experimental de estudiantes de la

asignatura Algoritmos y Programación de la de la Escuela de Ingeniería de Sistemas de la Universidad Pedagógica y Tecnológica de Colombia en el primer semestre del año 2015.

REFERENCIAS

[1] D. J. Eck, *Introduction to Programming Using Java*. San Francisco, USA: Hobart and William Smith Colleges, 2010.

[2] B. Eckel, *Thinking in Java*. Pearson Education, 2006.

[3] D. Jana, *C++ and Object-Oriented Programming Paradigm*. PHI Learning Pvt. Ltd., 2005.

[4] R. C. Martin, *Clean code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

[5] J. W. Cooper, *Java Design Patterns: A Tutorial*. Addison Wesley, 2000.

[6] D. T. Campbell and J. C. Stanley, *Experimental and quasi-experiment al designs for research*. Rand McNally, 1966.

[7] Universidad Pedagógica y Tecnológica de Colombia, "Procedimiento de Autoevaluación de Programas." 2011.

[8] I. Utting, S. Cooper, M. Kolling, J. Maloney, and M. Resnick, "Alice, Greenfoot, and Scratch – A Discussion," in *ACM Transactions on Computing Education*, New York, NY, USA, 2010, vol. 10.

[9] I. H. Losada, "Diseño de software educativo para la enseñanza de la programación orientada a objetos basado en la taxonomía de Bloom TESIS DOCTORAL 2012," Tesis Doctoral, Universidad Rey Juan Carlos, Madrid España, 2012.

[10] W. Dann, D. Cosgrove, D. Slater, and D. Culyba, "Mediated transfer: Alice 3 to Java," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, New York, NY, USA, 2012, pp. 141–146.

[11] G. Salomon and D. Perkins, "Teaching for transfer.," *Educ. Leadersh.*, pp. 22–32, 1988.