EVALUACIÓN DEL USO DE SOFTWARE LIBRE, EN LA ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORES.

OMAIRA ISABEL GALINDO PARRA

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA – UNAB

FACULTAD DE INGENIERÍA DE SISTEMAS

PROGRAMA DE MAESTRÍA EN SOFTWARE LIBRE CONVENIO UNAB-UOC

BUCARAMANGA

2015

EVALUACIÓN DEL USO DE SOFTWARE LIBRE, EN LA ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORES.

OMAIRA ISABEL GALINDO PARRA

Tesis presentada como requisito parcial para optar el título de:
Magister en Software Libre

Director:

M.Sc Paulo Cesar Ramírez Prada

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA – UNAB

FACULTAD DE INGENIERÍA DE SISTEMAS-POSGRADOS

PROGRAMA DE MAESTRÍA EN SOFTWARE LIBRE CONVENIO UNAB-UOC

BUCARAMANGA

2015

Nota de aceptación

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Bucaramanga para optar al título de Magíster en Software Libre.

M.Sc Paulo Cesar Ramírez Prada

_____

Director

Tunja, Junio de 2015

DEDICATORIA

A mi princesita Sofia, que con su llegada a mi vida me dio mucha más motivación para culminar esta etapa académica, a mi esposo Carlitos que con su paciencia me alentaba a terminar mi maestría.

A mis papitos, y hermanos y sobrinos, por el apoyo y amor que me han brindado durante toda la vida.

AGRADECIMIENTOS

Principalmente a Dios, por la oportunidad que me ofrece cada mañana de vivir un día más, y demostrarme que a pesar de las adversidades no pone pruebas en mi camino que yo no pueda cumplir de su mano, y a ti mamita María por ser mi dulce protectora.

A los Ingenieros Jairo Botero, Helver Valero por permitirme realizar pruebas con sus grupos de estudiantes, al ingeniero Omar Moreno por todo el apoyo brindado y los buenos consejos para orientar mi investigación, a los Ingenieros Germán Amézquita y Alexander Castro, por sus buenas ideas y revisiones del documento final. A Yeyita por tener las palabras adecuadas en esos momentos difíciles y animarme a seguir.

A todos y cada uno de los compañeros de la UPTC, que estuvieron pendientes del desarrollo del presente proyecto, y que me dieron ánimo en momentos difíciles, y a los estudiantes que hicieron parte de mi grupo experimental, y que participaron activamente de este proceso alternativo de enseñanza-aprendizaje

A la Escuela de Ingeniería de Sistemas y Computación de la UPTC, por brindarme los espacios para llevar a cabo esta investigación.

A la Ingeniera Olga Lucia Monroy por brindarme la idea inicial para la formulación del presente proyecto.

Al Ingeniero Paulo Cesar Ramírez Prada, quien asumió la dirección de esta tesis con mucho compromiso, y me brindó sus conocimientos y colaboración en todo momento.

Al personal docente y administrativo de la UNAB y de la UOC, que me brindaron los medios para cursar esta maestría y continuar con mi formación académica.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS DIGITALES

Anexo A. Encuesta

Anexo B. Cuestionario final Algoritmos y Programación

Anexo C. Cronograma de Actividades

Anexo D. Presupuesto y recursos necesarios

Anexo E. Formato de resolución de problemas

Anexo F. Practica 1. Estructura secuencial y declaración de variables

Anexo G. Practica 2. Declaración y uso de métodos

Anexo H. Practica 4. Estructura de repetición for

Anexo I. Practica 5. Estructura de repetición while

Anexo J. Practica 6. Arrays unidimensionales de objetos

Anexo K. Modelo dinámico de enseñanza de la POO.

GLOSARIO

AYP-EXP: Grupo experimental asignatura Algoritmos y Programación
EISC: Escuela de Ingeniería de Sistemas y Computación
UPTC: Universidad Pedagógica y Tecnológica de Colombia
UNAB: Universidad Autónoma de Bucaramanga
POO: Programación Orientada a Objetos.
MVC: Patrón de arquitectura Modelo Vista Controlador

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; 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.

La Escuela de Ingeniería de Sistemas y Computación de la UPTC, no es ajena a esta problemática general, por lo que surge la idea de realizar la presente investigación, planteando como objetivos principal “Evaluar el uso del Software Libre en la enseñanza de los conceptos básicos de la Programación de Computadores, para identificar una estrategia alternativa en el proceso enseñanza-aprendizaje.”

El desarrollo de esta investigación contribuye al proceso de mejoramiento de los procesos al implementar una estrategia de enseñanza alternativa que motive a los estudiantes en el proceso de aprendizaje de la programación.

En los capítulos 1, 2 y 3 se tratan los aspectos formales del proyecto, entre los que se describen la definición del problema, la justificación y los objetivos planteados.

En el capítulo 4, se hace una contextualización del marco referencial en el que se realiza una sencilla definición de los temas sobre los que se sustenta la implementación del presente proyecto.

El capítulo 5, se hace una descripción del marco metodológico implementado para el cumplimiento de los objetivos propuestos, por medio de una distribución de actividades organizadas en tres fases generales.

El capítulo 6, describe el desarrollo de la investigación, brindando los detalles del proceso realizado, mediante el desarrollo de las 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.

El capítulo 7, está relacionado con las conclusiones, basadas en las experiencias vividas durante la realización del proyecto y los resultados obtenidos.

Para finalizar se relacionan las fuentes de información utilizadas como base para la realización del presente documento, tales como libros, artículos, memorias y páginas web.

1 DEFINICIÓN DEL PROBLEMA

1.1 ANTECEDENTES

Partiendo de la preocupación de que los estudiantes tienen un bajo nivel de motivación hacia el aprendizaje de la programación, además de las dificultades de diferente índole que les presentan, y más aún los estudiantes que no tienen conocimientos en computación (Fesakis & Serafeim, 2009), es importante brindar nuevos aportes en la enseñanza de la programación, siendo conscientes que han surgido herramientas y novedosos enfoques que contribuyen al mejoramiento del proceso enseñanza-aprendizaje de la programación de computadores y que corresponden a los avances tecnológicos y a las exigencias del mundo actual (Lopez, et al., 2013).

En el ámbito de la educación en ciencias de la computación, se ha identificado que los estudiantes encuentran dentro del proceso de aprendizaje de la programación de computadores dificultades de diferente índole, sobre todo en los cursos introductorios en donde se enseñan los conceptos básicos; por esta razón investigadores y docentes relacionados con esta área de conocimiento, manifiestan su preocupación por identificar los problemas que presentan los estudiantes cuando están aprendiendo a programar (Moström, 2011).

Tomando como base el estudio hecho por el Instituto de Sistemas de Software de la Universidad de Tecnología de Tampere-Finlandia, es posible identificar que el principal problema de los programadores novatos no parece ser la comprensión de los conceptos básicos, sino el aprender a aplicarlos, por tanto es de vital importancia que esto conduce a que se presente un bajo nivel de motivación en los estudiantes cuando se enfoca la enseñanza en su mayoría al aprendizaje teórico (Lahtinen, Ala-Mutka, & Järvinen, 2005).

1.1.1 Referentes a nivel internacional. En Estados Unidos en la California Polytechnic State University, tomando como base un estudio que estima que entre 2008 y 2018, Estados Unidos tendrá 1,4 millones de ofertas de trabajo en los campos de la informática y que a partir del año 2000 el número de estudiantes matriculados en carreras de informática disminuyó en un 50%, se realiza una tesis orientada a realizar una comparación de dos plataformas de computación, por un lado Scratch y por otro Arduino, las dos son plataformas de desarrollo de software interactivas para la enseñanza de los conceptos básicos introductorios de programación (Beug, 2012).

Para incrementar el nivel de motivación de los estudiantes se han buscado estrategias y métodos de enseñanza novedosos, por medio de ambientes de aprendizaje interactivos, que brinden herramientas para la enseñanza de los conceptos de programación de forma práctica obteniendo resultados positivos como los conseguidos en el Departamento de Ciencias de la Computación de la Universidad del Cairo (Salim et al., 2010), donde decidieron implementar el uso de Alice 3D, herramienta desarrollada por un equipo de investigadores de la Carnegie Mellon University, con el propósito de ofrecer un ambiente de programación innovador, diseñado para ser el primer acercamiento que tenga un estudiante con la programación orientada a objetos, los estudiantes pueden realizar programas con instrucciones correspondientes a los lenguajes de programación orientada a objetos: Java, C++ y C# (W. P. Dann, Cooper, & Pausch, 2012).

En el estado de California, participaron voluntariamente 325 estudiantes de siete colegios públicos, durante 2 años en el aprendizaje de los conceptos de ciencias de la computación con Alice y Scratch, para estudiar cómo puede la programación de juegos promover el pensamiento computacional en estudiantes de secundaria (Werner, Campe, & Denner, 2012).

1.1.2 Referentes a nivel nacional. Como se puede observar a nivel internacional, las dificultades presentadas por los estudiantes en el aprendizaje de la programación, se ven reflejadas en el bajo nivel de estudiantes que optan por terminar un programa académico relacionado con las ciencias de la computación, y Colombia no es la excepción, y a través del Ministerio de Educación Nacional ente que tiene dentro de sus funciones gestionar el Sistema de aseguramiento de la calidad de la educación superior, siendo un factor muy importante el ingreso, permanencia y graduación de estudiantes, en los diferentes programas académicos ofrecidos en el país, este ministerio por medio del Consejo Nacional de Acreditación, incluye dentro de los lineamientos para la acreditación de programas de pregrado en el Factor 11: Bienestar Institucional, en su Característica Nº 32. Permanencia y Retención Estudiantil, con el fin de acreditar de alta calidad a los programas académicos (Consejo Nacional de Acreditación, 2013).

Es preocupante que en informes emitidos por el Ministerio de Educación Nacional en el Boletín No. 14 Febrero 2010 se presenten en los programas de Ingeniería de Sistemas, Telemática y Afines, niveles de deserción por cohorte en el primer semestre una tasa del 27,36%, en quinto semestre del 55,24% y en décimo semestre del 64,45%, siendo estas las tasas más altas de

deserción dentro del área del conocimiento de Ingeniería, Arquitectura, Urbanismo y Afines (Ministerio de Educación Nacional, 2010).

Por tanto, el presente proyecto surge de la intención de motivar a los estudiantes de Ingeniería de Sistemas y Computación, en la asignatura de Algoritmos y Programación, de la Universidad Pedagógica y Tecnológica de Colombia, para que mejoren su nivel de aprendizaje de los conceptos de programación de computadores, a través de una herramienta interactiva de código abierto que facilite la transición a la codificación de algoritmos en un lenguaje de programación orientado a objetos.

1.2 DESCRIPCIÓN

En la Escuela de Ingeniería de Sistemas y Computación de la Universidad Pedagógica y Tecnológica de Colombia, se ha manifestado la preocupación por los bajos niveles de rendimiento de los estudiantes en las asignaturas relacionadas con la programación de computadores, se ha detectado que muchos estudiantes principiantes, llegan a cursar sus estudios de educación superior sin conocimientos de computación y les resulta complicado aplicar los conceptos teóricos en la construcción de algoritmos en los lenguajes de programación, comprender el paradigma de programación orientada a objetos y aplicarlo usando buenas prácticas de programación.

Con el propósito de mejorar los procesos académicos, el programa 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 POO, 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 de forma práctica, obteniendo resultados positivos.

1.3 FORMULACIÓN

Partiendo de las experiencias exitosas del uso de entornos interactivos la presente investigación busca dar respuesta a la pregunta: ¿Cuál es el efecto de usar software libre en la enseñanza de los conceptos básicos de la Programación de Computadores y la orientación a objetos, en un grupo de la asignatura Algoritmos y Programación?

1.4 HIPÓTESIS

El uso de una herramienta interactiva de programación de código abierto, para la enseñanza de los conceptos básicos de la programación de computadores y la orientación a objetos, mejorará el nivel de aprendizaje de un grupo de estudiantes de la asignatura Algoritmos y Programación de la UPTC.

No se presentan mejoras significativas en el proceso de aprendizaje de la programación, usando una herramienta interactiva de programación de código abierto.

1.5 ALCANCE Y LIMITACIONES

La presente investigación busca evaluar el efecto que tiene el uso de una herramienta interactiva en la enseñanza de la programación, con un grupo de estudiantes de la asignatura algoritmos y programación, los conceptos básicos de programación y del paradigma de orientación a objetos que se abordan, están delimitados por los contenidos temáticos establecidos por la EISC. Los conceptos básicos de programación utilizados son: variables, operadores matemáticos, lógicos, relacionales y de asignación, declaración de sentencias, estructura secuencial, estructura condicional, estructuras de repetición, arreglos unidimensionales.

En cuanto a los conceptos fundamentales del paradigma de la orientación a objetos, se delimitan por los siguientes: clase, atributos, métodos, objetos, relaciones entre clases, arreglos de objetos.

Solamente se realizó la experimentación en un grupo ya conformado, debido a que en la EISC, se tiene la política de que un docente oriente un grupo correspondiente a la misma asignatura, y por ende se decidió evaluar el resultado de una prueba final del grupo experimental AYP-EXP con respecto a los resultados obtenidos por los otros cuatro grupos de la asignatura Algoritmos y Programación.

2 JUSTIFICACIÓN

En la actualidad para los programas académicos es muy importante recibir la acreditación de calidad, y para ello adicional a otros factores, deben demostrar la aplicación de métodos de enseñanza y aprendizaje acordes con la metodología y con las posibilidades tecnológicas que vayan surgiendo, y que suplan las necesidades de los estudiantes en atención a su diversidad.

Es así que el desarrollo de la presente investigación contribuye con la exploración de una estrategia alternativa, que incluye el uso de una herramienta interactiva en el proceso de enseñanza de la programación, donde se ha detectado que se presentan diversos inconvenientes en el aprendizaje de conceptos.

Al ser una investigación de tipo cuasi-experimental se abre la posibilidad de explorar el proceso y nivel de aprendizaje de los estudiantes, con el fin de reflexionar sobre las metodologías utilizadas tradicionalmente, y plantear un modelo que estandarice el proceso de enseñanza-aprendizaje en las diferentes asignaturas de línea de programación del programa de Ingeniería de Sistemas y Computación, beneficiando al estudiante al utilizar una metodología homogénea utilizada por los diferentes docentes del área de programación.

Teniendo en cuenta que la EISC, se encuentra en proceso de autoevaluación, los resultados del presente proyecto sirven como base para analizar los contenidos programáticos y pensar en la reestructuración de los mismos, en pro del mejoramiento de los procesos académicos del programa.

3 OBJETIVOS

3.1 OBJETIVO GENERAL

Evaluar el uso del Software Libre en la enseñanza de los conceptos básicos de la Programación de Computadores, para identificar una estrategia alternativa en el proceso enseñanza-aprendizaje.

3.2 OBJETIVOS ESPECÍFICOS

- Revisar la literatura sobre el uso del Software Libre en la enseñanza de los conceptos básicos de la Programación de Computadores, con el fin de identificar una herramienta que será usada en la enseñanza de los conceptos básicos de la Programación de Computadores.

- Implementar una estrategia orientada a la enseñanza de los conceptos básicos de la Programación de Computadores, usando Software Libre, con un grupo de estudiantes de Ingeniería de Sistemas y Computación de la Universidad Pedagógica y Tecnológica de Colombia.

- Evaluar la implementación de la estrategia orientada a la enseñanza de los conceptos básicos de la Programación de Computadores, usando Software Libre.

4 MARCO REFERENCIAL

4.1 MARCO CONCEPTUAL

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, conceptos como el de algoritmo, que se ha definido como una secuencia de instrucciones orientadas a resolver un tipo de problema específico, tiene cinco características importantes: finitud, ser definido, entrada, salida y efectividad (Knuth, 1997), el de programa que es considerado como un conjunto de instrucciones escritas en un lenguaje de programación, para realizar diferentes tareas, dichas instrucciones pueden incluir acciones, controles y cálculos (Reynolds, 2009).

Por otra parte es importante resaltar que para la escritura de un programa se debe tener claridad sobre los tipos de datos, que hacen referencia al tipo de información que almacenará una variable, adicionalmente indica la cantidad de espacio en memoria que se debe reservar para los datos, existen los tipos de datos primitivos gestionados por la mayoría de lenguajes de programación, que van desde los datos numéricos (enteros y reales), de tipo carácter, hasta los booleanos (Van Roy & Haridi, 2003).

Como se mencionó los tipos de datos van directamente ligados al concepto de variables, que se toman como los espacios de la memoria del computador que permiten almacenar valores de diferente tipo, y como su nombre lo indica pueden cambiar de valor durante la ejecución del programa, además poseen un identificador o nombre (Van Roy & Haridi, 2003) que para el caso de la programación en lenguaje Java debe cumplir con unas reglas de nombrado, como por ejemplo que inicie en minúscula y si el nombre está compuesto por dos o más palabras las primeras letras de esas palabras inicien en mayúscula, algo adicional muy importante es que el nombre de la variable debe ser significativo y representar la información que está almacenando.

Las variables tienen un valor almacenado el cual puede ser asignado y operado, través de una sentencia, definida como simple acción que se puede realizar en un lenguaje de programación, generalmente finaliza con un punto y coma (Leeds & Weinberg, 2009), en la declaración de las sentencias es

importante el uso de operadores, definidos como un símbolo que representa una operación que será realizada entre uno o más operandos, los operadores pueden ser aritméticos, lógicos, de asignación, relacionales o bit a bit (Jain, 1986).

4.2 MARCO TEÓRICO

Para la presente investigación, es de vital importancia exponer las teorías en la cual se basó, para el cumplimiento de los objetivos planteados, que a nivel general están relacionadas, con las estructuras de programación, el paradigma de la programación orientada a objetos, la arquitectura Modelo Vista Controlador, los lenguajes de programación y el Software Libre.

4.2.1 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; existen tres tipos de estructuras de programación, la secuencial, condicional o de selección y las de iteración o repetición.

4.2.1.1 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.

4.2.1.2 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.

4.2.1.3 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”(Eckel, 2007).

Otras de las teorías que afecta de manera importante el desarrollo del trabajo es el paradigma de la programación orientada a objetos, el cual toma como concepto principal el objeto, el cual según (Eck, 2010) 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.

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.

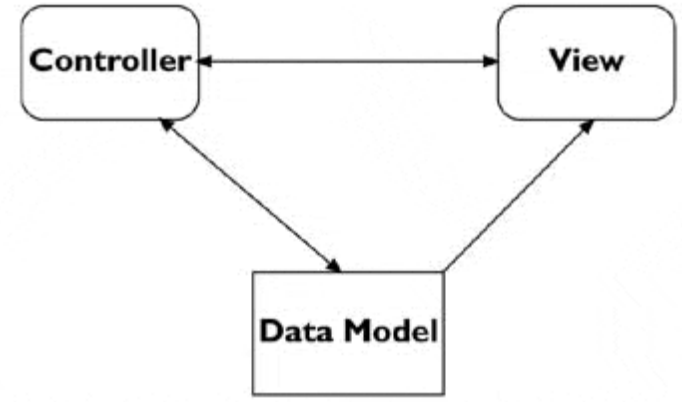
Es común que un objeto pertenezca a la misma familia o clasificación de otro objeto, lo que conlleva a decir que los objetos que poseen el mismo tipo de datos y responden a los mensajes de la misma forma, pertenecen a la misma clase, esta clase describe un grupo de objetos en particular que tienen idénticas características (datos) y comportamientos (funcionalidad), aplicando buenas prácticas de programación, para definir el nombre de una clase según Martin (Martin, 2008) 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.

Para una clase, que como la define (Eckel, 2007), es un tipo de dato, debido a que igual que un número en coma flotante posee un conjunto de características y comportamientos, en POO las características se denominan atributos, y los comportamientos como métodos, los cuales se describen como la serie coordinada de sentencias para llevar a cabo una acción, y en su defecto gracias a que de una clase se pueden crear tantos objetos, como se desee, los métodos de esa clase pertenecen también a los objetos que se creen de dicha clase.

4.2.1.4 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.

Figura 1. Patrón Modelo Vista Controlador



Fuente: (J. W. Cooper, 2000)

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 (J. W. Cooper, 2000).

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.

Controlador: el propósito de esta capa es actuar como un intermediario entre la capa de la vista de una aplicación y la capa del modelo, es por tanto un canal, por medio del cual en la capa vista se presentan los cambios presentados en el modelo y viceversa. Esta capa del controlador también puede realizar la configuración y las tareas de coordinación para una aplicación y gestionar los ciclos de vida de otros objetos (Lange, 2011).

Dentro de las teorías que portan al presente proyecto se encuentran los lenguajes de programación, los cuales pueden ser considerados como el medio de comunicación entre los seres humanos y el computador, permiten expresar las instrucciones que el programador quiere que el computador ejecute (Pratt & Zelkowitz, 1998). Se categorizan en lenguajes de máquina, de bajo nivel y de alto nivel, en el cual se encuentra categorizado Java, el cual es un lenguaje de programación de alto nivel orientado a objetos que fue comercializado por primera vez en 1995 por Sun Microsystems. Dentro sus características se encuentran que es independiente de la plataforma, capaz de contener varios subprocesos y soportar varios multimedia, ha sido programado para recoger la basura automáticamente, existen muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado, adicionalmente se puede decir que Java es rápido, seguro y fiable. Funciona desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet (Wu, 2009).

Una de las teorías relevantes para este proyecto es la filosofía del Software Libre, porque el ámbito de aplicación es académico, y desde el principio del proyecto se decidió revisar las herramientas de enseñanza de la programación de código abierto, así como el lenguaje de programación, y herramientas CASE para realizar los diagramas requeridos. El término de Software Libre fue definido por fue definido por Richard Stallman, el creador de la Free Software Foundation, quien lo define como: “es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software” (Free Software Foundation, 2001). Se dice además, que un programa es software libre si los usuarios tienen las cuatro libertades esenciales:

La libertad 0, si el usuario posee la libertad de ejecutar el programa como desee, y con el propósito que sea, la libertad 1, si el usuario puede estudiar cómo funciona el programa, y realizarle los cambios para adaptarlo a las propias necesidades, por tanto el acceso al código fuente es muy importante para que se cumpla esta libertad, adicionalmente debe cumplir con la libertad 2, que exige que sea posible redistribuir copias para compartirlo con más

usuarios, y por último la libertad 3, que está orientada a que se puedan distribuir copias a terceros con versiones modificadas del programa.

5 MARCO METODOLÓGICO

5.1 TIPO DE INVESTIGACIÓN

La realización del presente proyecto se hizo a través de la ejecución de tres fases generales, la primera fase orientada a realizar una revisión de herramientas usadas en la enseñanza de la programación, la segunda se enfocó en determinar algunas estrategias usadas en el proceso de enseñanza con la herramienta seleccionada para determinar la estrategia a usar en el desarrollo del proyecto y en la tercera enfocada a analizar los resultados obtenidos para comprobar la hipótesis planteada.

El desarrollo del presente proyecto se realiza dentro de una investigación cuasi experimental, que según Campbell y Stanley (Campbell & Stanley, 1966) 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 diferentes aspectos como: número diferente de estudiantes, estudiantes que están cursando la asignatura por primera, segunda o tercera vez, entre muchos otros que no se pueden controlar.

5.2 DESCRIPCIÓN DEL MÉTODO DE INVESTIGACIÓN

A continuación se describe el proceso llevado a cabo para la realización del proyecto, por medio de las tres fases descritas en la Tabla 1.

Tabla 1. Fases generales del proyecto

Nº	FASE	Actividades
I	Identificación y selección de herramientas	• Identificar herramientas de Software Libre usadas en la enseñanza de los conceptos básicos de la Programación de Computadores. • Caracterización de las herramientas libres usadas en la enseñanza de los conceptos básicos de la Programación de Computadores y definición de criterios de evaluación. • Evaluación comparativa de las herramientas identificadas. • Selección de la herramienta que obtuvo los mejores resultados en la evaluación comparativa.

Fuente: Autor

Tabla 2. Fases generales del proyecto (Continuación)

Nº	FASE	Actividades
II	Implementación de estrategia de enseñanza	• Identificar las estrategias usadas en la enseñanza de los conceptos básicos de la Programación de Computadores, usando la herramienta de Software Libre seleccionada. • Determinar una estrategia para enseñar los conceptos básicos de la Programación de Computadores. • Diseñar talleres y prácticas para enseñar los conceptos básicos de programación, a través de la herramienta seleccionada. • Aplicar la estrategia de enseñanza de los conceptos básicos de la Programación de Computadores
III	Evaluación de la implementación de la estrategia	• Diseñar encuesta para aplicarla con el grupo AYP-EXP • Diseñar cuestionario final para aplicarla a los grupos AYP-EXP, AYP-G1, AYP-G3, AYP-G4, AYP-G5 • Aplicar las encuestas y talleres a los dos grupos seleccionados de la asignatura Algoritmos y Programación. • Analizar los resultados obtenidos en la aplicación de la encuesta y el cuestionario final.

5.2.1 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 orientado 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.

5.2.2 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, debido a que dos estudiantes en el transcurso del semestre no terminaron de asistir a las sesiones de clase, 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. El grupo está conformado de 17 estudiantes, pero dos de ellos aproximadamente iniciando la segunda mitad del semestre no volvieron a asistir a las sesiones de clase, debido a que manifestaron que se iban a cambiar de carrera porque estaban repitiendo las asignaturas de Algoritmos y Programación y Cálculo Diferencial, y que optaban por un programa académico que no incluyera tanta matemática. La encuesta realizada puede verse en detalle en el Anexo A.

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. Para ver el cuestionario aplicado, remitirse al Anexo B.

5.2.3 Proceso de análisis de resultados. Los resultados de las encuestas fueron tabulados, y graficados para evaluar los resultados obtenidos. Los resultados del cuestionario fueron exportados desde la cuenta de moodle de cada uno de los docentes, con los resultados obtenidos por parte de cada uno de los estudiantes en una hoja de cálculo, se calculó la media aritmética de cada uno de los grupos, estos resultados se graficaron con el objetivo de comparar el rendimiento de cada uno de los grupos con respecto a los resultados obtenidos por el grupo AYP-EXP, y determinar si el proceso de aprendizaje de los contenidos de la asignatura fueron o no mejores en el grupo experimental, y si se comprueba la hipótesis planteada.

5.2.4 Herramientas informáticas utilizadas.

Es importante mencionar las herramientas que se utilizaron en el desarrollo del presente proyecto:

Entorno de programación: Alice Versión 3.2.5.0.0

Lenguaje de Programación: JAVA

Lenguaje de Modelado: Adaptación de UML

Editor de texto: Notepad++

Plataforma de aprendizaje: moodle

6 RESULTADOS

Como se comentó en el capítulo anterior el proyecto se desarrolló, a través de tres fases, las cuáles son descritas a continuación:

6.1 FASE I: REVISIÓN DE LA LITERATURA SOBRE EL USO DEL SOFTWARE LIBRE EN LA ENSEÑANZA DE LOS CONCEPTOS BÁSICOS DE LA PROGRAMACIÓN DE COMPUTADORES.

Partiendo de la preocupación que han manifestado las diferentes universidades al detectar que el rendimiento de los estudiantes en las asignaturas de programación es bajo, y aprovechando que se han creado herramientas interactivas que sirven como recurso a los docentes para implementar estrategias alternativas para la enseñanza de los conceptos básicos de la programación y adicionalmente los relacionados con la programación orientada a objetos. Se ha percibido según lo estipulado por (UTTING, COOPER, KOLLING, MALONEY, & RESNICK, 2010) y por (Losada, 2012) que existen diversas entornos de programación y dentro de las que ellos catalogan como más utilizados se encuentran Jeroo, Greenfoot, Scratch, Alice, DrJava, Stencyl, StarLogoTNG, Karel, entre otros.

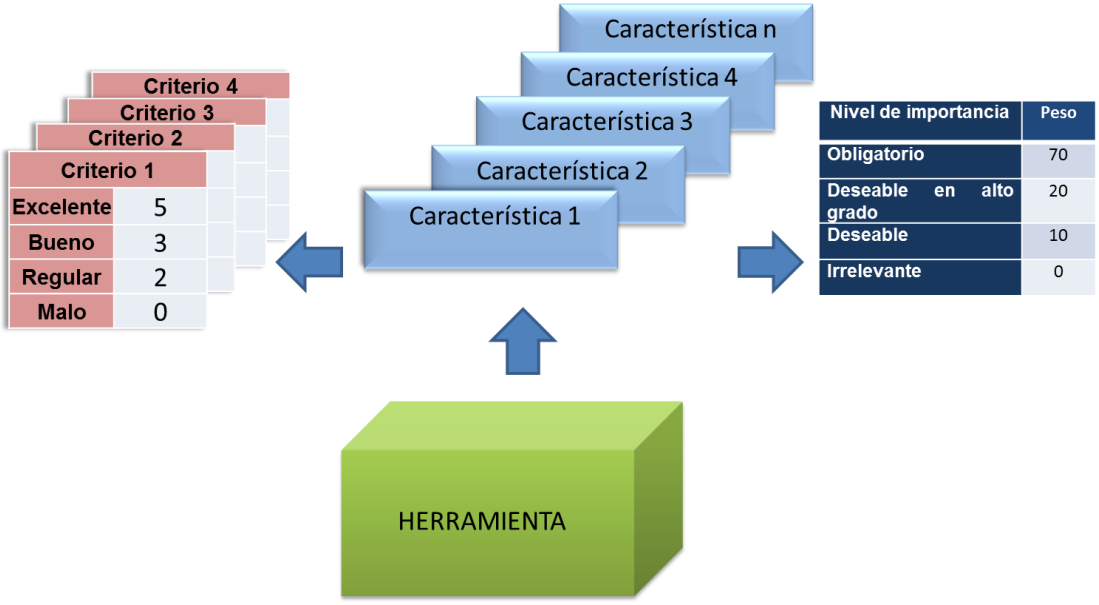
Teniendo en cuenta la diversidad de entornos de programación, se consideró pertinente realizar la identificación de las características de cada una de estas herramientas, con el fin de seleccionar la herramienta que más se adaptara a los contenidos programáticos y las políticas de enseñanza usadas en la Escuela de Ingeniería de Sistemas y Computación de la Universidad Pedagógica y Tecnológica de Colombia-UPTC.

6.1.1 Descripción de la metodología a usar en la evaluación de herramientas.

A continuación se describe el proceso realizado en la identificación de las herramientas interactivas de Software Libre, usadas en la enseñanza de los conceptos básicos de la Programación de Computadores y la orientación a objetos, debido a que se encuentran una variedad de este tipo de herramientas se vuelve compleja su selección, en especial cuando se busca características especiales que se adecuen a las necesidades del proyecto, por lo tanto hizo necesario realizar la primera fase del proyecto de acuerdo al esquema que se observa en la Figura 1, basado en él, 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 (Universidad Pedagógica y Tecnológica de Colombia, 2011), se realizaron los respectivos estudios y posterior selección.

Figura 2. Esquema de Comparación



Fuente: Autor

Con base a este esquema se realiza el estudio comparativo de las herramientas interactivas, que dan soporte al proceso de enseñanza de los conceptos básicos de programación y la orientación a objetos, para ello se utilizó la metodología descrita a continuación:

6.1.1.1 Primer paso. Se definen los niveles de importancia y el peso correspondiente a cada uno de ellos, ver Tabla 3, a partir de los cuales se determina el grado de importancia que poseerá la presencia de cada característica en cada una de las herramientas, con base a los lineamientos de enseñanza establecidos por la Escuela de Ingeniería de Sistemas y Computación y las ponderaciones usadas en los procesos de acreditación de la UPTC, estos valores fueron asignados buscando que las herramientas que contengan las características necesarias en el desarrollo del proyecto obtengan un puntaje diferencial respecto a las que no, y que aquellas que ofrecen funcionalidad adicional puedan mejorar su puntaje y así contar con la mejor selección posible.

Tabla 3. Nivel de importancia de características

Nivel de importancia	Peso
Obligatorio	70
Deseable en alto grado	20
Deseable	10
Irrelevante	0

Fuente: Autor

6.1.1.2 Segundo Paso. Se relacionan las características de cada tipo de herramienta que serán evaluadas para el desarrollo del estudio comparativo, definiendo su obligatoriedad respecto a los lineamientos de la EISC.

Tabla 4. Características a evaluar

CARACTERÍSTICA	OBLIGATORIA
Característica 1	SI
Característica 2	SI
Característica 3	NO
Característica 4	SI
Característica 5	NO

Fuente: Autor

6.1.1.4 Tercer paso. Definidas las características a evaluar, se procede a calificar cada una de acuerdo al nivel de importancia y se calcula su valor de ponderación (Tabla 4).

Tabla 5. Ponderación de características.

CARACTERÍSTICA	NIVEL DE IMPORTANCIA	PONDERACIÓN
Característica 1	Valor1	Porcentaje de importancia 1
Característica 2	Valor2	Porcentaje de importancia 2
Característica 3	Valor3	Porcentaje de importancia 3

Fuente: Autor

Tabla 6. Ponderación de características. (Continuación)

CARACTERÍSTICA	NIVEL DE IMPORTANCIA	PONDERACIÓN
Característica 4	Valor4	Porcentaje de importancia 4
Característica 5	Valor5	Porcentaje de importancia 5
Totales	**Total_Valores**	**Porcentaje total de importancia = 100%**

Fuente: Autor

En la Tabla 4, la columna de *característica,* se refiere a cada una de las características a evaluar para cada herramienta, la columna nivel de importancia, representa el estimado que posee cada característica, se basa en la tabla Nivel de Importancia de características (Tabla 2), y de su obligatoriedad. A partir de las anteriores se obtiene los valores de la columna de ponderación, de la siguiente forma:

Se suma todos los valores correspondientes a la columna *Nivel de Importancia*, en donde se obtiene el valor de la celda *Total_Valores*.

Ecuación 1. Fórmula total de valores para la ponderación

$$Total\_Valores \quad = \sum_{1}^{n} Valor(n)$$

A partir del anterior valor, se aplica la siguiente fórmula:

Ecuación 2. Fórmula ponderación

$$Ponderación = Valor * \frac{100}{(Total\_Valores)}$$

Esta fórmula (Ecuación 2.) es aplicada a cada una de las características evaluadas.

6.1.1.4 Cuarto paso. Para poder calificar las características se definen unos criterios por cada una de ellas, que serán valorados de acuerdo a su desempeño según una escala general de evaluación (Tabla 5).

Tabla 7. Escala general de evaluación.

Criterio	Peso
Excelente	5
Bueno	3
Regular	2
Malo	0

Fuente: Autor

Para cada característica, se definen los criterios específicos a tener en cuenta para la asignación del puntaje (Tabla 6).

Tabla 8. Criterios específicos de evaluación.

Característica n	
Peso	**Criterio**
5	Criterio 1.
3	Criterio 2.
2	Criterio 3.
0	Criterio 4.

Fuente: Autor

Como se puede observar, cada criterio definido posee un peso correspondiente a los valores contemplados en la Escala General de Evaluación (Tabla 6).

6.1.1.5 Quinto paso. A partir de la declaración planteada en la tabla anterior, 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, plasmados en la Tabla 7.

Tabla 9. Resultados consolidados.

PONDERACIÓN	CARACTERÍSTICA	Herramienta1	Herramienta2	Herramienta3	Herramienta4
Porcentaje1	Característica1	V1H1[1]	V1H2	V1H3	V1H4
Porcentaje2	Característica2	V2H1	V2H2	V2H3	V1H4
Porcentaje3	Característica3	V3H1	V3H2	V3H3	V3H4
Ponderación total = 100%	**RESULTADOS**	**Resultado1**	**Resultado2**	**Resultado3**	**Resultado4**

Fuente: Autor

Las celdas concernientes a V1H1, V1H2, V1H3, V1H4, representan los valores correspondientes a la evaluación de la característica 1 presentes en cada una de las herramientas escogidas, basados en los criterios específicos de evaluación de cada característica contemplados en la Tabla 6.

Los resultados obtenidos para cada característica surgen de aplicar la siguiente fórmula:

Ecuación 3. Resultados por característica

$$Resultado1 = \sum_{1}^{n} (VnH1 * Ponderacion\, n))/100$$

A partir de dichos resultados se toman las decisiones concernientes para elegir la herramienta que mejor se acople a los lineamientos de la EISC para la enseñanza de la programación en la asignatura de Algoritmos y Programación, los cuales se exponen a continuación:

- Paradigma de Programación Orientada a Objetos
- Lenguaje de Programación Java
- Desarrollo del pensamiento espacial y lateral
- Implementación de buenas prácticas

[1] V1H1: Valor uno de la característica uno de la herramienta uno

- Edición y compilación de programas Java por consola, sin uso de un entorno de desarrollo integrado.
- Desarrollo por capas.

6.1.2 Estudio comparativo de entornos interactivos de programación. Este segmento hace referencia al estudio realizado para elegir el entorno de programación interactivo que mejor se integrara dentro del proceso de enseñanza del nivel introductorio de la programación, en la EISC.

A partir de los trabajos realizados por (UTTING et al., 2010), en dónde se evalúan herramientas Jeroo, Greenfoot, Scratch, Alice, DrJava, Stencyl, StarLogoTNG, Karel, como los entornos de programación más utilizados. Tomando en cuenta estas herramientas se deciden evaluar a Jeroo, Greenfoot, Scratch, Alice, StarLogoTNG, teniendo en cuenta que BlueJ, Stencyl y Karel son la base y la mejora de algunas de las herramientas seleccionadas, se decide no comparar herramientas que tienen una estructura y funcionalidad muy similar.

Una vez determinado lo anterior se procede a aplicar los criterios de ponderación definidos; de igual forma se determinan las características presentes en los entornos de programación (ver Tabla 8), con base a los lineamientos de la EISC expuestos anteriormente.

Tabla 10. Ponderación de cada característica entornos de programación

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 POO	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**

Definido lo anterior se procede a explicar la importancia de las características representadas en la Tabla 8, así:

- Las características con una importancia del 13%, *Visualización de código Java*, *Manejo de tipos de datos*, *Manejo de estructuras de programación*, *Manejo de conceptos Programación Orientada a Objetos*, *Documentación y Soporte* son obligatorias debido a:

  - ✓ *Visualización de código Java:* con el fin de ir inculcando en los estudiantes la escritura de código en Java, es indispensable que el entorno visualice el código Java que se va generando.

  - ✓ *Manejo de tipos de datos primitivos:* esta característica es indispensable, debido a que la declaración de variables, es uno de los conceptos básicos de la programación y para ello se requiere que la herramienta provea el manejo de los tipos de datos primitivos.

  - ✓ *Manejo de estructuras de programación:* Es indispensable que la herramienta ofrezca los medios para enseñar las estructuras de programación: secuencial, condicional y de repetición.

  - ✓ *Manejo de conceptos Programación Orientada a Objetos:* Es obligatorio que la herramienta dentro de su funcionalidad ofrezca los medios para enseñar los conceptos de la POO, como clase: atributo, método, instanciación de objetos, entre otros, con el fin de orientar y aplicar dichos conceptos en las practicas que se planteen.

  - ✓ Documentación: es de vital importancia que se disponga de documentos que apoyen, tanto a los docentes como a los estudiantes, en el manejo de la herramienta, y en la disposición de material de ejemplo.

  - ✓ Soporte: Es importante que se cuente con diversos medios de soporte, que permitan superar los inconvenientes que se presenten en el uso del entorno de programación.

- Las características con una ponderación del 3,7% de importancia: *Grupo de Edades, Sistema operativo, Entorno 3D, Entrada de texto*

Java, Interfaz Drag and Drop, son características *deseables en alto grado*, ya que para seleccionar la herramienta a usar en la enseñanza de los conceptos básicos de programación es importante conocer a que edades está orientada; por otro lado es importante que la herramienta pueda ejecutarse en diversos sistemas operativos, y dicha herramienta sea independiente del sistema operativo, así mismo, con el propósito de ofrecer un entorno virtual atractivo, es deseable en alto grado, que el entorno de visualización sea en 3D lo que contribuye al desarrollo del pensamiento lateral y que se provea una interfaz drag and drop, evitando la generación de errores de sintaxis. Por último, pero no menos importante encontramos la característica de entrada de texto que permita al usuario editar el código que se va generando, para irse familiarizando con la sintaxis del lenguaje de programación.

- Las características *Comunidad de Usuarios* y *Ejecución inmediata* con una ponderación de 1,9%, son características *deseables* que los entornos de programación posean, sin embargo no tienen influencia directa en el desarrollo del proyecto, sino que aportan funcionalidades adicionales a las características más importantes de las herramientas de entornos de programación y facilitan el proceso de enseñanza aprendizaje de los conceptos básicos de programación y la orientación a objetos.

Tomando como base la Tabla 5, para la definición de la escala general de evaluación, a partir de la cual se pueden definir los criterios específicos de las características de las herramientas de entornos de programación.

Con el objetivo de evaluar cada una de las características propuestas en la Tabla 9, se describen los siguientes criterios de evaluación, para cada característica:

6.1.2.1 Grupo de Edades

- ✓ Definición: Esta característica busca evaluar el rango de edades al cual está orientada la herramienta, acorde al rango de edad de los estudiantes de educación superior.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación.

Tabla 11. Criterios Característica Grupo de Edades

Peso	Criterio
5	Mayor a 15 años
3	N/A
2	Menor a 15 años
0	N/A

Fuente: Autor

6.1.2.2 Sistema Operativo

- ✓ Definición: Esta característica está orientado a identificar los sistemas operativos bajo los cuales las herramientas funcionan adecuadamente.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Windows, Linux, Mac OS

Tabla 12. Criterios Característica Sistema Operativo

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con dos criterios
2	Cumple con un criterio
0	Ningún criterio

Fuente: Autor

6.1.2.3 Visualización de código Java

- ✓ Definición: Determina si las herramientas dentro de sus funcionalidades ofrecen la visualización del código Java.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación.

Tabla 13. Criterios Característica Visualización de código Java

Peso	Criterio
5	Posee
3	N/A
2	N/A
0	No posee

Fuente: Autor

6.1.2.4 Manejo de tipos de datos primitivos

- ✓ Definición: En esta característica se determina si la herramienta permite la declaración de variables con los tipos de datos primitivos.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Tipos de datos enteros, decimales, cadena de caracteres y booleanos.

Tabla 14. Criterios Característica Manejo de tipos de datos

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con tres criterios
2	Cumple con dos criterios
0	Ningún criterio

Fuente: Autor

6.1.2.5 Manejo de estructuras de programación

- ✓ Definición: En esta característica se determina si la herramienta permite la declaración de variables con los tipos de datos primitivos.

- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Estructura secuencial, condicional y de repetición.

Tabla 15. Criterios Característica Manejo de estructuras de programación

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con dos criterios
2	Cumple con un criterio
0	Ningún criterio

Fuente: Autor

6.1.2.6 Manejo conceptos Programación Orientada a Objetos

- ✓ Definición: En esta característica se evalúa que las herramientas dentro de su funcionalidad ofrezca los medios para enseñar los conceptos básicos de la POO.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Creación de clases, métodos, atributos, objetos.

Tabla 16. Criterios Característica Manejo conceptos Programación Orientada a Objetos

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con dos criterios
2	Cumple con un criterio
0	Ningún criterio

Fuente: Autor

6.1.2.7 Documentación

- ✓ Definición: Se evalúa la capacidad de la herramienta de proporcionar documentación que ofrezca información de las funcionalidades y material de apoyo para el proceso de enseñanza-aprendizaje de la programación.

- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Libros, Videos, pdf, wiki, presentaciones, tutoriales, laboratorios.

Tabla 17. Criterios Característica Documentación

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con cuatro criterios
2	Cumple con dos criterios
0	Ningún criterio

Fuente: Autor

6.1.2.8 Soporte

- ✓ Definición: Se evalúa la capacidad de la herramienta de proporcionar documentación que ofrezca información de las funcionalidades y material de apoyo para el proceso de enseñanza-aprendizaje de la programación.

- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Página de ayuda, preguntas frecuentes, listas de correo, solución de problemas, bugs.

Tabla 18. Criterios Característica Soporte

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con cuatro criterios
2	Cumple con dos criterios
0	Ningún criterio

Fuente: Autor

6.1.2.9 Entorno 3D

- ✓ Definición: En esta característica se determina si las herramientas, ofrecen un entorno en 3D permite la declaración de variables con los tipos de datos primitivos.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación.

Tabla 19. Criterios Característica Entorno 3D

Peso	Criterio
5	Posee
3	N/A
2	N/A
0	No posee

Fuente: Autor

6.1.2.10 Comunidad de Usuarios

- ✓ Definición: Esta característica busca evaluar la capacidad que tienen las herramientas de ofrecer los espacios para mantener una comunicación constante entre personas que utilizan las herramientas.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación: Foros, listas de correo, blogs, redes sociales.

Tabla 20. Criterios Característica Comunidad de Usuarios

Peso	Criterio
5	Cumple con todos los criterios de evaluación
3	Cumple con dos criterios
2	Cumple con un criterio
0	Ningún criterio

Fuente: Autor

6.1.2.11 Ejecución inmediata

- ✓ Definición: En esta característica se evalúa que el usuario pueda ir probando de forma inmediata las instrucciones se va realizando.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación.

Tabla 21. Criterios Característica Ejecución inmediata

Peso	Criterio
5	Posee
3	N/A
2	N/A
0	No posee

Fuente: Autor

6.1.2.12 Editor Drag and Drop

- ✓ Definición: En esta característica se evalúa que las herramientas ofrezcan un editor al que se puedan arrastrar las instrucciones, de forma que se pueda crear un programa de una forma rápida y fácil.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación.

Tabla 22. Criterios Característica Editor Drag and Drop

Peso	Criterio
5	Posee
3	N/A
2	N/A
0	No posee

Fuente: Autor

6.1.2.13 Entrada de texto Java

- ✓ Definición: En esta característica se evalúa si las herramientas, ofrecen la entrada de texto Java.
- ✓ Criterios de evaluación: Se plantean los siguientes criterios de evaluación.

Tabla 23. Criterios Característica Entrada de texto Java

Peso	Criterio
5	Posee
3	N/A
2	N/A
0	No posee

Fuente: Autor

Una vez definidos los criterios y su valor dentro de cada característica, se procede a realizar el análisis contemplado en las Tablas 22 y 23. Esta tabla muestra los resultados obtenidos por cada una de las herramientas en una escala de cero a cinco; esto teniendo en cuenta los criterios de evaluación de cada característica evaluada y la ponderación definida según el nivel de importancia dentro del desarrollo del presente proyecto.

Tabla 24. Resultados consolidados entornos de programación

CARACTERÍSTICA	PONDERACIÓN	Greenfoot	Scratch	Alice
Grupo de Edades	3.7	5	2	5
Sistema operativo	3.7	5	5	5
Visualización de código Java	13	5	0	5
Manejo de estructuras de programación	13	5	5	5
Manejo conceptos Programación Orientada a Objetos	13	3	0	3
Manejo de tipos de datos	13	5	3	5
Documentación	13	3	2	5
Entorno 3D	3.7	0	0	5
Entrada de texto Java	3.7	5	0	0

Tabla 25. Resultados consolidados entornos de programación (Continuación)

CARACTERÍSTICA	PONDERACIÓN	Greenfoot	Scratch	Alice
Soporte	13	2	3	5
Interfaz Drag and Drop	3.7	0	5	5
Comunidad de Usuarios	1.9	2	5	5
Ejecución inmediata	1.9	5	5	5
Resultados	**100**	**3.68**	**2.32**	**4.64**

Fuente: Autor

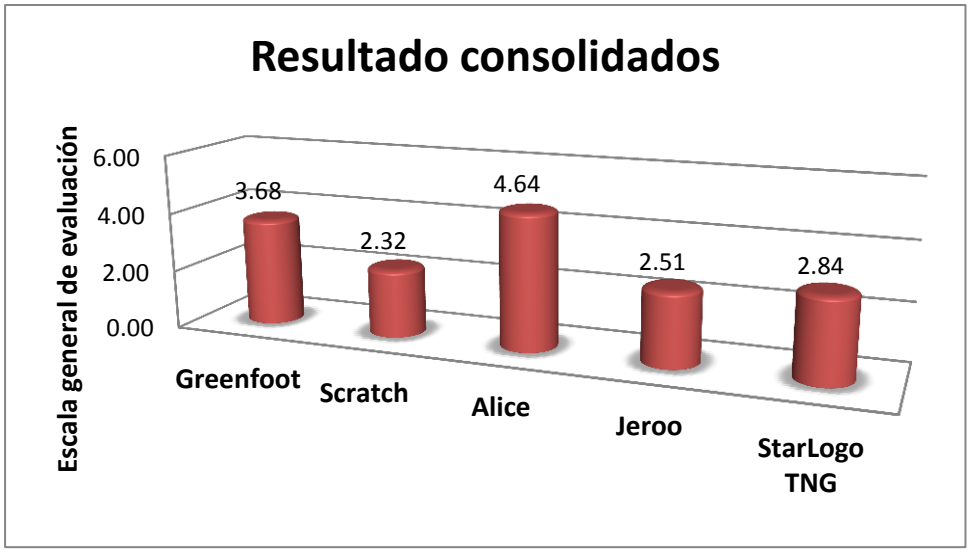
Tabla 26. Resultados consolidados entornos de programación

CARACTERÍSTICA	PONDERACIÓN	Jeroo	StarLogoTNG
Grupo de Edades	3.7	2	2
Sistema operativo	3.7	5	5
Visualización de código Java	13	5	0
Manejo de estructuras de programación	13	5	5
Manejo conceptos Programación Orientada a Objetos	13	3	2
Manejo de tipos de datos	13	0	5
Documentación	13	2	2
Entorno 3D	3.7	0	5
Entrada de texto Java	3.7	0	0
Soporte	13	2	2
Interfaz Drag and Drop	3.7	0	5
Comunidad de Usuarios	1.9	2	2
Ejecución inmediata	1.9	0	5
Resultados	**100**	**2.51**	**2.84**

Fuente: Autor

Los resultados mostrados en la Tabla 22 y 23, pueden observarse de forma consolidada en la Figura 3.

Figura 3. Gráfica de resultados consolidados



Fuente: Autor

La Figura 2 muestra los resultados de cada herramienta, correspondiente a cada una de las barras de la gráfica; los resultados de este estudio expuestos en las Tablas 22 y 23, proveen de la información suficiente para definir las siguientes consideraciones:

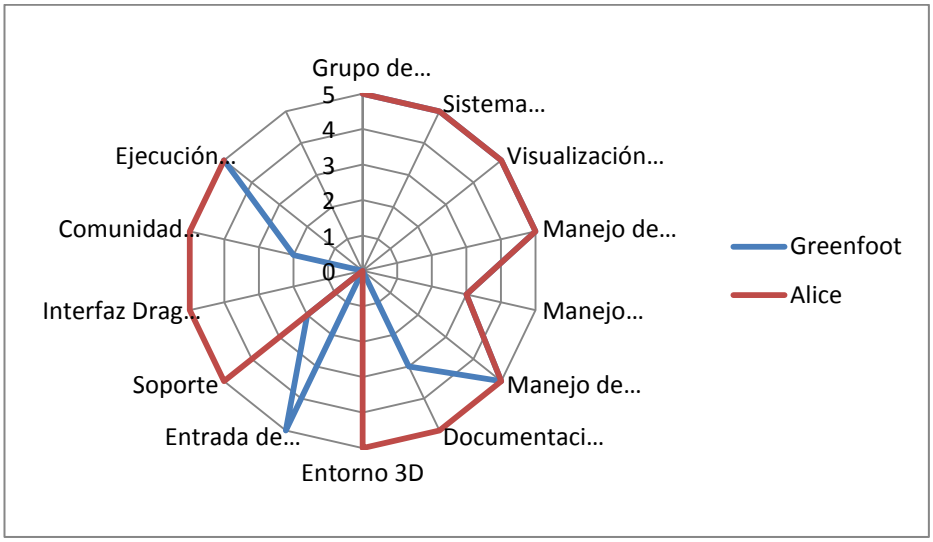
- A partir de los criterios, a través de los cuales se realizó la evaluación de las herramientas, Scratch y Jeroo son las herramientas que obtuvieron los resultados más bajos, por tanto son descartadas para ser implementadas en la enseñanza de la programación y la orientación a objetos en el desarrollo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

Figura 4, correspondientes al análisis de resultados de estas dos herramientas.

Figura 4. Gráfica de resultados herramientas Greenfoot y Alice



Fuente: Autor

Tomando como base la Figura 3, 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.

6.2 FASE II: IMPLEMENTACIÓN DE UNA ESTRATEGIA ORIENTADA A LA ENSEÑANZA DE LOS CONCEPTOS BÁSICOS DE LA PROGRAMACIÓN Y LA ORIENTACIÓN A OBJETOS

Tomando como base los resultados del estudio comparativo expuesto en el numeral 6.1, se procede a realizar una revisión de las estrategias que se han usado en el proceso de enseñanza de los conceptos básicos de la programación y la orientación a objetos con el entorno de programación Alice, esto con el fin de determinar la estrategia de enseñanza que se implementará en el grupo experimental AYP-EXP de la EISC, acorde a los lineamientos establecidos por el programa académico.

6.2.1 Identificación de las estrategias. Dentro de las investigaciones realizadas se encuentra la realizada por (Lucio Castillo, Ramírez-Gil, Garza-Saldaña, García-Mundo, & Vargas-Enríquez, 2011) quienes exponen el trabajo realizado con estudiantes de la carrera de ingeniería en sistemas computacionales del Instituto Tecnológico de Ciudad Victoria-México (ITCV), donde se usaron una muestra de dos grupos, uno experimental y uno de control, en el grupo experimental utilizaron Alice de forma independiente durante el tiempo de cubrimiento de las primeras tres unidades del curso de POO, impartiendo inicialmente tres sesiones de inducción y posteriormente realizando diez prácticas de un manual de ejercicios; como una actividad final, solicitaron a los estudiantes realizar el diseño y planteamiento de una solución a una problemática real.

En el trabajo titulado “Evaluating the Effectiveness of a 3D Visualization Environment While Learning Object Oriented Programming” los autores exponen una investigación realizada en la Universidad King Abdulaziz en Arabia Saudita, en la sección femenina del departamento de Ciencias Computacionales (Allinjawi, Al Nuaim, & Krause, 2014), allí optaron por evaluar los conceptos de la POO, antes y después de usar Alice, esto lo realizaron con un grupo experimental de 29 estudiantes que adicional a las clases tradicionales usaron el entorno de programación Alice y un grupo de control de 28 estudiantes que solamente tomaron clases de la forma tradicional. El taller realizado con Alice en el grupo experimental, lo llevaron a cabo ocho sesiones de 1,5 horas. Luego de este tiempo aplicaron una evaluación a libro abierto en los dos grupos, y tomaron las calificaciones obtenidas por los alumnos para determinar los niveles de aprendizaje del curso.

Este trabajo realizado en Universidad King Abdulaziz, afirma que resultados del análisis del aprendizaje entre los alumnos que habían utilizado Alice en un taller adicional, paralelo a sus clases tradicionales, y los estudiantes que sólo se estaban tomando las clases tradicionales, no reveló diferencias significativas en su capacidad de aprendizaje, a través de todos los conceptos de POO, y concluyen que este efecto pudo darse debido a la falta de correlación entre los ejemplos de resolución de problemas que se enseñan en las clases tradicionales y los ejemplos dados en el taller de Alice.

Por otra parte (S. Cooper, Nam, & Si, 2012), desarrollaron un sistema tutor inteligente (ITS), en Alice, en el que desarrollan trece tutoriales para enseñar conceptos específicos de programación introductorios, creados a través de plantillas. En dichos tutoriales han incluido conceptos como: el paso de parámetros, la anidación de las sentencias de control, la repetición (bucles

definidos e indefinidos), y manejo de listas. Para cada tutorial han creado una historia y declaran un objetivo de la guía de aprendizaje, acorde a al enfoque general que usan en la enseñanza de Alice, que es la resolución de problemas.

Dentro de las investigaciones que se han realizado usando Alice en la enseñanza de la programación Monroy y Arenas (Monroy & Arenas, 2014), exponen que en la Universidad Autónoma de Bucaramanga, han usado Alice en "sesiones prácticas del curso de Fundamentos de Programación (con estudiantes de primer semestre)", 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. Esta metodología de enseñanza fue aplicada a lo largo de dos semestres académicos, para contrastar los resultados con datos históricos de semestres anteriores de los cursos de Fundamentos de la Programación. Como resultado comentan que al utilizar analogías para crear programas en Java, que sean similares a los creados en Alice se logra una apropiación mayor de la sintaxis del lenguaje; sin embargo, se siguen presentando casos en que los estudiantes optan por cancelar el semestre manifestando gran dificultad en la escritura de código.

Por último, 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 (W. Dann, Cosgrove, Slater, & Culyba, 2012). Lo anterior lo sustentan en las teorías de educación citadas por Salomon y Perkings (Salomon & Perkins, 1988) quienes afirman que el aprendizaje debe ser transferible, en otras palabras que lo se aprende en cierto contexto debería ser aplicable en otro contexto.

6.2.2 Determinación de una estrategia para enseñar los conceptos básicos de la Programación de Computadores. Partiendo de la revisión de estrategias citadas anteriormente, en las cuales se puede identificar que algunos optaron por orientar los cursos introductorios usando Alice, otros orientaron la primera mitad del curso usando Alice y la segunda mitad de la forma tradicional, y otros por su parte basados en la estrategia usada por el equipo de desarrollo de Alice, optaron por realizar prácticas de forma paralela a través de analogías, se procedió a realizar una reunión con el Coordinador del área de programación el Ingeniero Germán Amézquita Becerra, exponiendo las diferentes estrategias identificadas y planteando el desarrollo de una nueva

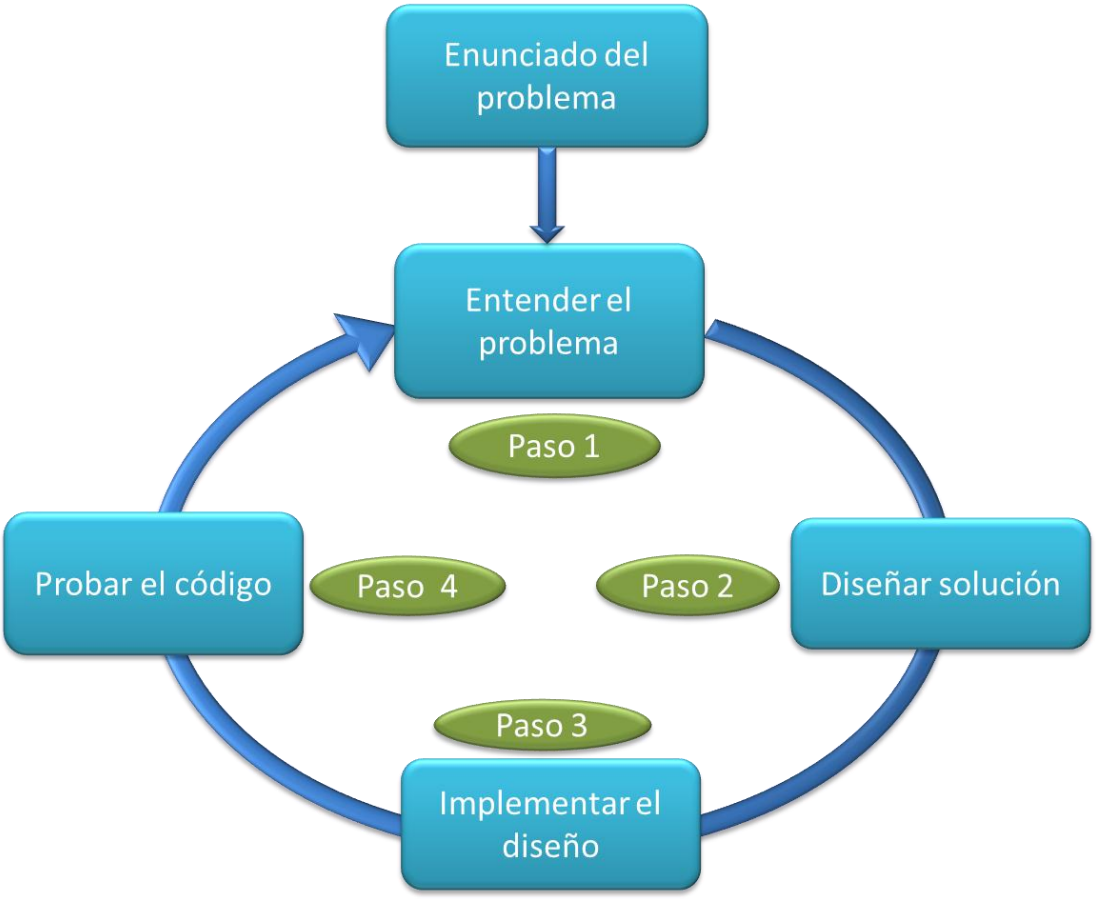
estrategia en la que se tomara como base la estrategia propuesta por el equipo de desarrollo de Alice relacionado con el uso de analogías y la transferencia mediada, junto con la utilizada en la UNAB en la que diseñaron prácticas paralelas en Alice y en Java, adicionalmente se propone utilizar en enfoque 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, como se comentó en el numeral 6.1.1.5, 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.

Como resultado de dicha reunión, se propone el uso de un formato de resolución de problemas, (ver Anexo E) que será utilizado para dar solución a los ejercicios planteados de forma paralela en Alice y en Java, por medio del desarrollo de seis prácticas que constan de una guía desarrollada por la docente autora del presente trabajo, y una evaluación en la que se plantea un ejercicio para desarrollar por los estudiantes mediante el formato de resolución de problemas, tanto en Alice como en Java.

El formato de resolución de problemas es una plantilla diseñada con base a lo planteado por (W. P. Dann, Cooper, & Pausch, 2011), donde se expone que para resolver un problema es necesario realizar principalmente cuatro etapas, basadas en la cuatro primeras fases del ciclo de vida del software: Análisis, Diseño, Implementación y Pruebas, el formato propuesto se puede observar en detalle en la Figura 4, en la que parte del enunciado de un problema, luego se realizan cuatro pasos, partiendo del entendimiento del problema o análisis, seguido del diseño de una solución, la implementación del diseño y por último el paso cuatro encargado de las pruebas del código, si se requiere se realiza nuevamente el ciclo en caso de no obtener los resultados esperados, como lo ilustra la siguiente figura.

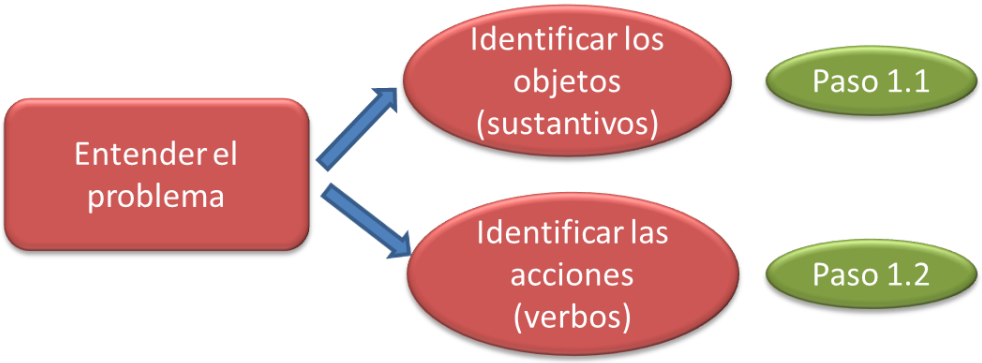
Figura 5. Pasos generales para resolver problemas



Fuente: Autor

En la Figura 5, se plantea un esquema del Paso 1, que está enfocado a tomar el enunciado del problema, y realizar el paso 1.1 para identificar los sustantivos que permitan abstraer los objetos que se requieren para plantear la solución al problema; adicionalmente como paso 1.2, 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, ver Figura 6.

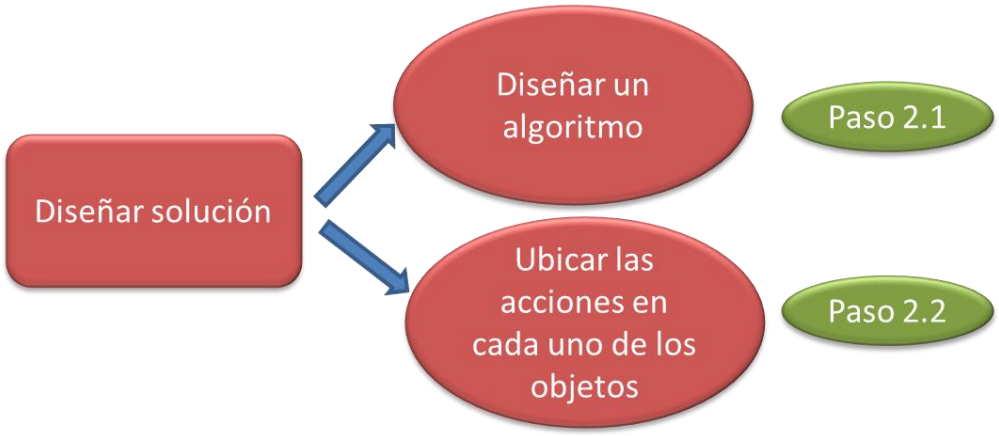
Figura 6. Paso 1. Entender el problema



Fuente: Adaptación de (W. P. Dann et al., 2011)

En la etapa de diseño correspondiente al Paso 2, como se puede observar en la Figura 6, se plantea el Paso 2.1 enfocado a organizar las acciones identificadas en el paso 1.2, en un diagrama de actividades que establezca el orden de ejecución de las acciones. Por otra parte, se plantea el paso 1.2, en el cual 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.

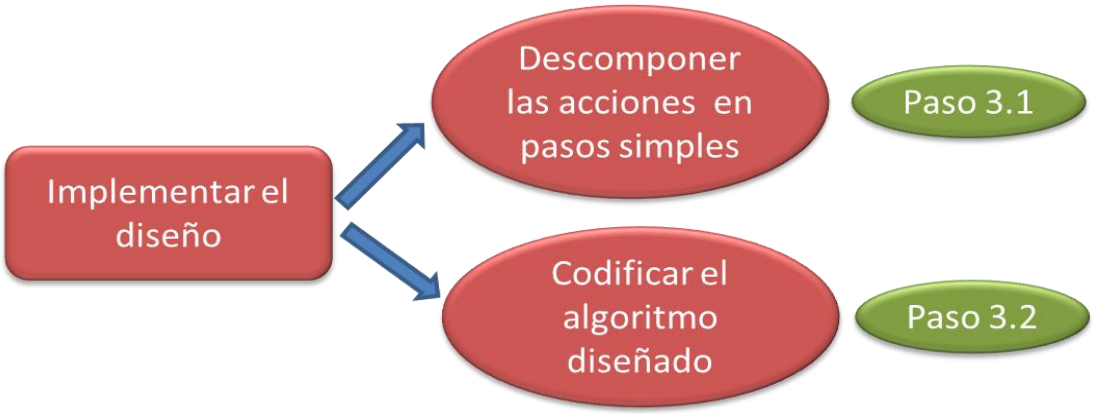
Figura 7. Paso 2. Etapa de Diseño



Fuente: Adaptación de (W. P. Dann et al., 2011)

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 procede a tomar cada acción y descomponerla en pasos simples, que permitan cumplir dicha acción a través del paso 3.1, posteriormente y teniendo en cuenta el diseño del paso 2.2 se procede a realizar el proceso de codificación del algoritmo diseñado en la herramienta correspondiente, ver Figura 7.

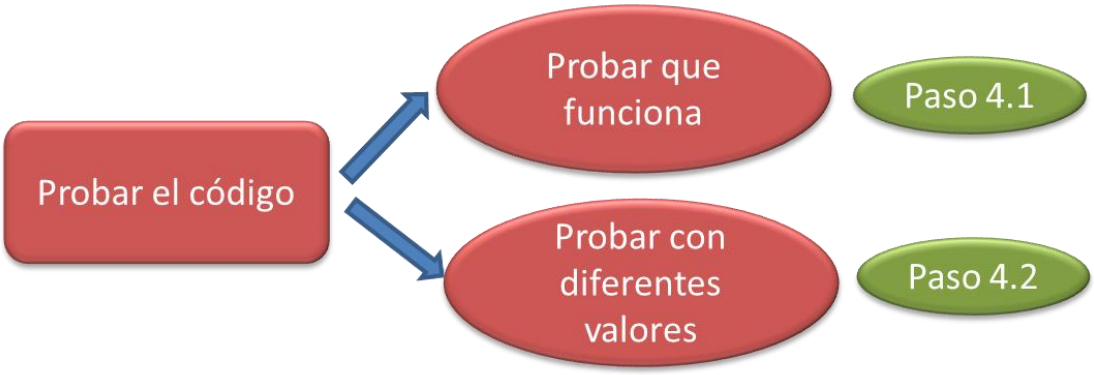
Figura 8. Paso 3 Etapa de implementación del diseño



Fuente: Adaptación de (W. P. Dann et al., 2011)

Este Paso 4, se enfoca a comprobar el funcionamiento del código implementado, iniciando con el paso 4.1, para probar que el código inicialmente funciona; el paso 4.2 se enfoca a ejecutar 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; esta etapa de pruebas es esquematizada en la Figura 9.

Figura 9. Paso 4. Probar el código



Fuente: Adaptación de (W. P. Dann et al., 2011)

6.2.3 Diseño e implementación de las prácticas para enseñar los conceptos básicos de programación, a través de la herramienta seleccionada. Para implementar la estrategia de enseñanza planteada en el numeral 6.2.2, 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.

Como apoyo al diseño de las prácticas se realizó la construcción de una presentación, que expone los conceptos relacionados con cada práctica, con el propósito de exponer ejemplos, reglas de declaración y sintaxis para la ejecución de cada práctica.

Las seis prácticas diseñadas son:

PRÁCTICA 1: ESTRUCTURA SECUENCIAL Y VARIABLES (Ver Anexo F)

PRÁCTICA 2: DECLARACIÓN Y USO DE METODOS (Ver Anexo G)

PRÁCTICA 3: ESTRUCTURA CONDICIONAL

PRÁCTICA 4: ESTRUCTURA DE REPETICIÓN FOR (Ver Anexo H)

PRÁCTICA 5: ESTRUCTURA DE REPETICIÓN WHILE (Ver Anexo I)

PRÁCTICA 6: ARREGLO UNIDIMENSIONAL DE OBJETOS (Ver Anexo J)

A continuación se presenta la plantilla de la guía utilizada en cada una de las prácticas:

Tabla 27. 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

Se presenta un ejemplo de Uso de la plantilla para el caso de la Práctica 3:

Tabla 28. Uso de la plantilla para el caso de la Práctica 3

Nº DE LA PRÁCTICA: 3	**NOMBRE DE LA PRÁCTICA:** ESTRUCTURA CONDICIONAL
OBJETIVOS: • Comprender el concepto y uso de los operadores relacionales. • Comprender el concepto y uso de los operadores lógicos. • Comprender el concepto y uso de las estructuras condicionales • Utilizar la jerarquía de operadores para construir adecuadamente sentencias if / else.	

Fuente: Autor

ENUNCIADO DEL EJERCICIO EN ALICE

Un troll está fuera de la puerta de un castillo, se dirige hacia la puerta, si la puerta está abierta el troll camina hacia el frente donde está una bruja y la saluda solicitándole un hechizo, si la puerta está cerrada, el troll sube su mano derecha y solicita que le abran la puerta diciendo “por favor abran la puerta”, la bruja dice un hechizo “Por la gallina tuerta que se abra ya esta puerta” levantando 90° su mano derecha, y baja la mano 90°, la puerta se abre, y el troll camina hacia la bruja, le reclama subiendo sus mano 180° diciéndole “porque tienes que cerrar la puerta”, la bruja por el mal genio dice un hechizo “Por los vegetales que hay en huerta, que se cierre ya esta puerta” y la puerta se cierra.

Paso 1. Entender el problema

Paso 1.1 Identificar los sustantivos u objetos en el enunciado

Troll, Bruja, Puerta del castillo, Escena

Paso 1.2 Identificar las acciones que se realizan en el enunciado

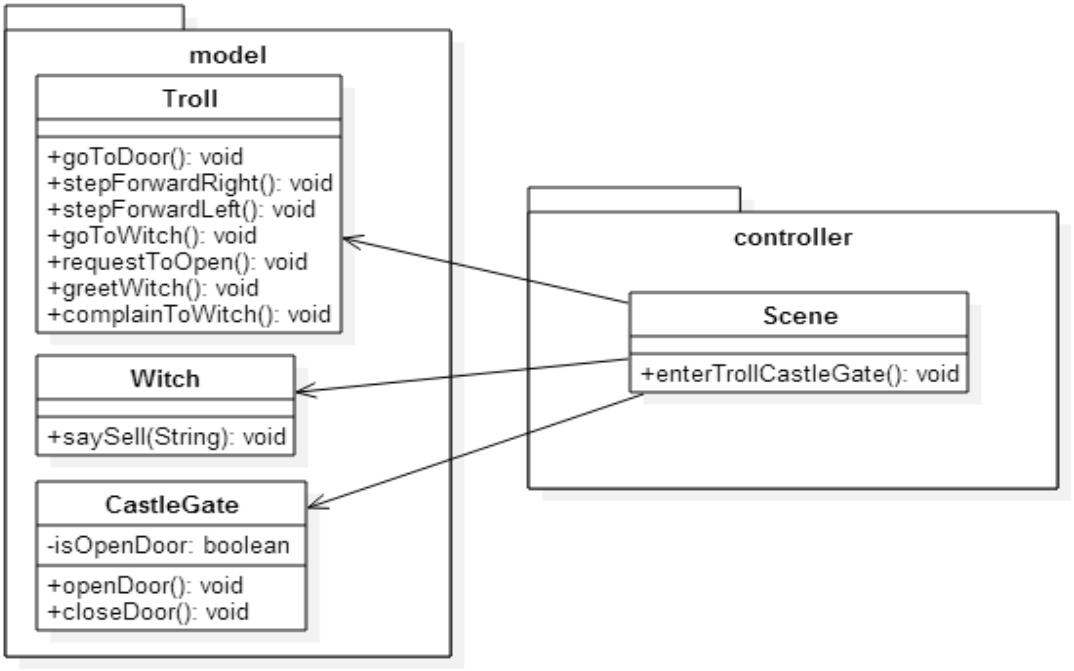
- Troll va hacia la puerta
- Troll camina hacia el frente hacia la bruja
- Troll saluda a la bruja, solicitándole un hechizo
- Troll solicita abrir la puerta
- Bruja dice un hechizo
- La puerta se abre
- Troll le reclama a la bruja

Paso 2. Diseñar una solución

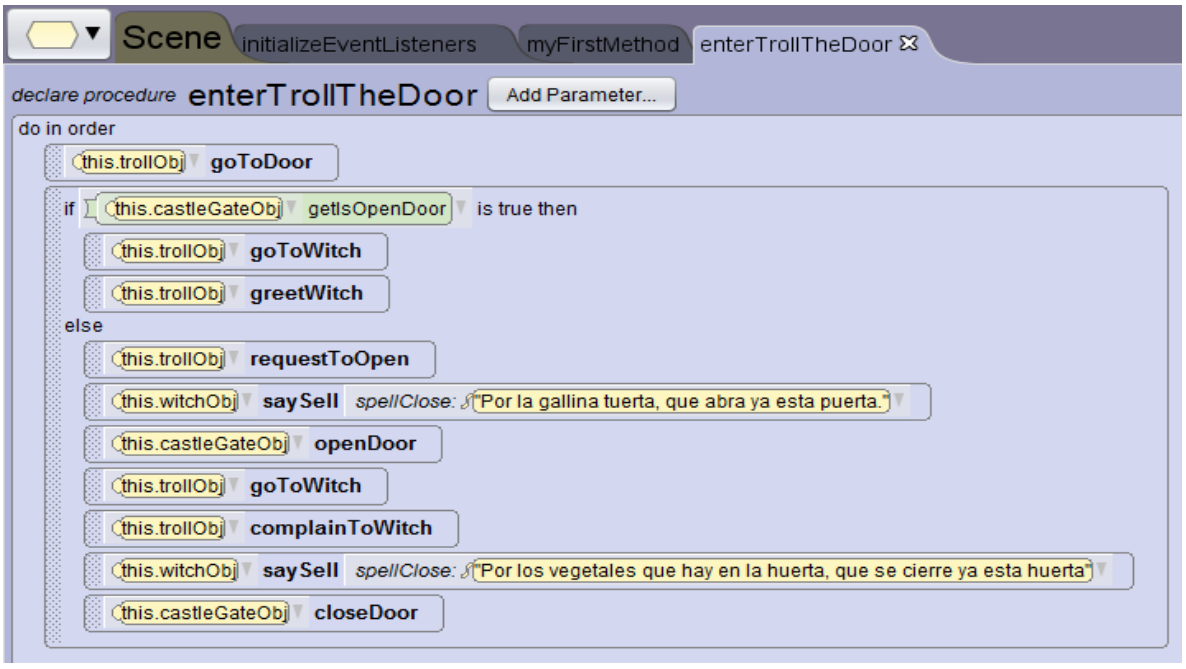
Paso 2.1 Diseñar algoritmo en diagrama de actividades


Troll va hacia la puerta
puerta esta abierta
si no esta abierta
Troll solicita abrir puerta
Bruja dice hechizo
La puerta se abre
Troll va hacia la bruja
Troll le reclama a la bruja
si esta abierta
Troll va hacia la bruja
Troll saluda a la bruja solictando hechizo

Paso 2.2 Ubicar las acciones en cada uno de los objetos que intervienen en el desarrollo del problema, por medio de un diagrama de clases, implementado el patrón de arquitectura Modelo Vista Controlador



Paso 3. Realizar el proceso de codificación del algoritmo diseñado en la herramienta correspondiente



Paso 4. Probar el código

En la imagen se pude observar que al encontrar la puerta cerrada el troll levanta la mano y pide que le abran la puerta, por cuanto el resultado es el esperado

ENUNCIADO DEL EJERCICIO EN JAVA

Se requiere que un estudiante de física, teniendo en cuenta que: un automóvil parte del reposo con una aceleración constante de 3 m/s², si está disponible calcule:

¿Qué velocidad tendrá el automóvil a los 8s de haber iniciado el movimiento?

Y se muestre el resultado de la velocidad por pantalla.

Si no está disponible debe calcular

¿Qué distancia habrá recorrido en ese lapso de 8s?

Y se muestre el resultado de la distancia recorrida por pantalla

Paso 1. Entender el problema

Paso 1.1 Identificar los sustantivos u objetos en el enunciado

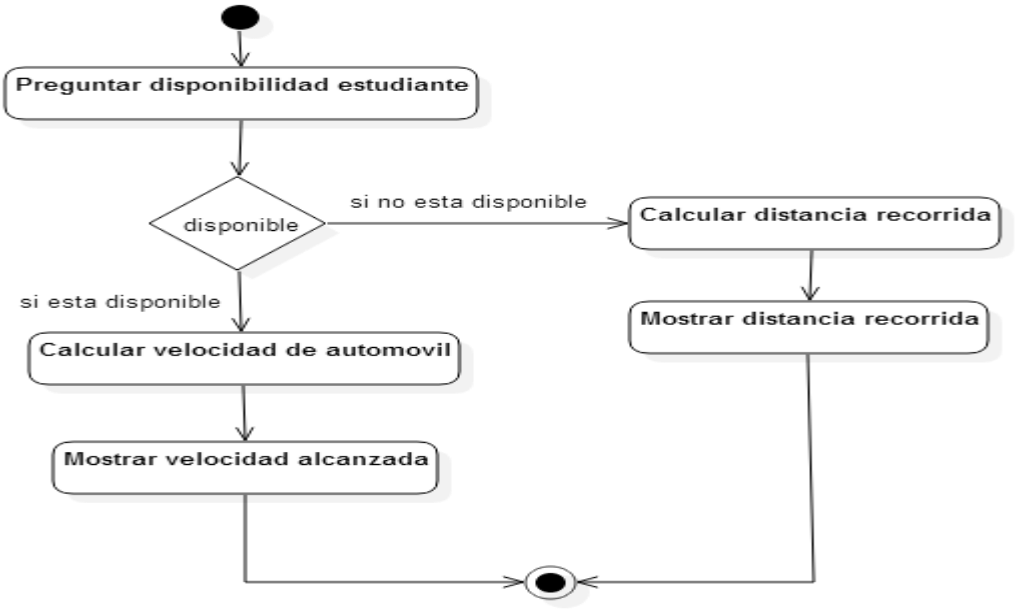
Estudiante de física, Controlador, Gestor de Datos

Paso 1.2 Identificar las acciones que se realizan en el enunciado

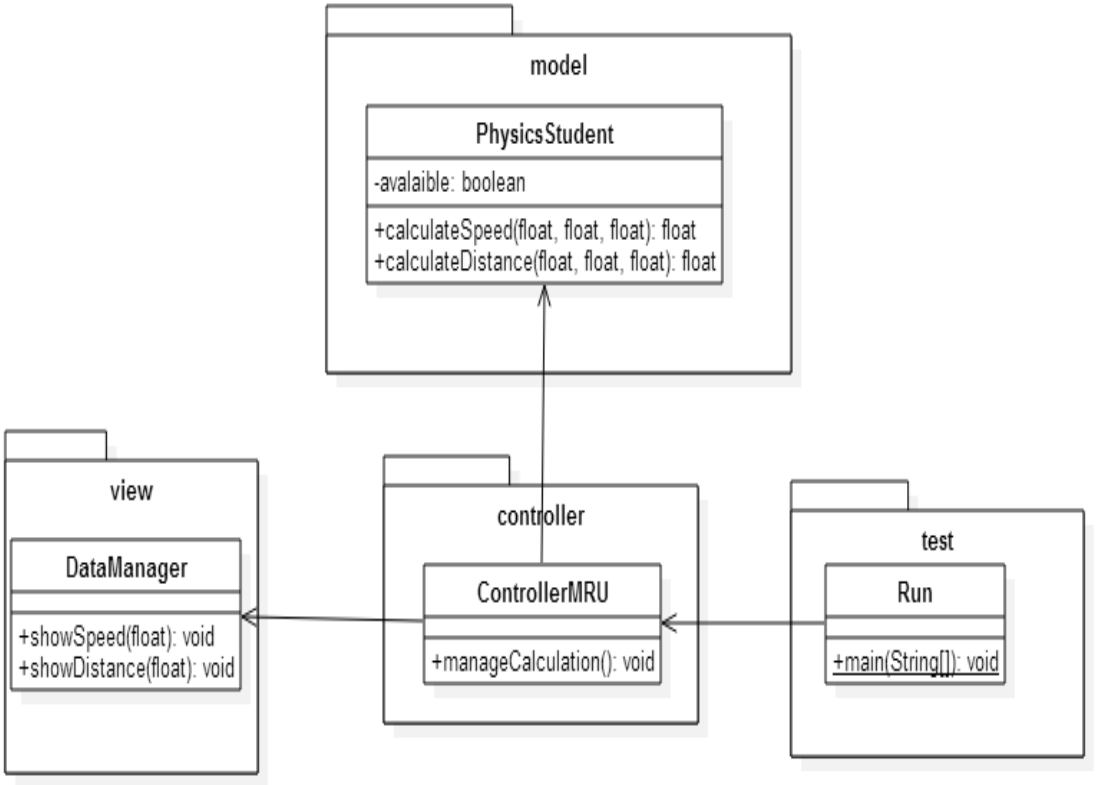
- Preguntar disponibilidad estudiante
- Calcular distancia recorrida por automóvil
- Calcular velocidad del automóvil
- Mostrar velocidad alcanzada
- Mostrar distancia recorrida

Paso 2. Diseñar solución

Paso 2.1 Diseñar un algoritmo: Enumerar las actividades a seguir en un diagrama de actividades



Paso 2.2 Ubicar las acciones en cada uno de los objetos que intervienen en el desarrollo del problema, por medio de un diagrama de clases, implementado el patrón de arquitectura Modelo Vista Controlador



Paso 3. Implementar el diseño

Paso 3.1 Descomponer las acciones en pasos simples

Preguntar disponibilidad del estudiante

- Si está disponible, solicitar a estudiante que calcule la velocidad del automóvil
- Si no está disponible, solicitar a estudiante que calcule la distancia recorrida por el automóvil.

Calcular la velocidad alcanzada por el automóvil

- Multiplicar la aceleración por el tiempo
- Sumar el resultado anterior a la velocidad inicial

Calcular la distancia recorrida por el automóvil

- Calcular el tiempo al cuadrado
- Multiplicar la velocidad inicial por el tiempo
- Calcular el producto de aceleración por el resultado del tiempo al cuadrado y dividirlo en dos.
- Sumar el resultado anterior al producto de la velocidad por el tiempo.

Paso 3.2 Realizar el proceso de codificación del algoritmo diseñado en la herramienta correspondiente

A continuación se muestra el código de la clase PhysicsStudent, correspondiente a la capa del Modelo

```
package model;

public class PhysicsStudent {
   private boolean avalaible;

   public boolean getAvalaible() {
                    return avalaible;
   }

   public void setAvalaible(boolean avalaible) {
         this.avalaible = avalaible;
   }

   public PhysicsStudent() {
         this.avalaible = true;
   }
```

```
    public float calculateSpeed(float aceleration, float time, float speedInitial) {
      float speedFinal= 0.0f;
                    speedFinal = speedInitial + (aceleration * time);
       return speedFinal;
    }

    public float calculateDistance(float aceleration, float time, float speedInitial) {
          float distance = 0.0f;
          float timeSquared = time * time;
          distance = speedInitial*time + (aceleration*timeSquared)/2;
       return distance;
    }
}
```

Paso 4. Probar el código

Paso 4.1 Probar que el código funciona

```
[root@omaira bin]# java test/Run
La velocidad alcanzada es: 420.0
La distancia recorrida es: 5880.0
[root@omaira bin]#
```

Paso 4.2 Ejecutar el programa con diferentes casos de prueba

La salida anterior corresponde a los datos de entrada: aceleración = 15, tiempo = 28 y velocidad inicial = 0

```
[root@omaira bin]# java test/Run
La velocidad alcanzada es: 168.0
La distancia recorrida es: 1176.0
[root@omaira bin]#
```

El resultado presentado en la anterior imagen, corresponde a los datos de entrada: aceleración = 12, tiempo = 14 y velocidad inicial = 0. Comprobando que los resultados son los esperados.

Evaluación Práctica 3

EVALUACIÓN	**EJERCICIO PARA REALIZAR EN ALICE** Una tortuga se dispone a realizar una sesión de ejercicios, la tortuga tiene un peso de 11 kilos y una edad de 9 años, la tortuga levanta una piedra que pesa 7 kilos y hace dos sentadillas, si el peso de la piedra no supera el 50% del peso de la tortuga, si no se cumple la condición la tortuga debe realizar 3 flexiones de brazos.
	EJERCICIO PARA REALIZAR EN JAVA Un estudiante se dispone a presentar un test, el estudiante debe calcular su índice de masa corporal, si su edad es menor a 10 años y el peso corporal es mayor a 37 kilos, si no debe calcular su porcentaje de masa corporal. El resultado de ser visualizado por pantalla.

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[2] 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.

La etapa de análisis planteada en la estrategia, lleva a comprender los conceptos fundamentales de la programación orientada a objetos con la identificación de los sustantivos incluidos en el enunciado del problema, ayudando al estudiante a abstraer los objetos que intervendrán en la solución al problema planteado, esto lleva como resultado a que se identifiquen las clases que harán parte de la solución, de la misma forma, la identificación de las acciones conducen a identificar los pasos que se deben llevar a cabo en el planteamiento del algoritmo.

El resultado de esta etapa de la resolución problemas es el insumo para la etapa de diseño que toma como base las acciones identificadas para plantear en un diagrama de actividades el algoritmo con un orden específico de los pasos identificados; a su vez toma las clases abstraídas para asignar a cada una de ellas las acciones (métodos) que debe realizar cada clase, organizadas en un diagrama de clases aplicando el patrón de arquitectura MVC; al tener el algoritmo diseñado en estos diagramas, se tiene un punto de partida para iniciar el proceso de codificación, ya sea en un entorno interactivo como Alice, o en un editor de texto, y no se comienza de cero a codificar sin tener claridad de lo que se quiere hacer, causando frustración en el alumno al obtener resultados no esperados.

La inculcación de las buenas prácticas de programación en Java, se les reforzó con la ayuda de presentaciones de diapositivas, en la que se enunciaban las reglas de nombrado de variables, clases, métodos, objetos, declaración de las estructuras de programación, vectores de objetos, entre otros.

Los ejercicios de evaluación se plantearon durante las primeras tres prácticas para que fueran desarrollados de forma individual, a partir de la cuarta práctica se permitió que realizaran los ejercicios en grupos de dos estudiantes, esto a

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

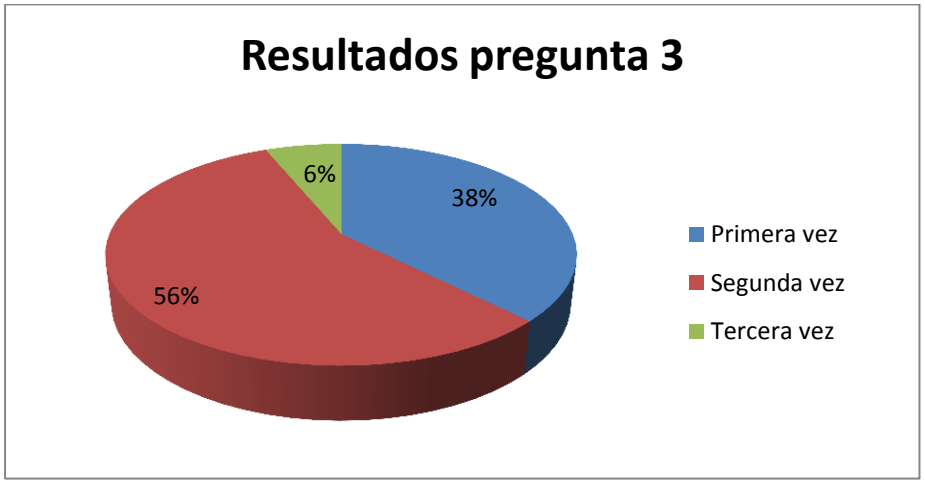
petición de ellos y con el fin de permitir el trabajo colaborativo, y que ellos discutieran sobre la solución que debían dar a los problemas planteados.

6.3 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, de la que se obtuvieron los siguientes resultados:

Se les consultó a los estudiantes, si estaban cursando la asignatura por primera, segunda o tercera, y los resultados fueron los mostrados en la Figura 9, donde se evidencia que el 38% estaba cursando la asignatura por primera vez, el 56% por segunda vez, y 6% por tercera vez.

Figura 10. Resultados Pregunta 3.



Fuente: Autor

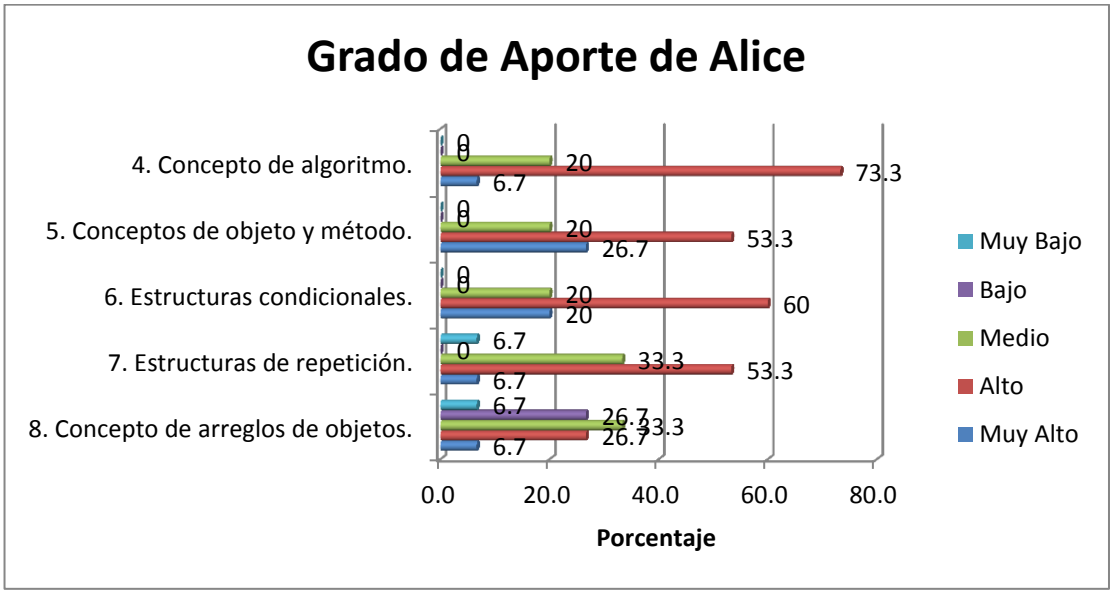
Con el propósito de evaluar el grado de aporte del entorno de programación Alice en el proceso de aprendizaje de los alumnos del grupo experimental se plantearon las preguntas incluidas en la Tabla 25.

Tabla 29. Resultados Preguntas 4-8 de la Encuesta

Actividad	Escala de valoración				
	Muy Bajo	Bajo	Medio	Alto	Muy Alto
4. El concepto de algoritmo.	0%	0%	20%	73.3%	6.7%
5. Los conceptos de objeto y método.	0%	0%	20%	53.3%	26.7%
6. Las estructuras condicionales.	0%	0%	20%	60%	20%
7. Las estructuras de repetición.	6.7%	0%	33.3%	53.3%	6.7%
8. El concepto de vectores de objetos.	6.7%	26.7%	33.3%	26.7%	6.7%

Con base a los resultados presentados en la Figura 9, 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.

Figura 11. Resultados Grado de aporte de Alice



En la Tabla 28, se exponen las preguntas 9 a 12 de la encuesta realizada, orientada a evaluar en qué grado la estrategia de enseñanza implementada, aportó en el proceso de resolución de problemas para desarrollar un programa.

Tabla 30. Resultados Preguntas 9-12 de la Encuesta

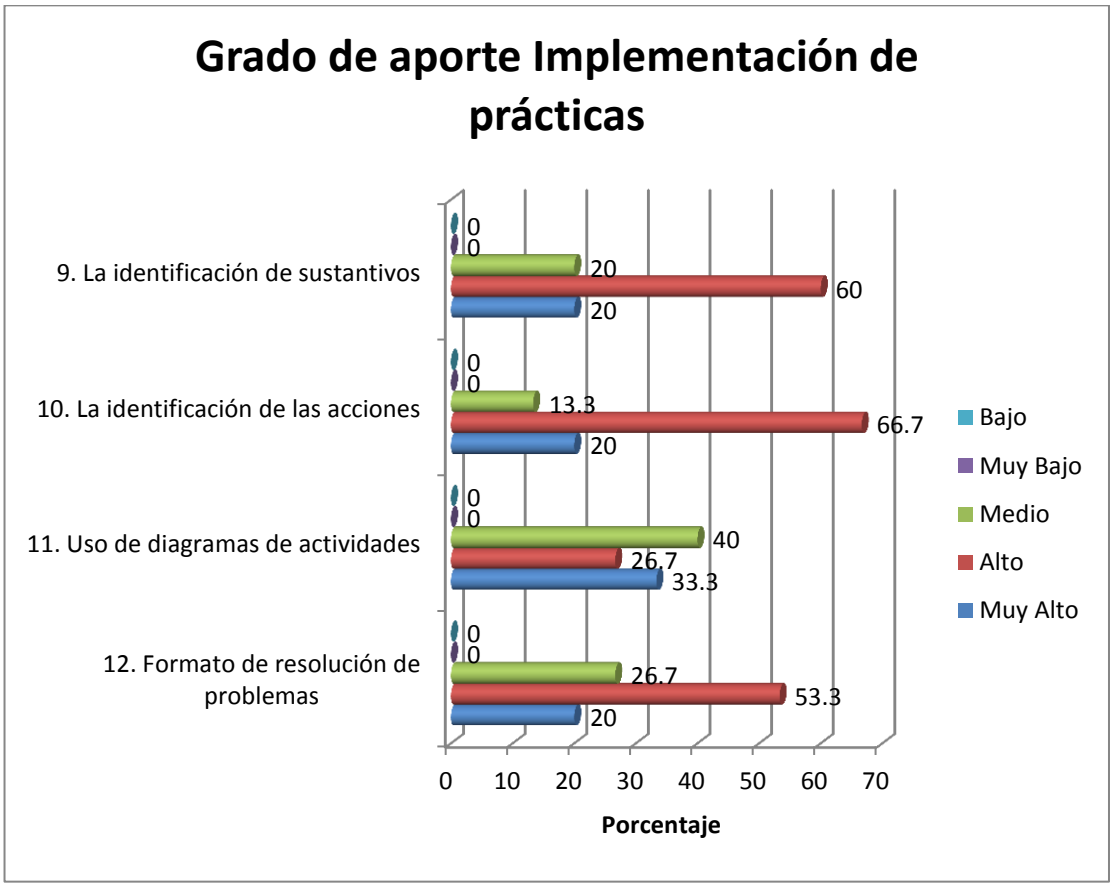
Implementación de las prácticas	Escala de valoración				
	Muy Bajo	Bajo	Medio	Alto	Muy Alto
9. La identificación de los sustantivos en el enunciado de un problema, le aportó en la identificación de las clases que se requieren para plantear una solución.	0	0	20%	60%	20%
10. La identificación de las acciones en el enunciado de un problema, le aportó para identificar los pasos que se debían realizar para plantear una solución.	0	0	13.3%	66.7%	20%
11. El uso de diagramas de actividades, le aportó en el diseño de un algoritmo para dar solución a un problema.	0	0	40%	26.7%	33.3%
12. El formato de resolución de problemas, le sirvió de guía para el desarrollo de un programa.	0	0	26.7%	53.3%	20%

Fuente: Autor

En la Figura 12, 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.

Figura 12. Resultados Grado de aporte de la implementación de las prácticas



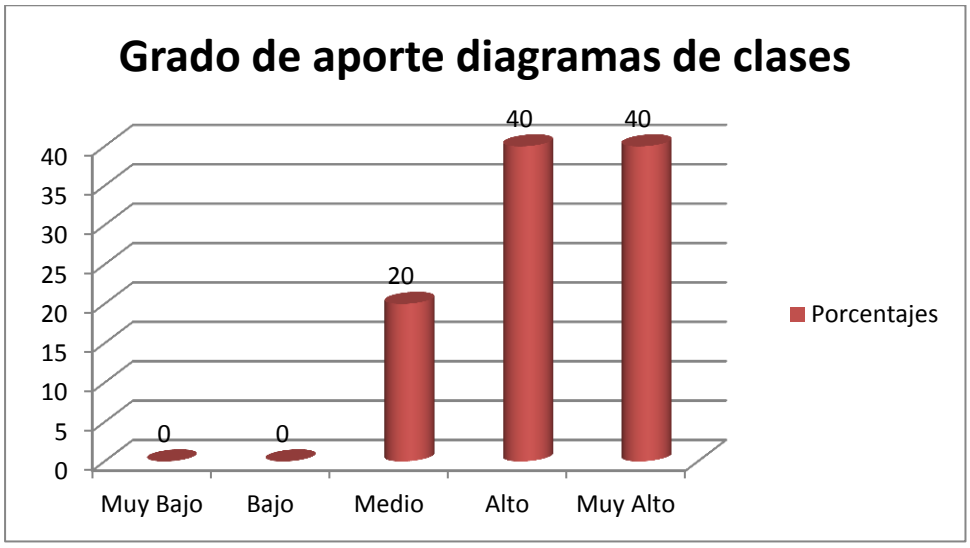
Fuente: Autor

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".

Figura 13. Resultados pregunta 13, grado de aporte de los diagramas de clases



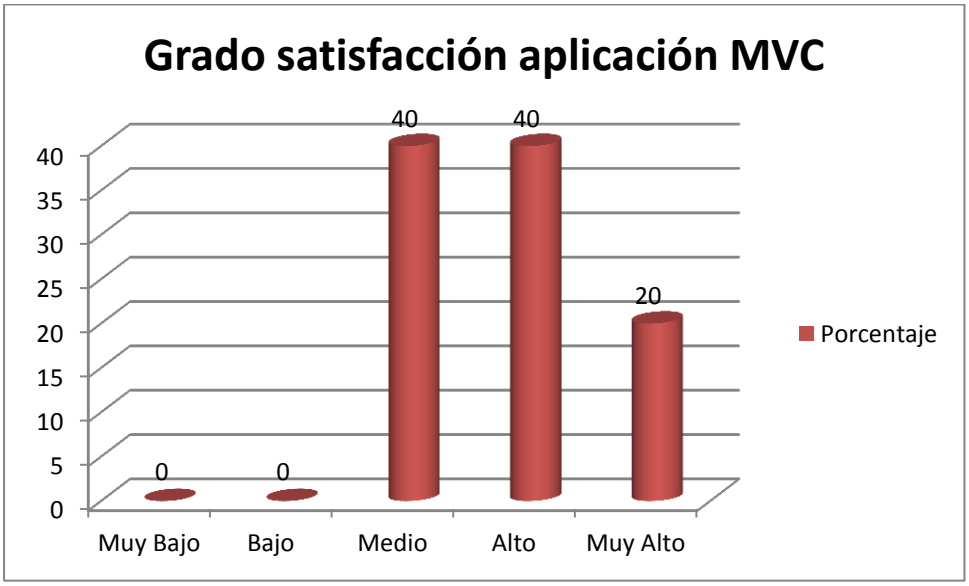
Fuente: Autor

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 14. Resultados pregunta 14, grado de satisfacción aplicando el patrón MVC



Fuente: Autor

La pregunta 15 estuvo enfocada a evaluar el grado de dificultad que se les presentó a los estudiantes en los ítems contemplados en la Tabla 29.

Para evaluar la hipótesis planteada en la presente investigación, se diseñó un cuestionario final, que fue aplicado tanto al grupo experimental AYP-EXP, como a los grupos no intervenidos de la asignatura, los resultados de las notas obtenidas por los estudiantes se exponen a continuación:

Tabla 31. Pregunta 15. Grado de dificultad en proceso de aprendizaje

ITEM	Muy Bajo	Bajo	Medio	Alto	Muy Alto
El entorno de programación Alice	6,7%	46,7%	46,7%	0%	0%
Las estructuras condicionales	20%	40%	33,3%	6,7%	0%
Las estructuras de repetición	20%	26,7%	33,3%	20%	0%
Los arreglos de objetos	20%	20%	46,7%	6,7%	6,7%
El lenguaje de programación Java	13,3%	26,7%	40%	6,7%	13,3%
Aplicación del patrón de arquitectura MVC	6,7%	26,7%	33,3%	26,7%	6,7%

Fuente: Autor

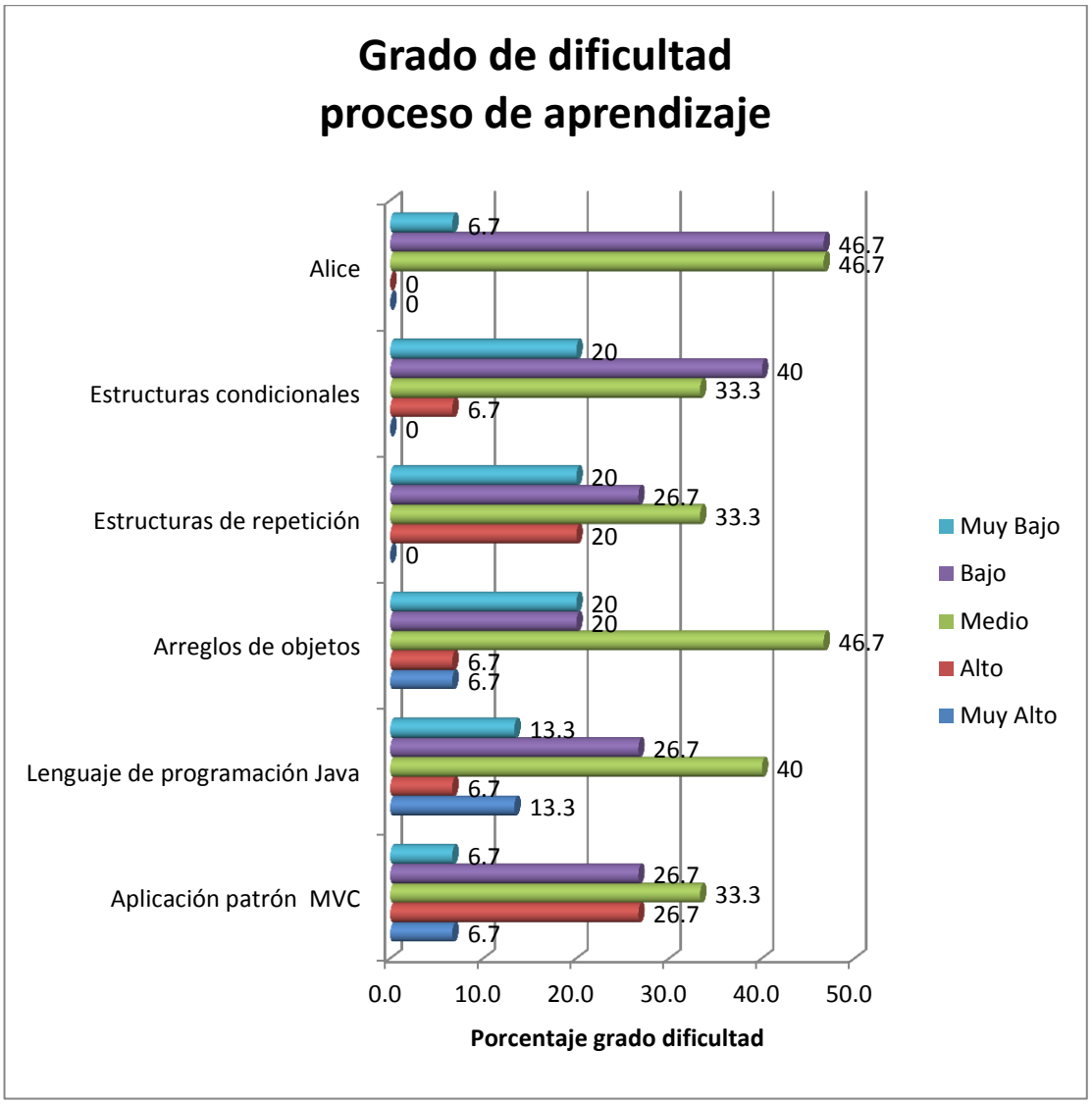
Los resultados de la pregunta 15, muestran que en el aprendizaje del entorno Alice el 46.7% de los estudiantes presentaron un nivel de dificultad bajo y el mismo porcentaje de estudiantes una dificultad media, en el aprendizaje de las estructuras condicionales los estudiantes presentaron solamente el 6,7% de alto nivel de dificultad, en el caso de las estructuras de repetición el 20% de los estudiantes presentaron un alto nivel de dificultad, el cual aumenta el 13.3% con respecto a las estructuras condicionales.

En el aprendizaje de los arreglos de objetos el 46,7% de los estudiantes presentaron un grado de dificultad medio, y 6.7% en alto y muy alto nivel de dificultad, en cuanto al aprendizaje del lenguaje de programación Java el comportamiento de los resultados es muy similar al de arreglos de objetos debido a que el 40% de los estudiantes presentaron un nivel medio de dificultad, y un 13,3% de muy alto grado de dificultad, se puede relacionar que los estudiantes no presentaron altos niveles de dificultad en el aprendizaje del lenguaje, debido a que como manifestaron los estudiantes en las apreciaciones de la pregunta 13, el uso de los diagramas en la etapa de diseño les aportaba una guía visual para iniciar el proceso de codificación.

En la aplicación del patrón de arquitectura MVC, se observa en la Figura 15, que el 26,7% de los alumnos presentó un grado de dificultad alto, que si lo comparamos, con los otros ítems evaluados es superior y esto se puede soportar con lo manifestado por los estudiantes, en cuanto a que los estudiantes que estaban viendo la asignatura por segunda y tercera vez,

habían codificado sus programas en una sola clase, y presentaban inconvenientes al momento de separar las clases en diversas capas, sin embargo comprendían que era una guía muy útil para sistemas más complejos de los realizados en clase, y que tenían que proyectarse a adquirir buenas prácticas en el desarrollo de sistemas de información.

Figura 15. Resultados grado de dificultad en proceso de aprendizaje



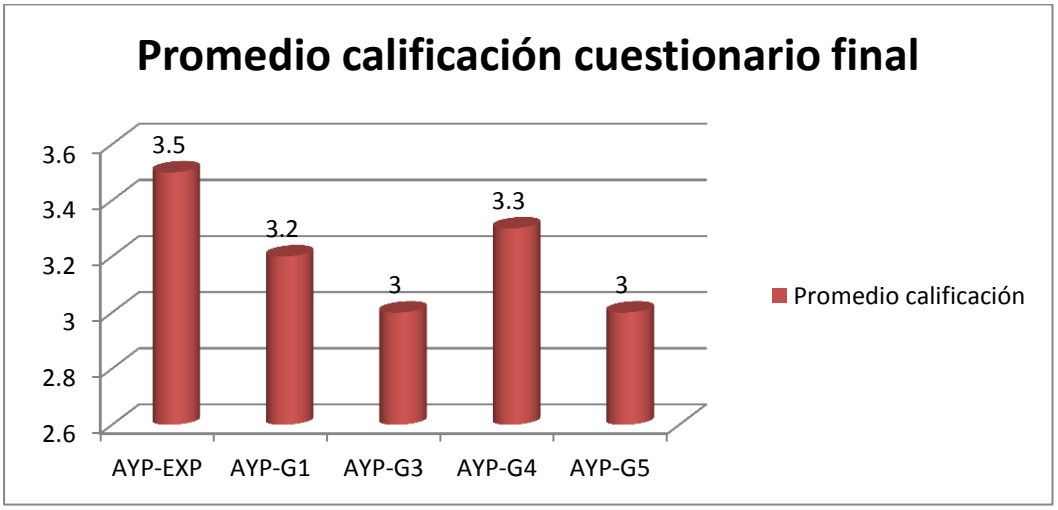
Fuente: Autor

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. Para ver el cuestionario aplicado, ver el Anexo B.

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 aprendizaje de los conceptos básicos de programación y los conceptos fundamentales del paradigma de la POO.

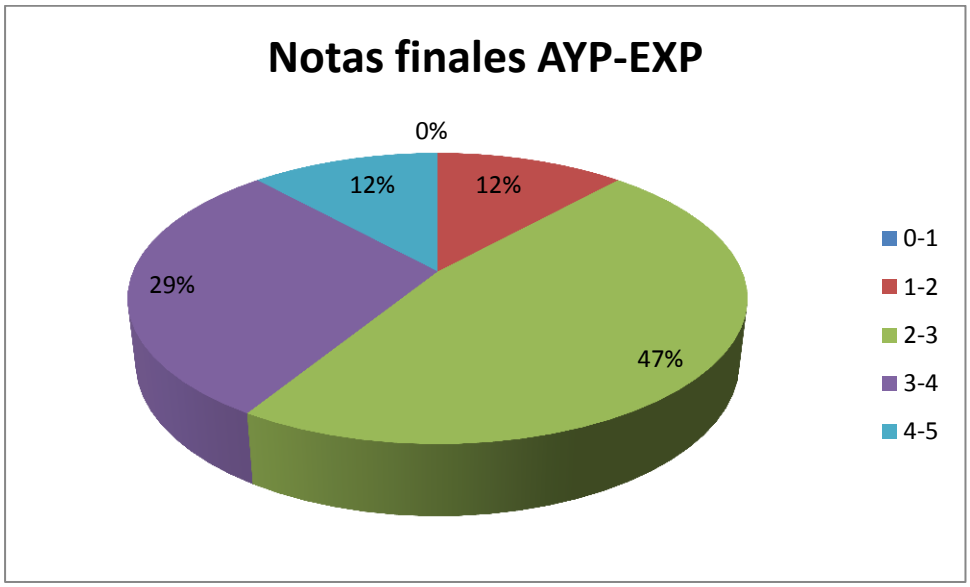
Figura 16. Consolidado resultados cuestionario final



Fuente: Autor

Finalmente, a partir de las notas finales de los estudiantes del grupo experimental, se generó la gráfica expuesta en la Figura 16, 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.

Figura 17. Resultados notas finales grupo AYP-EXP



Fuente: Autor

7 CONCLUSIONES

Cuando se requiere evaluar diversas herramientas que contribuyan a la realización de algún proceso, en este caso el proceso de enseñanza de la programación, es importante que dicha evaluación se haga con la aplicación de alguna metodología, que a partir de una caracterización de las herramientas permita elegir aquella que va a aportar en mayor grado en el proceso que se requiera.

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.

8 REFERENCIAS Y BIBLIOGRAFIA

Allinjawi, A. A., Al Nuaim, H. A., & Krause, P. (2014). Evaluating the Effectiveness of a 3D Visualization Environment While Learning Object Oriented Programming. *Journal of Information Technology and Application in Education*, *3*, 47–56.

Beug, A. (2012). *Teaching Introductory Programming Concepts: A Comparison of Scratch and Arduino* (Tesis de Maestría). California Polytechnic State University, San Luis Obispo, California.

Campbell, D. T., & Stanley, J. C. (1966). *Experimental and quasi-experiment al designs for research*. Rand McNally.

Consejo Nacional de Acreditación. (2013). Lineamientos para la acreditación de programas de pregrado. Retrieved from http://www.cna.gov.co/1741/articles-186359_pregrado_2013.pdf

Cooper, J. W. (2000). *Java Design Patterns: A Tutorial*. Addison Wesley.

Cooper, S., Nam, Y. J., & Si, L. (2012). Initial results of using an intelligent tutoring system with Alice. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 138–143). New York, NY, USA: ACM. http://doi.org/http://doi.acm.org/10.1145/2325296.2325332

Dann, W., Cosgrove, D., Slater, D., & Culyba, D. (2012). Mediated transfer: Alice 3 to Java. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 141–146). New York, NY, USA. http://doi.org/10.1145/2157136.2157180

Dann, W. P., Cooper, S., & Pausch, R. (2011). Programming in Alice: Design and Implementation. In *Learning to Program with Alice* (3rd ed., pp. 20–42). Pearson.

Dann, W. P., Cooper, S., & Pausch, R. (2012). *Learning to Program with Alice*. Prentice Hall.

Eckel, B. (2007). *Piensa en Java*. Prentice-Hall.

Fesakis, G., & Serafeim, K. (2009). Influence of the Familiarization with "Scratch" on Future Teachers' Opinions and Attitudes about Programming and ICT in Education. New York.

Free Software Foundation. (2001). ¿Qué es el software libre? Retrieved from http://www.gnu.org/philosophy/free-sw.html

Jain, V. K. (1986). *Basic Computer Programming*. Pustak Mahal.

Knuth, D. E. (1997). *The Art of Computer Programming: Fundamental Algorithms* (Vol. 1). Addison-Wesley.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A Study of the Difficulties of Novice Programmers. In *ITiCSE '05 Proceedings of the 10th annual*

SIGCSE conference on Innovation and technology in computer science education (Vol. 37, pp. 14–18). New York.

Lange, N. (2011). *Relevance of design patterns within web applications*. Bachelorarbeit.

Leeds, H. D., & Weinberg, G. M. (2009). *Computer programming fundamentals*. New York (Usa): McGraw-Hill.

Lopez, L., Amaro, S., Godoy, I., Alonso de Armino, A. C., Alonso de Armi ˜ no, A. P., & Leiva, M. (2013). Tópicos Avanzados en la Programación de Computadoras.

Losada, I. H. (2012). *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.

Lucio Castillo, M., Ramírez-Gil, M. del P., Garza-Saldaña, J. J., García-Mundo, L. del C., & Vargas-Enríquez, J. A. (2011). “Alice”: un entorno diferente para aprender programación orientada a objetos. *CienciaUAT*, *22*(4), 64–68.

Martin, R. C. (2008). *Clean code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.

Ministerio de Educación Nacional. (2010). Boletín No. 14 Febrero 2010 - Jaque a la deserción. Retrieved from

http://www.mineducacion.gov.co/1621/articles-92779_archivo_pdf_Boletin14.pdf

Monroy, O. L., & Arenas, D. (2014). Enseñando Programación Orientada a Objetos usando un ambiente 3D. In *IX Encuentro Internacional de Investigadores de la RLCU*. Cali-Colombia.

Mosström, J. E. (2011). *A Study of Student Problems in Learning to Program* (Tesis Doctoral). Umea° University, Uema°, Sweden.

Pratt, T. W., & Zelkowitz, M. W. (1998). *Lenguajes de programación : diseño e implementación* (3rd ed.). Mexico: Prentice Hall Hispanoamericana.

Reynolds, J. C. (2009). *Theories of Programming Languages*. Cambridge University Press.

Salim, A., Hassan, S., Hamdi, S., Youssef, S., Adel, H., Khattab, S., & El-Ramly, M. (2010). On using 3D animation for teaching computer programming in Cairo University.

Universidad Pedagógica y Tecnológica de Colombia. (2011). Procedimiento de Autoevaluación de Programas. Retrieved from http://virtual.uptc.edu.co/acreditacion/MODELO/ANEXOS/GUIASYFORMATOS/GUIAS/GUIA_02_PONDERACIONES.pdf

UTTING, I., COOPER, S., KOLLING, M., MALONEY, J., & RESNICK, M. (2010). Alice, Greenfoot, and Scratch – A Discussion. In *ACM Transactions on Computing Education* (Vol. 10). New York, NY, USA.

Van Roy, P., & Haridi, S. (2003). *Concepts, Techniques, and Models of Computer Programming*. The MIT Press.

Werner, L., Campe, S., & Denner, J. (2012). Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*.

Wu, T. C. (2009). *Introducción a la programación orientada a objetos con Java*. Madrid España: McGraw-Hill.