Evaluación de las Emociones de los Estudiantes en el Aprendizaje Visual de la Programación

Evaluating Students' Emotions on Programming Visual Learning

Darwin Alulema[1], Maximiliano Paredes[2]
doalulema@espe.edu.ec, maximiliano.paredes@urjc.es

[1]Universidad de las Fuerzas Armadas ESPE
Sangolquí, Ecuador

[2]Universidad Rey Juan Carlos
Móstoles, Madrid, España

Resumen.- Aprender a programar suele ser una tarea difícil y compleja para los estudiantes, especialmente para los que se inician por primera vez en esta disciplina. Son varios los motivos que pueden originar estas dificultades en el aprendizaje de la programación, pero probablemente uno de los más relevantes tiene que ver con la dificultad que conlleva el manejar las expresiones sintácticas de los lenguajes de programación utilizadas en las tareas de aprendizaje. El estudiante, además de enfrentarse al entendimiento de los conceptos de programación tiene que manejar el lenguaje de programación para expresar los programas que codifican, lo que conlleva una carga cognitiva extra, y que en muchos casos origina una desmotivación del alumno en el aprendizaje. En este artículo se propone una herramienta de programación visual que permite que el alumno pueda codificar programas abstrayendo partes complejas de las estructuras sintácticas del lenguaje, combinando representaciones gráficas sencillas para el alumno con expresiones textuales del lenguaje según el nivel de manejo que tenga el alumno con el mismo. En este artículo se describe una experiencia donde un grupo de estudiantes han utilizado la herramienta y otro grupo ha usado un entorno de desarrollo habitual en el aprendizaje de la programación. Los resultados indican que el número de estudiantes que redujeron su tasa de errores tras la experiencia fue más de un 23,4% en el grupo que usó la herramienta frente al que usó el entorno de desarrollo habitual. Además, se detectó que fue mayor el número de estudiantes que experimentaron emociones positivas en el grupo que usó la herramienta frente al que no lo usó (60% vs. 44,4%).

Palabras clave: Aprendizaje visual, Aprendizaje de la programación, Emociones, Motivación.

Abstract- Learning to program is an often complex and difficult task for students, especially for those who are beginning in this discipline for the first time. There are several causes of such difficulties when learning how to program, but probably one of the most relevant results from the difficulty involved in handling the syntactic expressions of the programming languages utilised in learning tasks. The student, in addition to confronting the understanding of programming concepts, has to manage the programming language in order to express the programs he/she code, which entails an additional cognitive load, and which in many cases leads students to lose motivation for learning. This article proposes a visual programming tool that allows the student to code programs by abstracting complex parts of the syntactic structure of the language, combining simple graphical representations with textual expressions of the language according to the students' level of understanding. This article describes an experiment in which a group of students used the aforementioned tool whilst another group utilised a development environment commonly used when learning programming. The results indicate that the number of students who reduced their error rate after the experiment was more than 23.4% in the group that used the tool compared to the group that used the typical development environment. Furthermore, it was found that the number of students who experienced positive emotions was greater in the group that used the tool in comparison to those that did not (60% vs 44.4%).

Keywords: Visual Learning, Programming Learning, Motions, Motivation.

1. INTRODUCCIÓN

El rápido avance de la tecnología ha cambiado la forma en que los profesores enseñan y los alumnos aprenden. Pese a que estos avances también están presentes en el aprendizaje de la programación (Amer & Ain, 2017), el aprendizaje de ésta no es una tarea fácil. Aprender a programar conlleva alcanzar una serie de logros difíciles, tanto para los alumnos de carreras de informática como para las que no lo son, aunque los primeros, debido al propio ámbito de estudio de la misma, deberán desarrollar más esta capacidad (Forte & Guzdial, 2005). Los trabajos de Aktunc (2013) enumeran numerosos desafíos a los que se enfrentan los instructores y que complican el aprendizaje de la programación en las asignaturas introductorias de las carreras de informática y afines: a) gran variación del perfil de conocimiento de los alumnos; b) desánimo y desmotivación de la mayoría de los estudiantes ya que perciben la programación como una tarea cognitiva difícil y compleja; c) excesivo tiempo destinado a la enseñanza de la sintaxis del lenguaje de programación. Hay que tener en cuenta que pasar demasiado tiempo aprendiendo la sintaxis del lenguaje sin aplicarlo en un contexto de uso es perjudicial para los estudiantes (Mclver & Conway, 1996); y d) la mayoría de los entornos de programación utilizados en aprendizaje son confusos, ya que fueron desarrollados para ingenieros de software profesionales y no tienen un enfoque didáctico. Todos estos factores generan problemas en los cursos introductorios de programación (Kaila et al. 2018).

La forma tradicional de introducir la programación para alumnos que se inician en la misma es a través de un curso introductorio orientado al lenguaje (Mahmudur & Paudel,

DOI: 10.26754/CINAIC.2019.0051

2018). Pero este enfoque orientado al lenguaje produce varios problemas: a) los alumnos tienen dificultades por la complejidad de la sintaxis, b) la sintaxis exige un tiempo extra de aprendizaje y c) el lenguaje en sí mismo no aporta ventajas en el entendimiento de los conceptos de programación que subyacen en sus estructuras, incluso su sintaxis puede dificultar la comprensión de los conceptos. Sin embargo, el uso de un lenguaje de programación es necesario para el aprendizaje y práctica de los conceptos de programación. Es precisamente este el motivo por el que frecuentemente la oferta real de un curso de programación de primer año de carrera dedicará una considerable cantidad de tiempo de clases al aprendizaje de la sintaxis del lenguaje (Hulls et al. 2005). Por tanto, no se debe prescindir del lenguaje en el proceso de aprendizaje y se debería buscar soluciones que gestionen de una manera adecuada su uso como instrumento de aprendizaje en el contexto educativo para mitigar los inconvenientes señalados. Nuestra investigación se centra precisamente en buscar soluciones con esta orientación. Para alentar a los jóvenes en el primer año, la experiencia de aprendizaje debe ser enriquecedora incorporando actividades prácticas, creativas y "divertidas" (Schmidt et al. 2019). Sin embargo, en las asignaturas de introducción a la programación la percepción que tienen los alumnos no es precisamente ésta (Debdi, Paredes-Velasco & Velazquez-Iturbide, 2016). Los alumnos, además de enfrentarse a los retos de aprender a formar soluciones estructuradas para los problemas de programación, deben enfrentarse también a la dificultad de la sintaxis y comandos del lenguaje de programación que usan para plasmar las soluciones de los problemas, cuyos comandos pueden tener aparentemente nombres confusos. En este contexto de dificultad para el estudiante, éste suele percibir que no se genera un contexto de aprendizaje personalmente significativo, experimentando una desmotivación y pudiendo incluso desalentarse ante el aprendizaje (Mahmudur & Paudel, 2018).

El objetivo de nuestra investigación es proponer recursos educativos para el aprendizaje de la programación que abstraigan de manera progresiva de la sintaxis del lenguaje de programación (con técnicas *scaffolding*), motivando al estudiante desde un estado emocional positivo en el aprendizaje. De esta forma, los estudiantes adquirirán un cierto nivel de fluidez en un lenguaje de programación antes de comenzar a implementar sus soluciones en código fuente directamente. Algunas investigaciones proponen que, en lugar de implementar sistemas orientados al aprendizaje completo de un lenguaje de programación, se implementen herramientas u objetos de aprendizaje a pequeña escala para conceptos específicos de programación (Hulls et al. 2005). Estas herramientas deben integrar técnicas *scaffolding* (Willey & Gardner, 2012) que adapten la estructura y contenidos del aprendizaje al estudiante. La investigación en educación y psicología cognitiva sugiere que muchos alumnos hoy en día presentan un perfil de aprendizaje visual e interactivo. Por tanto, parece razonable reducir el detalle textual de la sintaxis del lenguaje de programación que tienen que aprender los alumnos en las primeras etapas, y desarrollar más contenidos visuales sobre los conceptos de programación en sí, intentando facilitar la relación de estos conceptos con el proceso de resolución de problemas.

En este artículo proponemos una herramienta denominada VILEP (*VIsual LEarning Object-oriented Programming*) que está desarrollada sobre esta idea para el aprendizaje de POO (Programación Orientada a Objetos), la cual puede ocultar o hacer visible detalle de la sintaxis del lenguaje que utiliza el alumno combinando representaciones visuales y textuales. De esta forma, combina programación visual con programación textual de código fuente de manera adecuada, orientando así el proceso a un aprendizaje visual. Estudios previos demuestran que el uso del aprendizaje visual tiene un impacto significativo para mejorar la resolución de problemas y habilidades de pensamiento analítico de los alumnos y promueve el aprendizaje activo (Nelson & Crow, 2014).

Ya hay herramientas de programación con enfoque más o menos visual para introducir a los estudiantes en la programación, como Scratch (Basogain-Olabe, Olabe-Basogain & Olabe-Basogain, 2015), Alice (Aktunc 2013), Blockly (Dumitrescu et al. 2009), Greenfoot (Kölling 2010), entre otras. Sin embargo, estas herramientas visuales hacen que los alumnos se dispersen y provoque distracción de su atención (Mahmudur & Paudel, 2018). Además, este tipo de herramientas no establecen técnicas *scaffolding* para el aprendizaje de la programación como la propuesta. No se debe perder de vista que el objetivo de un curso introductorio en programación no debe ser sólo enseñar un lenguaje de programación, sino que debe enseñar además las diferentes formas de resolución de problemas, lógica de razonamiento, diseño básico de algoritmos y conceptos de programación generales, intentando tener poco o mínimo énfasis en la sintaxis del lenguaje (Mahmudu & Paudel, 2018). En este contexto, el uso de herramientas de programación visual puede facilitar el aprendizaje. La herramienta VILEP descrita en este artículo ha sido desarrollada mediante ingeniería dirigida por modelos y facilita un editor gráfico que permite a los alumnos implementar programas en Java mediante recursos visuales ocultando expresiones sintácticas complejas del lenguaje. El objetivo de este artículo es describir dicha herramienta desde un enfoque docente y demostrar la validez de la herramienta en el contexto educativo. Para ellos se ha realizado una experiencia con alumnos de programación de primer año de grado en el aula y se ha medido el conocimiento y las emociones que han experimentado con su uso.

El artículo está estructurado de la siguiente manera: la sección 2 describe la herramienta VILEP y el enfoque docente de su uso en el aula. La sección 3 describe la experiencia realizada con alumnos mientras que la sección 4 muestra las conclusiones del trabajo.

2. Aprendizaje visual de la programación con VILEP

En esta sección se describe la herramienta, la cual es organizada en dos partes: en primer lugar, se describe a groso modo la interfaz de usuario de la misma, y en segundo lugar, se describe su uso desde un enfoque docente.

A. Interacción con VILEP

VILEP permite que el alumno trabaje con conceptos básicos de POO, como son clase, métodos y atributos, y también con otros conceptos de programación en general como son expresiones aritméticas, asignaciones, operaciones de entrada y de salida. Estos elementos de programación son representados mediante componentes visuales disponibles en una paleta de controles, los cuales el alumno los selecciona y arrastra sobre un editor (denominado Lienzo) donde realiza la creación de un programa (ver Figura 1), que en este caso es en lenguaje Java. El editor visualiza la composición del programa en cada momento mostrando algunas partes de la sintaxis de Java y

ocultando otras (las más complejas) mediante iconos visuales. Por ejemplo, la herramienta muestra la sintaxis completa de una instrucción sencilla en Java como es una operación de salida: "`System.out.println("Ingresar A");`" (ver la Marca A de la Figura 1). Sin embargo, una instrucción más compleja como por ejemplo una operación de entrada sobre una variable como la siguiente: "`a=Double.parseDouble(leer.nextLine());`", es representada de forma visual (ver Marca B de la Figura 1), donde un símbolo en forma de lápiz representa que se trata de una operación de escritura del usuario por el teclado y un símbolo de una x entre corchetes expresa que el resultado se asigna a una variable (en este caso una variable denominada "a" de tipo *double*, ver Marca B de la Figura 1). Al no enfocarse la actividad del alumno en la sintaxis del lenguaje, ésta presenta menor carga cognitiva, por lo que el alumno puede centrarse más en los conceptos de programación durante la creación de programas. Sin embargo, la herramienta va eliminando niveles de abstracción y mostrando más detalle de las estructuras del lenguaje a medida que el alumno avanza en el manejo de la sintaxis, trabajando en las últimas etapas prácticamente con el código fuente directamente. De esta forma, VILEP adapta el nivel de *scaffolding* para un entendimiento progresivo del alumno en la creación de un programa para un lenguaje de programación concreto. La herramienta ofrece cuatro funcionalidades principales:

- Agregar al programa declaraciones de clases, métodos y variables. Estos componentes están en la paleta y el alumno los arrastra para componer el programa (Figura 1).
- Realizar escritura y lectura por consola (operaciones de entrada y salida estándar).
- Realizar operaciones matemáticas básicas (suma, resta, multiplicación, división y resto). A estas operaciones se pueden asociar las variables con los valores a operar y la variable en la cual se asigna el resultado de la operación.
- Describir invocaciones de métodos y sus parámetros reales y formales.

Las Tabla 1 muestra los componentes de VILEP y su significado. Teniendo en cuenta el significado de las representaciones visuales de la Tabla 1, podemos interpretar fácilmente el fragmento de programa de la Figura 1: se declara una clase denominada *Suma*, la cual contiene el método *Main*. Dentro de este método se describe la siguiente secuencia de instrucciones: se escribe un mensaje por pantalla ("Ingresar A"), se declara y lee el teclado en la variable "a", posteriormente realiza lo mismo para la variable "b" y a continuación realiza la multiplicación de sus dos contenidos y el resultado lo asigna a la variable "x" declarada como *double,* para por último imprimir el contenido de esta variable por pantalla.

Tabla 1. Componentes y sus representaciones visuales

Componente	Representación visual	Descripción
Class	Class	Declaración de una clase
Method	Method	Declaración de un método
Argument	Argument	Parámetros reales de métodos
Variable	Variable	Declaración de variable
Operador	Operator	Operador aritmético
Message	Message	Escritura de texto por consola
Input		Lectura desde el teclado
Output		Escritura de una variable por consola
Conection		Enlace entre componentes del programa

La herramienta VILEP ha sido desarrollada mediante MDE (*Model Driven Engineering*). Al emplear un modelado MDE el lenguaje de aprendizaje que soporta la herramienta puede ser sustituido por otros lenguajes y paradigmas de programación fácilmente, permitiendo por tanto que la herramienta incluya soporte de varios lenguajes de programación con diferentes sintaxis y estructuras. VILEP permite además generar en cualquier momento el código fuente del programa que ha construido el alumno (el cual incluye comentarios didácticos para facilitar la lectura e interpretación del código fuente), pudiendo así compilar y ejecutar el resultado, viendo el alumno el comportamiento del programa que ha creado.

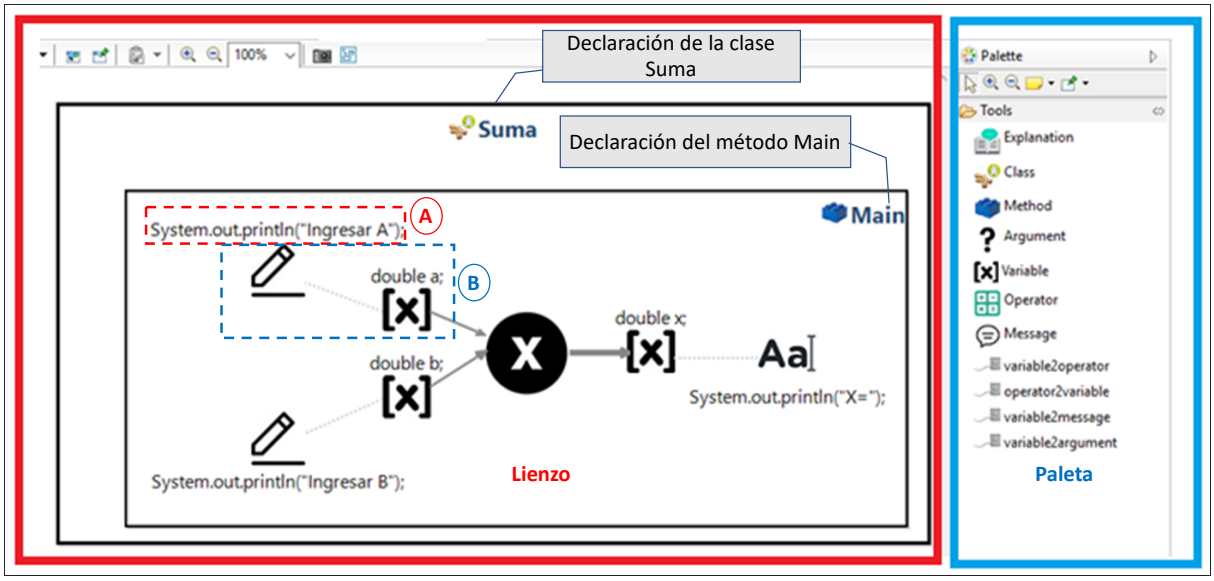


Figura 1. Composición de un programa con VILEP

Como ya se ha indicado, VILEP ha sido desarrollada empleando técnicas de MDE, para lo cual se empleó Eclipse Modeling Framework (EMF) en la creación del metamodelo, la tecnología Sirius para el diseño de la interfaz gráfica y el generador de código fuente Acceleo para la transformación de las instancias específicas del metamodelo a código Java. El resultado es un *plug-in* que el alumno puede instalar fácilmente en el EDI de Eclipse, herramienta habitual en las prácticas de programación de las asignaturas de los grados de informática.

B. Método docente

El objetivo de la herramienta es reducir la carga de trabajo cognitiva de un lenguaje para los alumnos inexpertos en programación. Con este objetivo, las actividades de aprendizaje con la herramienta se desarrollan con tareas cortas de trabajo, de esta forma el alumno va adquiriendo confianza para enfrentarse a problemas de mayor dificultad y enjundia, los cuales requerirán estructuras del lenguaje más complicadas. El enfoque docente se basa principalmente en que los alumnos usarán la herramienta para aprender los principios de la sintaxis básica del lenguaje de programación. Mientras emplean el editor de la herramienta, visualmente van observando cómo se van construyendo algunas líneas de código y al final observarán el código completo. Esto permite al estudiante ver inmediatamente cómo las decisiones que toma en el diseño de los programas se reflejan directamente en una sintaxis específica de un lenguaje de programación. En este contexto, el aprendizaje visual que se genera permitirá que los alumnos poco a poco ganen manejo en la escritura de los programas para un lenguaje concreto.

La metodología docente de uso de la herramienta VILEP se resume a grandes rasgos en los siguientes pasos:

1. El profesor explica los conceptos de programación básicos y presenta brevemente la sintaxis de Java. Además, debe mostrar la representación visual de los componentes de VILEP de estos conceptos.
2. El profesor propondrá un enunciado de un problema a resolver y reflexionará con los alumnos sobre los pasos a dar para resolverlo.
3. A continuación, los alumnos usan VILEP para implementar el programa que resuelve el problema propuesto. Durante este proceso de diseño la herramienta irá mostrando fragmentos de código fuente asociado a determinadas acciones, y ocultando otras (según el nivel *scaffolding* con el que se esté trabajando). Finalmente, la herramienta generará el código fuente que ha implementado ampliado con comentarios aclaratorios.
4. El profesor explicará las dudas y los estudiantes revisarán y podrán ejecutar el programa que han obtenido y verificar la validez de sus soluciones.

3. Experiencia en el aula y resultados

Para demostrar la validez de la herramienta en el proceso de enseñanza se realizó una experiencia con alumnos en el aula. Se detalla a continuación la misma.

A. Objetivo

La experiencia con alumnos tuvo como objetivo validar si la utilización de VILEP en un curso de introducción de la programación mejora los resultados de aprendizaje y el estado emocional del estudiante durante el proceso de aprendizaje.

B. Muestra

La muestra seleccionada fueron alumnos del primer curso del grado de Electrónica y Automatización, de la Universidad de las Fuerzas Armadas ESPE, en Quito Ecuador, en el segundo semestre de 2018, estando constituida por 19 sujetos (17 hombres y 2 mujeres). Esta muestra se organizó de forma aleatoria en dos grupos: grupo experimental (GE) y grupo de control (GC).

C. Variables e instrumentos

La variable independiente fue la herramienta docente aplicada: en el GC se aplicó la herramienta docente clásica de un entorno de desarrollo, mientras que en el GE se usó la herramienta VILEP. Las variables dependientes que se midieron fueron el nivel de conocimiento adquirido y las emociones positivas y negativas experimentadas. Para ello, en los dos grupos, se hizo un pre-test de estas variables al inicio de la experiencia, y un pos-test al finalizar. Los instrumentos para medir estas variables fueron dos escalas. En primer lugar, una escala de conocimiento con 6 ítems multi-opción diseñada específicamente para la experiencia. Esta escala planteaba cuestiones sobre conceptos básicos de POO en las que el alumno tenía que interpretar código fuente en Java. En segundo lugar, se utilizó una escala validada para medir las emociones: PANAS (*Positive Affect & Negative Affect Scale*) de Watson y sus colegas (Watson, D., Clark, L., & Tellegen 1988). El motivo de usar esta escala es que ya está validada en el contexto educativo y permite valorar la emociones positivas y negativas del estudiante en la tarea de aprendizaje. La escala se compone de 20 términos (Tabla 2) que describen emociones de carácter positivo o negativo (10 de ellas positivas y 10 negativas). El sujeto debe valorar cómo se siente para cada uno de estos términos emocionales mediante una escala Likert con 5 opciones de respuesta (nada, muy poco, algo, bastante, mucho).

Tabla 2. Escala de emociones PANAS

Términos de emociones positivas		**Términos de emociones negativas**	
Interesado	Decidido	Disgustado/enfadado	Tenso
Dispuesto	Atento	Culpable	Avergonzado
Animado	Activo	Temeroso	Nervioso
Entusiasmado	Enérgico	Enojado	Intranquilo
Orgulloso	Inspirado	Irritado	Asustado

D. Método

La experiencia comenzó explicando a los alumnos la finalidad de la misma y solicitando el consentimiento de participación (participaron el 100% de los alumnos). A continuación se organizaron los participantes aleatoriamente en dos grupos: a) GE (Grupo Experimental), grupo constituido por 10 participantes que utilizaron la herramienta VILEP, y b) GC (Grupo de Control), constituido por 9 participantes que tuvieron como método docente el habitual utilizando el entorno profesional EDI (Entorno de Desarrollo Integrado) Eclipse.

La Figura 2 muestra el desarrollo de la experiencia realizada. Una vez constituidos los grupos comenzó la intervención realizándose una evaluación inicial del conocimiento y del estado emocional de los alumnos. A continuación, el profesor (fue el mismo en los dos grupos) explicó los fundamentos teóricos de POO (clases, métodos y atributos) y la sintaxis básica de Java (declaraciones de clases y atributos así como operadores aritméticos y entrada y salida).

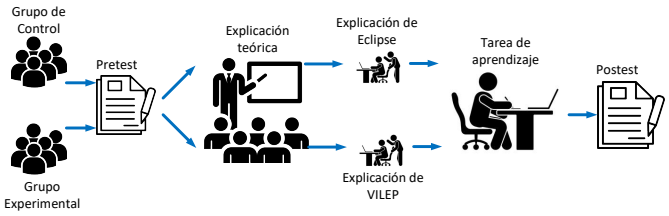
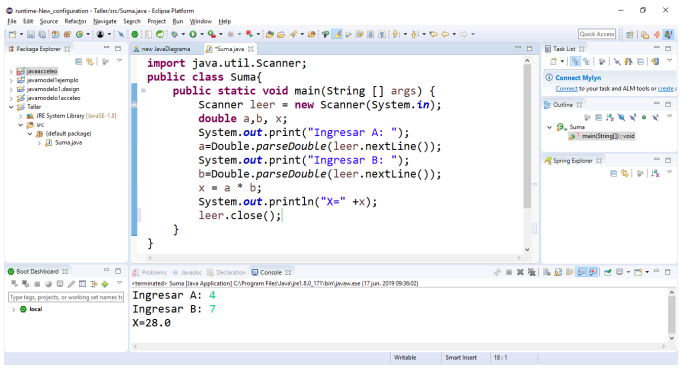


Figura 2. Metodología de la experiencia en el aula

Posteriormente en el GE se explicó el uso de la herramienta VILEP y en el GC se explicó el uso del EDI Eclipse. A continuación ambos grupos hicieron la tarea de implementar un programa en Java muy básico, el GE usando VILEP mediante programación visual y el GC usando Eclipse con programación clásica textual. Este programa básico consistió en declarar una clase con un método que escribía por pantalla el resultado de multiplicar dos números solicitados por teclado. La Figura 3 muestra una captura del programa que desarrollaron estudiantes del GC con Eclipse, mientras que en la figura 1 podemos ver el programa equivalente desarrollado por estudiantes del GE con VILEP. Por último se volvió a evaluar el conocimiento y las emociones tras la realización de la tarea.



```
import java.util.Scanner;
public class Suma{
    public static void main(String [] args) {
        Scanner leer = new Scanner(System.in);
        double a,b, x;
        System.out.print("Ingresar A: ");
        a=Double.parseDouble(leer.nextLine());
        System.out.print("Ingresar B: ");
        b=Double.parseDouble(leer.nextLine());
        x = a * b;
        System.out.println("X=" +x);
        leer.close();
    }
}
```

```
Ingresar A: 4
Ingresar B: 7
X=28.0
```

Figura 3. Tarea de aprendizaje del GC

La Tabla 3 muestra la planificación en tiempos de la experiencia para cada grupo.

Tabla 3. Temporalidad de la experiencia

Fase	Grupo Experimental	Grupo Control
Presentación y formación de grupos	15”	15”
Realización pre-test (conocimiento y emociones)	30”	30”
Explicación conceptos POO y Java	1h	1h
Presentación VILEP	30”	-
Presentación Eclipse	-	30”
Realización tarea de implementación	30”	30”
Realización post-test (conocimiento y emociones)	30”	30”

E. Resultados

En un primer estudio exploratorio de los resultados se ha revisado las tasas de error cometidas en el pre-test y post-test para cada grupo. En el GE al finalizar el 40% de los alumnos mejoró su tasa de error, el 50% la mantuvo y el 10% de los alumnos la aumentó. En el GC el 44.5% de los alumnos cometieron menos errores al finalizar, el 11.1% mantuvo el mismo nivel y el 33.4% aumentó su tasa de error. Como se observa, casi la mitad de los alumnos de los dos grupos mejoraron su tasa de error (44,5% del GC vs. 40% del GE) sin embargo, en el grupo de control existió un incremento significativo de alumnos con más errores frente al experimental (33,4% del GC vs. 10% del GE). La Figura 4 muestra gráficamente estos datos, donde se puede ver cómo los alumnos que usaron VILEP, o bien mejoraron la tasa de error o bien la mantuvieron, y muy pocos la empeoraron, frente a los que usaron Eclipse, que hubo un elevado número de alumnos que cometieron más errores en el post-test. Es decir, a tenor de estos datos parece ser que el uso de la programación visual con la herramienta VILEP hace que los alumnos cometan menos errores en la codificación de programas.

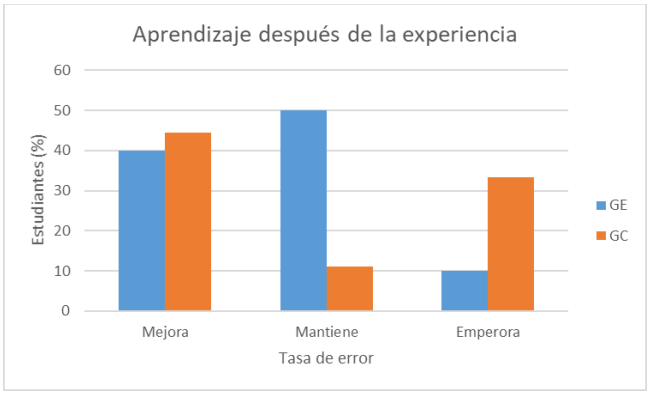


Figura 4. Resultados de aprendizaje según la tasa de errores

En relación con los datos medidos sobre las emociones de los alumnos en la experiencia, los resultados arrojan que en el GE el 60% de los estudiantes aumentaron sus emociones positivas durante la realización de la tarea, mientras que en el grupo de control solo fueron el 44,4% de los alumnos. Por tanto, podemos entender que el uso de la programación visual con VILEP fomenta las emociones positivas de los alumnos en el proceso de aprendizaje de la programación, aunque no hay que descuidar que el uso de la herramienta no disminuyó las emociones negativas tanto como ocurrió en el grupo de control (40% de GE frente 66,6% del GC). Los autores no saben exactamente cuál puede ser el motivo de este hallazgo, aunque creen que podría estar relacionado con el efecto colateral de la carga cognitiva que conlleva el uso de representaciones visuales en el aprendizaje de la programación (Crescenzi et al. 2011).

4. CONCLUSIONES

En las carreras de ingeniería de informática, las asignaturas de introducción a la programación presentan bajos resultados de aprendizaje y bajos porcentajes de aprobados (Lister et al. 2004). Uno de los motivos que genera estos malos resultados es la dificultad que tienen estos alumnos inexpertos en entender y manejar adecuadamente la sintaxis de los lenguajes de programación (Aktunc, 2013). Estas dificultades acaban generando una desmotivación en el estudiante. El alumno no sólo debe enfrentarse al reto de entender conceptos fundamentales de programación, sino que además debe aprender el lenguaje de programación asociado a esos conceptos. En este artículo se presenta la herramienta VILEP, la cual facilita un editor de programación visual que abstrae al alumno de la complejidad del uso del lenguaje y se centra en los conceptos de programación. Esta herramienta oculta las partes más complejas de las estructuras sintácticas de los lenguajes de programación mediante representaciones visuales y las combina con partes textuales de los mismos, de tal forma que a medida que el estudiante va asimilando la sintaxis del lenguaje la herramienta va disminuyendo el nivel de abstracción y mostrando más destalle de las estructuras sintácticas hasta que

el alumno termina desarrollando el código fuente directamente. Se ha validado la herramienta con una experiencia con alumnos de primer curso de ingeniería informática organizándose dos grupos de trabajo para desarrollar un programa en Java: uno de ellos usó la herramienta VILEP propuesta y el otro trabajó con el entorno de desarrollo Eclipse mediante programación textual (herramienta habitual usada en las asignaturas de informática). Se utilizó una escala de conocimiento para medir los resultados de aprendizaje y la escala PANAS para medir las emociones de los alumnos durante el proceso de aprendizaje. Podemos concluir que la herramienta de programación visual propuesta redujo considerablemente el número de alumnos que al terminar la experiencia cometían errores con el lenguaje de programación respecto al grupo que no usó la herramienta (23,4% de alumnos menos). Además, se halló que hubo más estudiantes que experimentaron emociones positivas durante la tarea de programación cuando usaban la herramienta VILEP (60% de los alumnos del grupo) que cuando no la usaban (44,4% del grupo).

A tenor de estos resultados se abre una línea de trabajo futuro novedosa. El análisis de los resultados es un análisis exploratorio por lo que es necesario replicar nuevas experiencias en otras universidades con un análisis estadístico más profundo que permita confirmar si las mejoras en los resultados de aprendizaje y en las emociones halladas son estadísticamente significativas. Además, los autores realizarán estudios que analicen las correlaciones entre los resultados de aprendizaje y las emociones durante el proceso de aprendizaje de la programación.

AGRADECIMIENTOS

Este trabajo ha sido financiado gracias a iProg del MINECO (ref. TIN2015-66731-C2-1-R) y e-Madrid-CM (ref. P2018/TCS-4307) con fondos FSE y FEDER.

REFERENCIAS

Aktunc, Ozgur. 2013. "A Teaching Methodology for Introductory Programming Courses Using Alice." *International Journal of Modern Engineering Research* 3 (1): 350–53.

Amer, Hoda, and Al Ain. 2017. "Smart – Learning Course Transformation for an Introductory Programming Course." *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, 463–65. https://doi.org/10.1109/ICALT.2017.91.

Basogain-Olabe, Xabier, Miguel Ángel Olabe-Basogain, and Juan Carlos Olabe-Basogain. 2015. "Pensamiento Computacional a Través de La Programación: Paradigma de Aprendizaje." *Revista de Educación a Distancia (RED)* 46 (46). https://doi.org/10.6018/red/46/6.

Crescenzi, Pilu, Alessio Malizia, M. Cecilia Verri, Paloma Diaz, and Ignacio Aedo. 2011. "On Two Collateral Effects of Using Algorithm Visualizations." *British Journal of Educational Technology* 42 (6): 145–47.

Debdi, Ouafae, Maximiliano Paredes-Velasco, and J. Angel Velazquez-Iturbide. 2016. "Influence of Pedagogic Approaches and Learning Styles on Motivation and Educational Efficiency of Computer Science Students." *Revista Iberoamericana de Tecnologias Del Aprendizaje* 11 (3): 213–18. https://doi.org/10.1109/RITA.2016.2590638.

Dumitrescu, Crinela, Radu Lucian Olteanu, Laura Monica Gorghiu, and Gabriel Gorghiu. 2009. "Using Virtual Experiments in the Teaching Process" 1 (1): 776–79. https://doi.org/10.1016/j.sbspro.2009.01.138.

Forte, Andrea, and Mark Guzdial. 2005. "Motivation and Nonmajors in Computer Science : Identifying Discrete Audiences for Introductory Courses" 48 (2): 248–53.

Hulls, Carol C W, Adam J Neale, Benyamin N Komalo, Val Petrov, and David J Brush. 2005. "First Programming Course" 48 (4): 719–28.

Kaila, E, M Laakso, T Rajala, A Mäkeläinen, and E Lokkila. 2018. "Technology-Enhanced Programming Courses for Upper Secondary School Students," 683–88. https://doi.org/10.23919/MIPRO.2018.8400128.

Kölling, Michael. 2010. "The Greenfoot Programming Environment." *ACM Transactions on Computing Education* 10 (4): 1–21. https://doi.org/10.1145/1868358.1868361.

Lister, Raymond, William Fone, Robert McCartney, Otto Seppälä, Elizabeth S. Adams, John Hamer, Jan Erik Moström, et al. 2004. "A Multi-National Study of Reading and Tracing Skills in Novice Programmers" 32.

Mahmudur, Rahman, and Roshan Paudel. 2018. "Preliminary Experience and Learning Outcomes by Infusing Interactive and Active Learning to Teach an Introductory Programming Course in Python."

Mclver, Linda, and Damian Conway. 1996. "Seven Deadly Sins of Introducctory Prograimmting Language Design Linda." *Notes and Queries*, 309–16. https://doi.org/10.1093/nq/182.11.148i.

Nelson, Larry P., and Mary L. Crow. 2014. "Do Active-Learning Strategies Improve Students' Critical Thinking?" *Higher Education Studies* 4 (2): 77–90. https://doi.org/10.5539/hes.v4n2p77.

Schmidt, Mirko, Valentin Benzing, Amie Wallman-Jones, Myrto Foteini Mavilidi, David Revalds Lubans, and Fred Paas. 2019. "Embodied Learning in the Classroom: Effects on Primary School Children's Attention and Foreign Language Vocabulary Learning." *Psychology of Sport and Exercise* 43: 45–54. https://doi.org/10.1016/j.psychsport.2018.12.017.

Watson, D., Clark, L., & Tellegen, A. 1988. "Development and Validation of Brief Measures of Positive and Negative Affect: The PANAS Scales." *Journal of Personality and Social Psychology* 54 (6): 1063–70.

Willey, Keith, and Anne Gardner. 2012. "Collaborative Learning Frameworks to Promote a Positive Learning Culture." *Proceedings - Frontiers in Education Conference, FIE*, 1–6. https://doi.org/10.1109/FIE.2012.6462401.