Recibido: 19-10-2017 / Revisado: 24-01-2018
Aceptado: 24-01-2018 / Publicado: 28-03-2018

Catálogo de actividades para desarrollar habilidades algorítmicas para un Sistema Tutor

Catalogue of activities to develop algorithmic skills for a Tutor System

Guillermina Sánchez Román[1], Josefina Guerrero García[1], Daniel Mocencahua Mora[1], Itzel A. Reyes Flores[2]

[1] Benemérita Universidad Autónoma de Puebla, México
[2] Universidad Veracruzana, México

guillesroman@gmail.com , joseguga01@gmail.com , d.mocenca@gmail.com , zs15019630@estudiantes.uv.mx

RESUMEN. Actualmente, se han hecho múltiples esfuerzos de diversos investigadores y asociaciones por estudiar los factores para mejorar la enseñanza-aprendizaje de materias iniciales del área de programación. Sin embargo, existe un déficit en las habilidades algorítmicas de los estudiantes de nuevo ingreso a nivel universitario. Este artículo se centra en el proceso de enseñanza a través del diseño de actividades mediadas por la tecnología, por lo tanto, se presenta una estrategia de diseño instruccional basada en ejercicios por completar que de acuerdo a la literatura son viables para los estudiantes novatos. El proceso se lleva a cabo con base al modelo de Polya para la solución de problemas, posteriormente se realiza el proceso de gamificación de la actividad propuesta para generar la motivación y atraer al estudiante y finalmente se estructuran las actividades con base a la estrategia de ejercicios resueltos en diversos niveles de complejidad. Posteriormente, se define el prototipo de las actividades enfocadas en el desarrollo de las habilidades algorítmicas. Finalmente, la propuesta de diseño del prototipo es evaluada con pruebas de usabilidad aplicado a estudiantes de la Facultad de Ciencias de la Computación. Los resultados arrojaron un nivel de usabilidad satisfactorio, contemplando los comentarios de los evaluadores se propone mejorar las interfaces de cara a una implementación a futuro en un Sistema Tutor.

ABSTRACT. Nowadays, many efforts have been made by various researchers and associations to study the factors to improve teaching and learning of initial subjects in programming area. However, there is a deficit in the algorithmic skills of new entry students at university level. This article focuses on the teaching process through the design of activities mediated by technology, therefore, it presents an instructional design strategy based on exercises to complete that according to the literature are feasible for novice students. The process is based on the model of Polya for the solution of problems, later the process of gamification of the proposed activity is carried out to generate the motivation and attract the student and finally the activities are structured based on the strategy of exercises solved at various levels of complexity. Subsequently, the prototype of the activities focused on the development of algorithmic skills is defined. Finally, the design proposal of the prototype is evaluated with usability tests applied to students of the Faculty of Computer Science. The results showed a satisfactory usability level, considering the comments of the evaluators, it is proposed to improve the interfaces for a future implementation in a Tutor System.

PALABRAS CLAVE: Aprendizaje, Habilidades algorítmicas, Usabilidad, Interface, Gamificación, Diseño instruccional.

KEYWORDS: Learning, Algorithmic skills, Usability, Interface, Gamification, Instructional design.

1. Introducción

Un aspecto notorio en las primeras materias de programación en el área de ciencias de la computación es el reto en los estudiantes al enfrentarse a la resolución de problemas y plantear una solución algorítmica satisfactoria. Se detecta un nivel alto de reprobación (Sánchez, Guerrero, & Mocencahua, 2016) en la materia de Metodología de la programación a través de una análisis del perfil del estudiante. Los datos recolectados en la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla (BUAP), reflejan una realidad similar en cuanto al nivel de reprobación en materias de programación básica, dónde aproximadamente el 50% de los alumnos han reprobado esta asignatura. Normalmente el estudiante se queda con dudas y esto le puede generar un nivel de frustración y probablemente desinterés progresivo. Además, los estudiantes no logran identificar variables importantes del problema y estructuras básicas de selección y de repetición propias de la programación.

Por lo anterior, se tiene la necesidad de plantear nuevas estrategias de enseñanza que permitan al alumno llevar a cabo el proceso de resolución de problemas desarrollando sus habilidades algorítmicas. Por lo tanto, es necesario integrar actividades en que el estudiante practique y observe el avance en sus habilidades. El desarrollo de las actividades que se proponen se basa en el modelo de Polya (1965) integran técnicas de gamificación como estrategia didáctica, se pretende generar un compromiso de acompañamiento y retroalimentación para fomentar el interés y motivación en los estudiantes. Se realizaron pruebas de usabilidad para determinar la facilidad de uso con los prototipos que se tienen.

El presente artículo muestra los elementos propuestos para el diseño de las actividades llevadas a cabo durante la intervención a partir de la dinámica de ejercicios por completar. El objetivo es evaluar los prototipos de acuerdo a su nivel de usabilidad para mejorar las interfaces de usuario que tienen las actividades. Esta parte se integrará a futuro en un Sistema Tutor. La propuesta es evaluada usando tests de usabilidad sobre las actividades para desarrollar habilidades algorítmicas. Los resultados demuestran que la idea puede ser adecuada en el contexto universitario y mejorada de acuerdo a las respuestas obtenidas. Este artículo se estructura de la siguiente forma: en la sección 2 se aborda la conceptualización de las habilidades algorítmicas y el aprendizaje a través de ejemplos resueltos, en la sección 3 se describe el proceso de gamificación de las actividades propuestas, en la sección 4 se despliega los resultados de la evaluación de la usabilidad de las actividades, así como los niveles de evaluación de las habilidades algorítmicas, por último en la sección 5 se describen las conclusiones del trabajo respecto a las mejoras propuestas por los evaluadores de las actividades, así como el trabajo futuro a integrar las actividades en el prototipo del Sistema Tutor.

2. Desarrollando habilidades algorítmicas

2.1. Conceptualizando las habilidades algorítmicas

Existen varias definiciones sobre el pensamiento computacional en la literatura para efectos de este trabajo se tomará la dimensión de resolución de problemas a través del desarrollo de habilidades algorítmicas. El pensamiento algorítmico es un conjunto de habilidades que están conectadas a la construcción y comprensión de algoritmos. Según Futschek (2006), este pensamiento incluye las siguientes capacidades: a) analizar problemas dados; b) especificar un problema de manera precisa; c) encontrar las acciones básicas que son adecuadas para resolver el problema dado; d) construir un algoritmo correcto para resolver un problema determinado, utilizando las acciones básicas; e) pensar en todos los posibles casos tanto especiales como normales de un problema; y, f) mejorar la eficiencia de un algoritmo.

El desarrollo de habilidades algorítmicas se refiere al desarrollo y uso de algoritmos que puedan ayudar a resolver un tipo de problema o tarea. Estas habilidades contemplan las capacidades de abstracción y resolución de problemas en el estudiante. A partir de un conjunto de pasos sistémicos el estudiante pone en práctica habilidades cognitivas como: recordar, comprender, abstraer, analizar, probar y crear, para la solución de algún tipo de problema. Se trabaja con las habilidades algorítmicas que se deben de desarrollar para resolver problemas y crear algoritmos básicos en un lenguaje educativo como es Scratch (Resnick et al., 2009).

Sánchez, G.; Guerrero, J.; Mocencahua, D.; Reyes, I. A. (2018). Catálogo de actividades para desarrollar habilidades algorítmicas para un Sistema Tutor. *Campus Virtuales*, 7(1), 9-17.

Las habilidades algorítmicas están inmersas en las teorías (Sánchez & Guerrero, 2015) que sustentan el modelo conceptual como se puede observaren la figura 1, las habilidades son parte de los dos dominios que caracterizan el perfil del estudiante, además del estilo de aprendizaje. Se integran estas características para definir la estrategia de enseñanza-aprendizaje y se define en el diseño de actividades con elementos de gamificación y se evalúa con base a la taxonomía de Bloom.

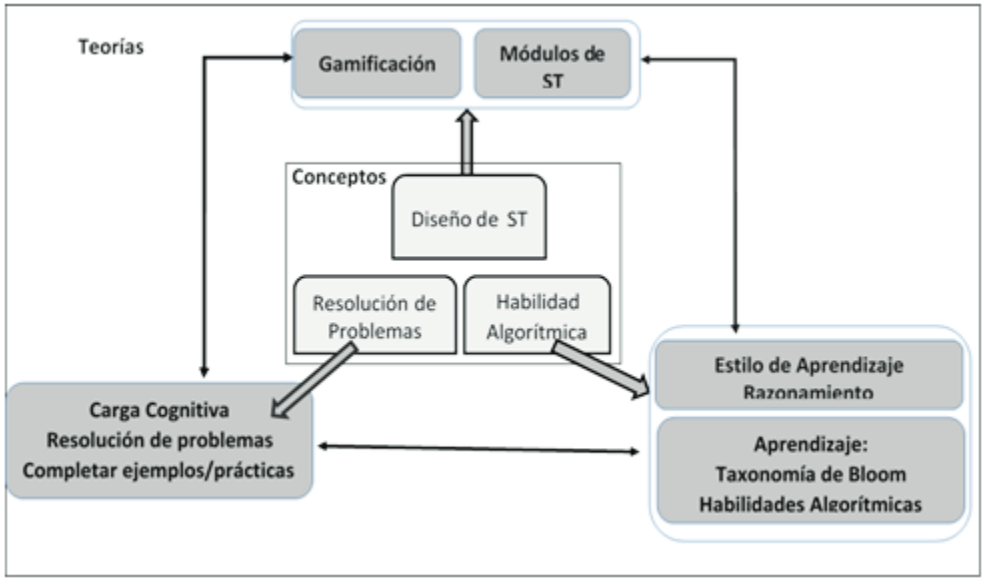

Figura 1. Articulación teórica-conceptual de la metodología. Fuente: Elaboración propia.

Las teorías que soportan la metodología se presentan fuera del cuadro y se interrelacionan con líneas delgadas como lo son: el perfil de estudiante, el aprendizaje de habilidades algorítmicas para caracterizar al estudiante. La teoría de Polya (1965), la teoría de ejemplos por resueltos, prácticas guiadas correspondientes y técnicas de gamificación respaldan la estrategia didáctica creada y se detallan en las subsecuentes secciones, donde se describen las actividades.

2.2. Aprendizaje con base a la estrategia de ejemplos resueltos

Para que el estudiante aprenda es necesario que las actividades a desarrollar no presenten demasiada carga cognitiva, de forma que se comprenda y avance en el proceso de aprendizaje que trascienda como un conocimiento significativo. La teoría de la carga cognitiva (Sweller, 1988; Sweller, 1994) sugiere estrategias instruccionales (llamadas efectos) que toman en cuenta las limitaciones intrínsecas del cerebro humano, en el contexto de esta teoría, se ha documentado que el uso de ejemplos resueltos y de problemas por completar son estrategias instruccionales efectivas para enseñar a estudiantes principiantes (Gerjets, Catrambone & Scheiter, 2004). De acuerdo a la literatura la estrategia de ejemplos resueltos apoyan al aprendizaje de los estudiantes de nuevo ingreso debido a que minimizan la carga cognitiva y van incrementando los pasos sin resolver de forma gradual conforme se va avanzando en su proceso de aprendizaje. La teoría de la codificación dual (Paivio, 1990; Sadoski & Paivio, 2004) señala que la mente humana es capaz de retener y recuperar información y conocimiento si estos se presentan en una modalidad dual, por medio de lenguaje verbal e imágenes. De acuerdo a las teorías planteadas se genera la actividad diseñada bajo un esquema instruccional de ejemplos guiados donde se incrementa el nivel de complejidad en cada uno de ellos.

Para llevar a cabo las actividades se utiliza el lenguaje de programación Scratch, para crear historias interactivas, animaciones, juegos, música, etc., que es desarrollado por el Lifelong Kindergarten en el MIT Media Lab (Resnick et al., 2009). La programación en Scratch se basa en un conjunto de instrucciones icónico/textuales que se combinan o arrastran para crear programas. Los bloques sólo se acoplan si la sintaxis

es correcta, lo que libera al estudiante de la complejidad de la misma y le permite concentrarse en la solución del problema El ambiente en el que se programa no arroja errores, los usuarios no son penalizados, les evita la frustración inicial al no poder ejecutar el programa (errores de compilación) y propicia la experimentación. Como se afirma en (Pozo, 2008) "puede decirse que si los aprendices se entrenan sólo en completar ejercicios (tareas cerradas o rutinarias para las que han aprendido ya una solución específica) difícilmente aprenderán a resolver problemas (tareas más abiertas para las que hay que buscar vías de solución) [...]. Sólo entrenándose en la solución de problemas se aprende a resolver problemas" (p. 168). De tal manera que la estrategia que se propone comienza con ejemplos por completar para estudiantes novatos, pasando por ejercicios sin resolver y finalmente la integración de su solución en código de Scratch.

3. Diseño de la instrucción

3.1. Proceso de gamificación de una actividad

Gamificar es crear un proceso cualquiera, contemplando al jugador como centro del juego. Además ese proceso debe integrar elementos de juego, tales como reglas, interactividad, retroalimentación, etc., e incluir niveles, recompensas, insignias y/o puntos (Gallego, Molina & Faraón, 2014; Kappa, 2012). En la literatura se ha reportado que la gamificación funciona como una estrategia didáctica, en el proceso de enseñanza-aprendizaje, para generar comportamientos específicos en el alumno, dentro de un ambiente que le sea atractivo, que genere un compromiso con la actividad en la que participa y que apoye al logro de experiencias positivas para alcanzar un aprendizaje significativo (Edutrends, 2016).

Algunas técnicas de gamificación son:

Niveles

- Teoría que intenta explicar por qué seguimos haciendo lo que hemos estado haciendo previamente.
- Son necesarios estímulos que "enganchen" y nos hagan ver lo útil que es hacer lo que estamos haciendo.

Desafíos, retos y misiones

- Ofertar una serie de tareas y recompensas específicas.

Sin embargo al gamificar una tarea o actividad, no es necesario que se integren todas las técnicas de gamificación. Existen algunos trabajos que recomiendan y aplican técnicas de gamificación en el ambiente educativo reportando grandes beneficios en el desempeño de las actividades escolares (Orejuela et al., 2013).

A continuación se describe el proceso de gamificación de la actividad propuesta.

Proceso a gamificar: Las actividades de aprendizaje para desarrollar habilidades algorítmicas.

Objetivo: Se explican brevemente algunos conceptos básicos como el estudiante identifica los tipos de estructuras de repetición e iteración. Se pretende que el estudiante identifique, analice y resuelva problemas aplicando las estructuras algorítmicas.

Se inicia el taller mostrando una forma en que el estudiante puede llegar de un punto A a un punto B a través de obstáculos físicos como bancas o compañeros. Un estudiante al azar describe los pasos a realizar y otro estudiante va siguiendo las instrucciones para llegar del punto A (inicial) a un punto B (final).

Se explica la forma de cada bloque y la tarea que realiza cada uno, para que proponga su solución.

El reto es incrementar su puntaje para alcanzar el nivel de experto, de acuerdo a los puntos obtenidos que

se acumularán para su clasificación en un ranking de grupo.

Las normas que se establecen son:

- Leer el problema con detenimiento
- Trabajar de forma individual, generado su posible respuesta en papel a través de pasos determinados en la guía de solución de problemas.
- Ejecutar y plasmar el resultado (prueba de escritorio).
- Utilizar la herramienta de Scratch para plasmar la solución planteada* (opcional).

El sistema de recompensas se da en cada pregunta de acuerdo en el nivel que está dará 5, 10 o 15 puntos. Una vez alcanzado el número de preguntas resueltas de cada categoría se avanza de nivel: novato, aprendiz, experto.

Competición motivante

Se mostrará al estudiante una tabla que indique la posición en que se encuentra con pseudónimo para que se identifique de forma "anónima" y posiblemente mande un mensaje de audio mencionando su avance. Se dará un puntaje extra si pasa del nivel aprendiz a experto.

Niveles de dificultad creciente

Debe haber equilibrio entre dificultad de un reto y la satisfacción de superarlo. El nivel debe ir en aumento, por lo tanto, la clasificación de las preguntas ayudará a mostrarlas en orden de dificultad.

Se pretende desarrollar las habilidades algorítmicas de acuerdo al nivel con el que se haya identificado al inicio del curso, dado que en la Facultad de Ciencias de la Computación existe un proceso de enseñanza-aprendizaje tradicional presencial, donde se integra un enfoque constructivista. Las actividades que se realizan en clase normalmente no se ven recompensadas sólo se considera como participación de clase. A través de esta gamificación de actividades se busca aprender a resolver problemas, generar algoritmos básicos y desarrollar habilidades de pensamiento computacional de forma presencial y autónoma de forma divertida para el alumno.

3.2. Propuesta didáctica de las actividades lúdicas

Futschek (2006) propone un conjunto de pasos para analizar problemas que atiende los cuatro primeros puntos para que los estudiantes desarrollen tanto la habilidad para solucionar problemas, como su pensamiento algorítmico. A partir del enunciado de un problema (en diferentes dominios), se pide al estudiante que identifique cinco elementos: formulación del problema, datos disponibles, restricciones, resultados esperados y procesos necesarios.

Para llevar a cabo las actividades se integraron técnicas básicas de gamificación que se involucran en las actividades de aprendizaje y que respondan los objetivos de aprendizaje y desarrollen procesos cognitivos (Lee & Hammer, 2011; Orejuela et al., 2013), en particular aspectos de retroalimentación en las diversas etapas del ejercicio a resolver como se mencionó en el apartado anterior.

La actividad integra una barra de avance y el puntaje alcanzado en la etapa de la actividad. Por otra parte, la interacción con los elementos visuales para la solución del problema integra los pasos de la solución como parte del apoyo de acuerdo a la teoría de ejemplos por completar, de tal manera que por un lado se tengan los bloques a completar y del otro lado la lista de posibles respuestas. El objetivo será que el alumno identifique el orden y el bloque en el que van, para sólo arrastrar cada enunciado al lugar indicado. El sistema dirá de acuerdo a la respuesta si está correcta o no la solución y si el usuario decide pedir retroalimentación en algún punto de la actividad se revisa la tarea y se propone el contenido temático apropiado de acuerdo a su nivel alcanzado como se ve en la figura 2. Posteriormente el estudiante integra el código elegido en el lenguaje de Scratch e implemente el juego.

En la interfaz, se integran elementos visuales como el puntaje, nivel de avance y retroalimentación. Una forma de captar su atención es que la retroalimentación será visual o auditiva a fin de que entienda los pasos en la resolución del problema.

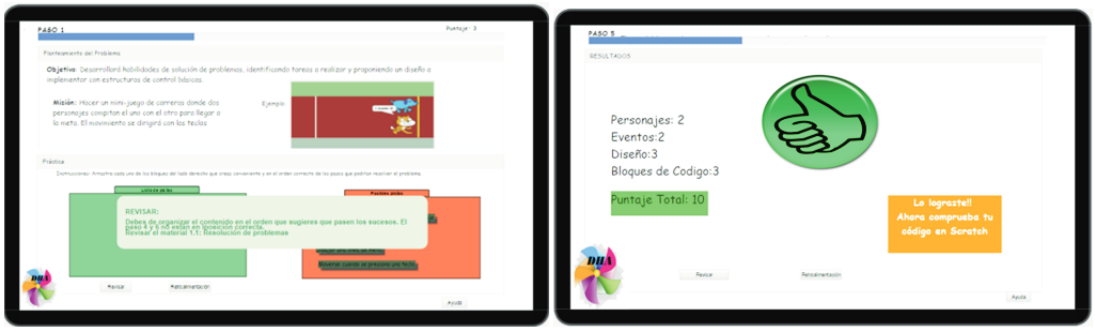

Figura 2. Prototipos de Interfaz de Usuario de la actividad. Fuente: Elaboración propia.

Una afirmación es que "nuestro recuerdo y aprendizaje serán el producto de la interacción entre materiales y los conocimientos previos que activamos" (p. 34). Otra de las condiciones es la creación de actividades que, basadas en conocimientos previos, propicien los nuevos aprendizajes, es decir "para que los [estudiantes] comprendan no basta con presentarles la información […] es preciso diseñar actividades o tareas que hagan más probable esa actividad cognitiva…" (Pérez & Pozo, 2009).

4. Pruebas

El desarrollo de sistemas interactivos implica realizar un diseño pensado en las necesidades reales de los usuarios. El diseño centrado en el usuario es un proceso cíclico en el que las decisiones de diseño están dirigidas por el usuario y los objetivos que pretende satisfacer el producto, y donde la usabilidad del diseño es evaluada de forma iterativa y mejorada incrementalmente.

La Usabilidad es "la medida en que un sistema, producto o servicio puede ser usado por usuarios específicos para alcanzar metas con efectividad, eficiencia y satisfacción en un contexto de uso específico" (Bevan, 2008; ISO, 2009). Las pruebas de usabilidad de un sistema de software se basan en las mediciones de la experiencia de los usuarios al interactuar con éste (Dix, 1993). La experiencia de usuario se refiere a "las percepciones y o respuestas que resultan del uso anticipado de un producto, sistema o servicio"(Bevan, 2008; ISO, 2009).

4.1. Evaluación de usabilidad

En el ciclo de vida de software, dentro del Diseño Centrado en el Usuario (DCU), una de las características es la evaluación de la usabilidad. Ésta puede realizarse desde etapas tempranas del desarrollo, ya que la única manera de estar seguros acerca de algunas características del diseño potencial es construir prototipos y probarlos con usuarios reales. Así, el diseño puede modificarse para corregir los errores encontrados en las pruebas e ir mejorando gradualmente el producto final (Dix, 1993).

Para la evaluación de los prototipos propuestos en este artículo se llevaron a cabo pruebas de usabilidad con 4 usuarios, todos alumnos del octavo semestre de la Ingeniería en Tecnologías de la Información de la Benemérita Universidad Autónoma de Puebla. Las pruebas fueron en un ambiente controlado en el que se les pidió a los usuarios simular la interacción con los prototipos realizando la actividad de diseño de algoritmo, que consiste en leer las instrucciones y completar la secuencia de pasos a resolver en el primer paso de la actividad debe arrastrar los enunciados de la derecha tratando de organizar la lista de pasos generales para resolver el problema. En el segundo paso se presentan diversas figuras que representan los personajes que deben de elegir, en cualquier momento se puede pedir retroalimentación de la actividad o ayuda sobre el uso de la aplicación. En el tercer paso se procede a identificar los eventos que manipularán el personaje y en el último

paso se le presenta al usuario los bloques que manipulan al personaje y él debe elegir el bloque correcto, al final se muestra el resultado de su desempeño durante la actividad (puntos, nivel de avance y categoría).

Las actividades aquí descritas serán integradas posteriormente a un Sistema Tutor, por lo tanto es importante evaluar los prototipos de acuerdo al uso y entendimiento que el usuario tiene al interactuar con el prototipo. A fin de garantizar la facilidad de uso, la calidad de la información y de la interfaz se solicitó a los usuarios contestar el Cuestionario de Usabilidad en Sistemas Informáticos (CSUQ, por sus siglas en inglés) adaptado y validado al español (Hedlefs et al., 2016), del cual se obtuvieron los resultados presentados en la tabla 1.

Resultados de la aplicación del test (CSUQ)Rango	Uso del Sistema	Calidad de la Información	Calidad de la Interfaz	Estimado General
Alta	6.450	6.898	7.294	6.707
Baja	4.382	4.351	4.038	4.792
Promedio	5.416	5.625	5.666	5.75

Tabla 1. Resultados de aplicación CSUQ. Fuente: Elaboración propia.

Como se puede observar en la tabla 1 todas las categorías (Uso del sistema, Calidad de la Información, Calidad de la IU, General) el valor mínimo no está por debajo de 4 y la media obtenida se encuentra entre 5 y 6, por lo que se considera un nivel de satisfacción buena pero que puede mejorar (Hedlefs et al., 2016). Durante la aplicación se valoraron aspectos que no se miden en el cuestionario como la forma de resolver el problema y como se veían interesados en resolver el ejercicio correctamente.

De acuerdo a los comentarios de los usuarios, se propone:

Usuario 1: Creo que se puede mejorar los botones en la parte inferior y usar palabras más fáciles de comprensión. Se podrían usar iconos para una mejor comprensión. Al arrastrar las instrucciones al otro cuadro, estaría bien poder enumerar las opciones."

Usuario 2: La interfaz es buena y muy amigable pero le falta que proporcione información más precisa de qué es lo que se debe realizar sería muy bueno resaltar los botones"

Usuario 3: En general el sistema me pareció muy bien pero creo que me gustaría que me llevara más de la mano en cuanto a la explicación de cada botón. En la parte de las respuestas donde aparece la parte del Scratch me diera la oportunidad de explicarme cada parte así poder dar una respuesta más adecuada".

Usuario 4: "A mi parecer la interfaz de retroalimentación para el usuario deberían de retirarse los botones de revisar y ayuda, considero se serían iconos visuales"

4.2. Evaluación de las habilidades algorítmicas

Las actividades propuestas al estudiante tienen el siguiente formato aplicado en la intervención. Se da la descripción del objetivo y una imagen donde se muestra como debe finalizar el juego creado. El lenguaje propuesto es Scratch además de una hoja que tiene los apartados a completar en los ejercicios dos y tres. Sólo el primer ejercicio se encuentra desglosado en los apartados de animar, eventos y lista de pistas. Cada uno de estos apartados se muestra y se llevan a cabo solo en el primer ejercicio para que el estudiante sepa cómo desarrollar su primer juego. Posteriormente, se da el siguiente ejercicio y se va completando con la solución propuesta por el estudiante.

Para evaluar las habilidades de los estudiantes se toma la adaptación al test de Román (Román-González, Pérez-González & Jiménez-Fernández, 2015) donde se clasifica las actividades de acuerdo a la taxonomía de Bloom y se determinan niveles de novato, aprendiz y experto (ver tabla 2).

Taxonomía Bloom	Habilidades	Nivel
Recordar	Reconocer la sintaxis de una instrucción.	Novato
	Reconocer la semántica de una instrucción.	
Comprender	Interpretar que cambios ocurren al modificar una instrucción.	
	Traducir un algoritmo dado a pseudocódigo o diagrama de flujo.	
Aplicar	Utilizar estructuras con un fin determinado y claro (decisiones, ciclos).	Aprendiz
	Modificar un algoritmo.	
Analizar	Predecir los posibles efectos al ejecutar un algoritmo línea por línea.	
	Detectar posibles errores en un algoritmo.	
Evaluar	Evaluar requerimientos a nivel de usuario, sistema, organización y desarrollar una solución algorítmica coherente que los tome en cuenta.	Experto
	Comparar soluciones algorítmicas que resuelven un mismo problema.	
	Verificar un conjunto de instrucciones a través de pruebas de escritorio)	
Crear	Generar un algoritmo completo a partir de las especificaciones de un problema	
	Generalizar un algoritmo.	

Tabla 2. Habilidades de acuerdo a la taxonomía de Bloom. Fuente: (Sánchez & Guerrero, 2016).

5. Conclusiones

Debido a que los estudiantes han mostrado la necesidad del acompañamiento en la resolución de problemas computacionales, es favorable continuar en el proceso de diseño del Sistema Tutor a fin de integrar los resultados encontrados en durante las intervenciones realizadas. En la aplicación de la estrategia de aprendizaje fue relevante identificar que aún un alto porcentaje de estudiante no ha alcanzado el nivel experto, y que denota que es importante agregar posiblemente más ejercicios.

De acuerdo a los resultados obtenidos en la evaluación de los prototipos, las interfaces se modificarán para dar un mejor aspecto visual y sobre todo una mejor comprensión de la realización de la actividad. Las observaciones hechas de forma textual y verbal aportan ideas a integrar con elementos que les llaman la atención como estudiantes jóvenes. Se pretende generar una primera versión de la interfaz en los próximos meses para revisar los aportes dados en ésta intervención.

Como trabajo futuro se llevará a cabo la implementación de las interfaces de usuario, haciendo una segunda evaluación de las interfaces. Dado que el proyecto completo es realizar un Sistema Tutor Inteligente que permita desarrollar las habilidades algorítmicas en los estudiantes de nuevo ingreso, posteriormente se integrarán las interfaces modificadas en el prototipo a generar.

Cómo citar este artículo / How to cite this paper

Sánchez, G.; Guerrero, J.; Mocencahua, D.; Reyes, I. A. (2018). Catálogo de actividades para desarrollar habilidades algorítmicas para un Sistema Tutor. *Campus Virtuales*, 7(1), 9-17. (www.revistacampusvirtuales.es)

Referencias

Bevan, N. (2008). UX: Usability and ISO Standards. W39RG.

Dix, A. (1993). Human-computer interaction. Prentice, Ed..

Edutrends (2016). Edutrends Gamificación.

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. Informatics Education – The Bridge between Using and Understanding Computers (pp. 159-168).

Gallego, F.; Molina, R.; Faraón, L. (2014). Gamificar una propuesta docente. Diseñando experiencias positivas de aprendizaje. España: Oviedo, Ed.

Gerjets, P.; Catrambone, R.; Scheiter, K. (2004). Using Visualizations to Teach Problem-Solving Skills in Mathematics: Which Kind of Visualization Works.

Hedlefs, H.; de la Garza, A.; Sánchez, M.; Garza, V. (2016). Adaptación al español del Cuestionario de Usabilidad de Sistemas Informáticos CSUQ / Spanish language adaptation of the Computer Systems Usability Questionnaire CSUQ. RECI Revista Iberoamericana de las Ciencias Computacionales e Informática, 4(8), 84-99.

ISO (2009). ISOFDIS9241-210.

Kappa, K. (2012). The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. San Francisco: J. W. y Sons, Ed.

Lee, J.; Hammer, J. (2011). Gamification in Education: What, How, Why Bother?. Academic Exchange Quarterly, 15, 1-5.

Orejuela, H. F.; García, A. A.; Hurtado, J. A.; Collazos, C. (2013). Analizando y Aplicando la Gamificación en el Proceso ChildProgramming (pp. 7-23).

Paivio, A. (1990). Dual Coding Theory: Retrospect and Current Status. Canadian Journal of Experimental Psychology/Revue Canadienne de Psychologie Expérimentale, 45(3), 32. DOI: https://doi.org/10.1037/h0084295

Pérez, J.; Pozo, E. (2009). Aprender para comprender y resolver problemas. Madrid: E. Morat, Ed.

Polya, G. (1965). Como plantear y resolver problemas. Trillas, Ed.

Pozo, J. (2008). Los rasgos de un buen aprendizaje, en Aprendices y maestros: la psicología cognitiva del aprendizaje. Alianza, Ed.

Resnick, M.; Silverman, B.; Kafai, Y.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; … Silver, J. (2009). Scratch. Communications of the ACM, 52(11), 159-168.

Román-González, M.; Pérez-González, J. C.; Jiménez-Fernández, C. (2015). Test de Pensamiento Computacional: diseño y psicometría general [Computational Thinking Test: design & general psychometry]. DOI: https://doi.org/10.13140/RG.2.1.3056.5521

Sadoski, M.; Paivio, A. (2004). A Dual Coding Theoretical Model of Reading. (I. R. Association, Ed.) (5o ed.). In Theoretical models and processes of reading.

Sánchez, G.; Guerrero, J. (2015). Diseño de un Sistema Tutor Inteligente para mejorar las habilidades en la resolución de problemas. In XIII Congreso Nacional de Innovación Educativa.

Sánchez, G.; Guerrero, J. (2016). Sistema Tutor Inteligente para el desarrollo de habilidades algorítmicas. In 11 Congreso Colombiano de Computación.

Sánchez, R. G.; Guerrero, G. J.; Mocencahua, M. D. (2016). CcITA-2016_GSR3. In M. E. Prieto & S. J. Pech (Eds.), Análisis del perfil de estudiantes de Computación para un Sistema Tutor (pp. 493-498). México.

Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. Cognitive Science, 12(2), 257-285.

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. Learning and Instruction, 4(4), 295-312.