ISSN: 1692-7257 - Volumen 1 – Número 39 - 2022

Revista Colombiana de Tecnologías de Avanzada

Recibido: 15 julio de 2021
Aceptado:3 octubre 2021

MODELO DE DISEÑO INSTRUCCIONAL ORIENTADO AL FORTALECIMIENTO DEL PENSAMIENTO ALGORÍTMICO EN LOS ESTUDIANTES DE CURSOS INICIALES DE PROGRAMACIÓN DE LA UNIVERSIDAD DE PAMPLONA

INSTRUCTIONAL DESIGN MODEL ORIENTED TO THE STRENGTHENING OF ALGORITHMIC THINKING IN STUDENTS OF INITIAL PROGRAMMING COURSES AT THE UNIVERSITY OF PAMPLONA

Msc. Durán-Villamizar Jesús Enrique[*], PhD. Yovanni Alexander Ruiz Morales[**], PhD. Diana Ruth Martínez Suárez*

* Universidad de Pamplona, Facultad de Ingeniería y Arquitectura, Ingeniería de Sistemas, Grupo de Investigación GIIDAC. Autopista Internacional Vía Los Álamos Villa Antigua Villa del Rosario - Norte de Santander.
E-mail: jesusenrique.duran@unipamplona.edu.co
E-mail: dmarsua@unipamplona.edu.co

** Universidad de Pamplona, Facultad de Ciencias de la Educación, Departamento de Pedagogia, Grupo de Investigación Pedagógica.
Km. 1 vía a Bucaramanga, Pamplona, Norte de Santander, Colombia
E-mail: yovanni.ruiz@unipamplona.edu.co

Resumen: La adquisición de habilidades para "el buen pensar" es el objetivo principal en cualquier ámbito de formación; tanto en los escenarios de alto rigor académico como en los escenarios orientados a un entrenamiento técnico, de tal manera que la puesta a prueba de los conocimientos adquiridos y las habilidades y entrenamientos fomentados durante un proceso de formación, permitan abordar de manera eficiente los retos que implican el diseño y desarrollo de las soluciones a los problemas que tengan lugar en la vida profesional. Es así, como la adquisición de conocimientos y el fortalecimiento de habilidades, deben estar especialmente orientados a la construcción del pensamiento en el área específica; esto es, un pensamiento matemático en el matemático, un pensamiento físico en el físico, y en el tema especial que nos ocupa de, un pensamiento computacional y/o algorítmico en los estudiantes que están interesados en formarse como Ingenieros de Sistemas, con un fuerte y riguroso proceso de formación en el área de la programación y el desarrollo de software. En la construcción de la presente propuesta, se sigue una rigurosa línea de exploración académica, desde la fundamentación filosófica de textos como ¿Que significa pensar? de M. Heidegger entre otros, hasta la impronta de las metodologías de Diseño Instruccional, un valioso recurso para la generación de guías descriptivas en la construcción de herramientas de formación y apoyo, operando bajo la estructura de fases claramente establecidas (Analysis, Design, Develop, Implement, Evaluate). De igual importancia son las contribuciones en el área de los aprendizajes dentro contexto educativo en Colombia, de los trabajos de Julián de Zubiría Samper, en el marco de las llamadas "Pedagogías Activas".

Palabras clave: pensamiento computacional, pensamiento algorítmico, diseño instruccional, lenguajes de programación, algoritmo, formación de pensamiento, pedagogía.

Abstract: The acquisition of skills for "good thinking" is the main objective in any training field; both in scenarios of high academic rigor and in scenarios oriented to technical training, in such a way that the testing of the knowledge acquired and the skills and training promoted during a training process, allow to efficiently address the challenges that involve the design and development. solutions to problems that arise in professional life. Thus, just like the acquisition of knowledge and the strengthening of skills, they must be especially oriented to the construction of thought in the specific area; that is, a mathematical thought in the mathematician, a physical thought in the physicist, and in the special topic that concerns us, a computational and/or algorithmic thought in the students who are interested in training as Systems Engineers, with a strong and rigorous training process in the area of programming and software development. In the construction of this proposal, a rigorous line of academic exploration is followed, from the philosophical foundation of texts such as What does it mean to think? of M. Heidegger among others, to the imprint of Instructional Design methodologies, a valuable resource for the generation of descriptive guides in the construction of training and support tools, operating under the structure of clearly established phases (Analysis, Design, Development, Implementation, Evaluate). Of equal importance are the contributions in the area of learning within the educational context in Colombia, based on the work of Julián de Zubiría Samper, within the framework of the so-called "Active Pedagogies".

Keywords: computational thinking, algorithmic thinking, instructional design, programming languages, algorithm, thought formation, pedagogy.

1. INTRODUCCIÓN

Promover y fortalecer las habilidades en la programación de computadoras, es una actividad fundamental en todo el proceso de formación de los Ingenieros de Sistemas. Desde los primeros años, es una tarea que tiene todo un conjunto de implicaciones y retos. Iniciar en la programación de computadores, es mucho más que la enseñanza de un paradigma o lenguaje determinado. Actividades mentales como la abstracción, análisis y síntesis de situaciones, ni la lógica de pensamiento, son lamentablemente presupuestos en los estudiantes, pues de manera casi-generalizada vienen de una educación caracterizada por prácticas de enseñanza que no brindan las mejores experiencias ni resultados en el desarrollo de habilidades y conocimientos, muy a pesar de los copiosos discursos sobre competencias de la educación básica y media, continuando sesgados al lastre de una educación conductista y normativa, que no procura el desarrollo adecuado de las estructuras de pensamiento; una escuela que sigue en su rol de "enseñante" pero con ánimo pertinaz y anquilosado.

De otro lado, en ningún área de la ingeniería es tan demandante, como en el caso de los ingenieros de sistemas, el condicionamiento a las necesidades de un mercado local, regional, nacional e incluso internacional, que conmina a este perfil profesional a responder requerimientos de conocimientos y experiencias (demostrables y en rangos mínimos de 3 años, como es el común denominador de las ofertas laborales) en "plataformas y lenguajes específicos" (que generalmente, son tendencia o moda).

En adición a lo anterior, aún se sigue dando el debate sobre la conveniencia de aplicar un enfoque estructurado, orientado a objetos o funcional para iniciar los fundamentos de la programación en Ingeniería de Sistemas. En el presente trabajo, se han tomado en consideración valiosas experiencias, como el proyecto Cupi2, de la Universidad de los Andes, como un importante modelo pedagógico; así como, la llevada a cabo en las IES de San Juan de Pasto, en la aplicación y evaluación del modelo funcional con el lenguaje Scheme y fomentar en el estudiante la comprensión y uso de la recursividad como principal apoyo en la solución de problemas de desarrollo de software (Timarán, 2009).

En la Universidad de Pamplona, Colombia, ha constituido una nueva propuesta curricular para el programa de Ingeniería de Sistemas con un mayor interés en el fomento de las habilidades de pensamiento, pero es evidente que los eventos de la práctica pedagógica deben ser pertinentes a los discursos inmersos en los documentos del PEP (Proyecto Educativo del Programa) y PEI (Proyecto

Educativo Institucional). En este sentido, la presente propuesta, pretende indagar en los muchos aspectos, que permitan exponer y clarificar sobre la validez de los caminos establecidos y el cúmulo de experiencias de los actores del proceso, como son estudiantes, docentes y directivos académicos.

En torno a las dificultades para la comprensión de los problemas, se ha seguido el horizonte brindado dentro del grupo de las llamadas "Pedagogías activas" y con un ejercicio de formación académica minucioso, riguroso, con la adecuada identificación de los conceptos fundantes, pilar fundamental de los modelos pedagógicos tan brillantemente planteados y desarrollados por Julián de Zubiría Samper (1999).

2. ENSEÑANZA DE LOS FUNDAMENTOS DE PROGRAMACIÓN EN INGENERÍA DE SISTEMAS

El objetivo de un primer curso de programación es proporcionar a los estudiantes, sin ninguna experiencia previa en programación, los mecanismos necesarios para solucionar problemas en el marco de expresión de un paradigma de programación para abordar los problemas y con un ejercicio riguroso para ir desde la especificación al programa correcto (Timarán, 2009).

Villalobos (2009), afirma que "El objetivo de los cursos de programación no es únicamente que el estudiante aprenda a escribir un programa de computador" (p. 2).

En la búsqueda de propuestas actuales frente al tema de innovar la enseñanza de la programación muchos autores concuerdan con la necesidad de integrar la generación de habilidades más que la transmisión de conocimientos planos (López, 2008).

Entre muchas de las miradas, posturas y discursos alrededor del tema de la enseñanza de la programación, continua presente la preocupación por parte de las comunidades académicas dado que, criterios como las tasas de mortalidad continúan siendo altas y el consiguiente grado de deserción que generan, y, el bajo o ningún interés motivacional de los estudiantes por los cursos de programación; tratándose incluso de estudiantes de Ingeniería de Sistemas. Frente a esta problemática, se ha hablado de enfoques, apalancados en las TIC, estrategias como el APP (Aprendizaje por proyectos), cambiar de paradigma o lenguaje de programación, incluso cambiar el orden en que se dan las temáticas –pero, nada parece funcionar de manera efectiva-.

Un curso típico de programación tradicionalmente se estructura de forma similar a la de un manual de programación. Se parte de conceptos fundamentales como tipos de datos, operadores, jerarquía de operadores (en algunos casos, se omite de manera voluntaria este tema), expresiones computacionales matemáticas y lógicas. Luego, se introducen las estructuras de control, funciones con sus detalles en torno a la firma de funciones (paso de parámetros y retorno de resultados), y una leve introducción a las estructuras de datos estáticas (vectores y matrices). Temáticas como la construcción de interfaces y puesta a punto de programas (test o pruebas) no son incluidos, y si lo fuesen difícilmente se podrían alcanzan a exponer.

El centrado interés institucional por el cumplimiento de los contenidos del curso, puede no encontrar eco en el estudiante, ya que puede no tener una claridad sobre la necesidad real de los mismos. Este es un aspecto, en que la falta de contextualización de un problema planteado, menoscaba su interés y sin duda, en la medida en que siente que lo que está aprendiendo no está siendo útil para su vida profesional, impactará sobre su proceso de aprendizaje.

La labor de programar, implica manejar conceptos y habilidades en dominios del conocimiento como el modelamiento de una realidad, la especificación de un problema, algoritmia, claridad en el proceso de construcción de un programa (análisis, diseño, pruebas), entre otras. Por lo tanto, la formación efectiva en la programación la constituye el equilibrio de muchos dominios y el entendimiento de las relaciones entre ellos, de tal manera que la solución de un problema quede plenamente expresada como un programa de computadora (Villalobos, 2009).

De igual manera, no es menos importante, el evitar crear programadores compulsivos, ansiosos por sentarse delante de la computadora para escribir directamente el código y "entrar en batalla" con el computador hasta obtener el programa correcto, a través de un tortuoso ciclo de escribir-ejecutar-modificar (Timarán, 2009).

2.1 MODELOS Y PARADIGMAS DE PROGRAMACIÓN

El paradigma imperativo data de los comienzos mismos de la programación; y aún continua vigente en los cursos de programación estructurada. Es este modelo el conjunto de instrucciones que constituyen un programa, le indican a la computadora la manera

en que debe encontrar la solución a un problema, al definirse en términos de los contenidos de la memoria. Los contradictores del paradigma imperativo afirman que el gran riesgo en su uso, estriba en el concebir estructuras mentales secuenciales y que necesariamente un criterio de validez lo brinda el número de líneas de codigo y los tiempos de ejecución.

Sin embargo, comprender adecuadamente un problema y diseñar una solución requieren de un importante componente de creatividad e imaginación, dejando de lado el estereotipo de personas lógicas, cuando la programación es una verdadera expresión de arte.

En el paradigma declarativo, que incluye la programación funcional y lógica, están más en pertinencia con las estructuras matemáticas, y en donde no es necesario tomar en consideración las concepciones de hardware o bajo nivel. Los programas en un lenguaje funcional se estructuran únicamente como funciones, entendiéndolas no como los clásicos subprogramas de los lenguajes imperativos, sino funciones de exclusiva naturaleza matemática. De igual manera se caracterizan por la no existencia de asignaciones de variables (en la programación imperativa, es estado de un programa está en dependencia del estado de las variables), ni tampoco incluyen estructuras como las iteraciones, por lo que se obliga que dichos procesos se implementen por medio de funciones recursivas.

2.2 VALIDACIÓN DEL MODELO FUNCIONAL CON EL LENGUAJE SCHEME

En la gran mayoría de instituciones de educación superior (IES) que ofrecen el programa de Ingeniería de Sistemas a nivel nacional, se contempla el estudio del modelo imperativo de programación que incluye la programación estructurada o procedimental y la programación orientada a objetos (OOP).

El proyecto de investigación Validación del modelo funcional en la enseñanza de la programación con el lenguaje Scheme, realizado por la Red Universitaria de Investigación en Ingeniería de Sistemas de Nariño (RUISNAR), estuvo conformado por las IES con sede en la ciudad de Pasto, Nariño.

La utilización del modelo funcional en la enseñanza de la programación comienza con las ideas de John Backus (1978) sobre la programación funcional, con pronunciaciones fuertes en contra del uso de variables, instrucciones de asignación y estructuras de control. Sus consideraciones acerca de la programación procedural como inapropiada para el modelamiento computacional de tareas matemáticas, en la medida en que no permiten el razonamiento adecuado de los programas, le llevó a proponer abiertamente la programación funcional como alternativa. En este modelo, la asignación y el uso de variables es innecesario; la recursividad sustituye a las instrucciones de control y el cálculo Lambda, propuesto por Alonzo Church (1941) proporciona el modelo matemático que hace posible razonar acerca de los programas y las funciones de orden superior pasan a ser el mecanismo clave para la reutilización de código.

El proyecto realizado por RUISNAR, permitió evaluar tres aspectos fundamentales en la solución de problemas: la comprensión del problema, la utilización de estructuras de programación y la eficacia de la solución. Además de evaluar conocimientos, se indagó sobre su apreciación respecto del desarrollo del curso (Timarán, 2009).

3. PROPUESTA DE FORTALECIMIENTO DEL PENSAMIENTO ALGORÍTMICO

En el programa de Ingeniería de Sistemas, de la Universidad de Pamplona, en atención al modelo de modernización curricular, ha establecido un curso inicial denominado “Pensamiento Computacional”, pensado no en la enseñanza de algún lenguaje de programación, sino en brindar lo propio en la promoción de habilidades de pensamiento.

La presente propuesta, toma tres aspectos fundamentales en éste propósito. En una primera instancia brinda al estudiante la identificación de componentes del sistema de cómputo y la comprensión de las interacciones que tienen lugar al ejecutarse una instrucción del programa cargado en memoria (el programa se sube a la “memoria de Instrucciones” y los datos que intervienen en un cálculo se toman de la “memoria de datos” y el resultado de la operación de igual manera se dirige a “la memoria de datos”). El microprocesador en el proceso de cálculo hace uso de registros y de la Unidad Aritmético Lógica **ALU**) en un esquema sencillo y con importante valor didáctico, que se muestra en la figura 1.

Figura 1: Preparación de la instrucción a ser ejecutada por la ALU. Fuente: los autores

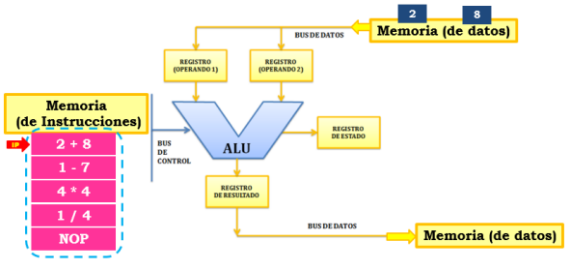
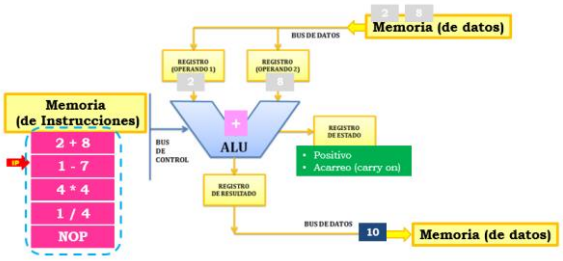
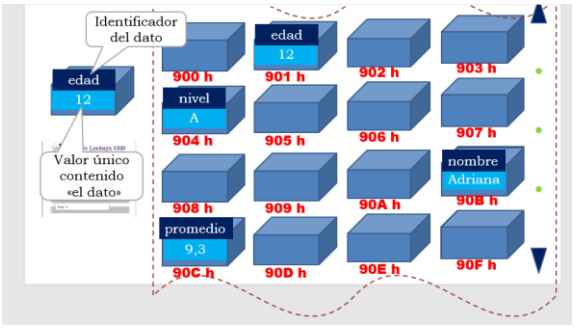


Figura 2: Ejecución de la instrucción por parte de la ALU. Fuente: los autores



El concepto de memoria es desarrollado en analogía a un "casillero" con sus compartimientos previamente identificados con un número. Este modelo mostrado en la figura 3, permite comprender el concepto computacional de *variable* como *posición de memoria* y el concepto de *identificador (del valor)* que, en los lenguajes de alto nivel, superó la necesidad de basarse en el conocimiento de direcciones de memoria para la realización de los programas, como fue el caso de la programación en Assembler (Ensamblador).

Figura 3: Modelo de la memoria y concepto de identificador. Fuente: los autores



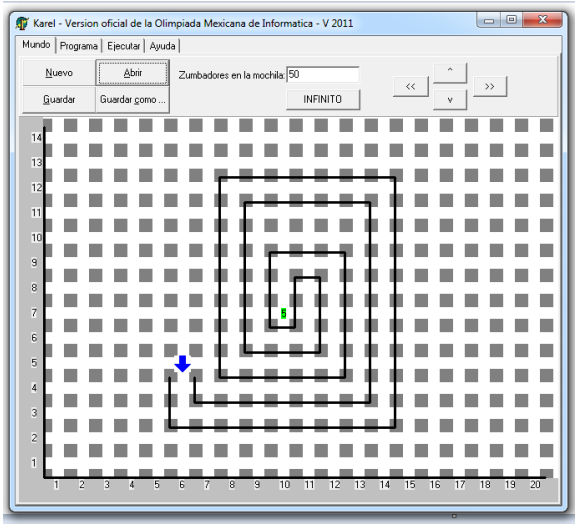
En una segunda instancia, para la comprensión de los primeros conceptos en algoritmia y de una manera divertida dar instrucciones a la computadora se empleó el programa del Robot "Karel" en su versión libre de la OMI (Olimpiada Mexicana de Informática).

Figura 4: Karel el robot – Olimpiada Mexicana de Informática. Fuente: Sitio web de la OMI

El programa del robot Karel, es una de las herramientas muy pertinente en los cursos iniciales de programación. Mediante este software se puede introducir al estudiante en un conjunto de conceptos fundantes (instrucción, programa, compilación) y en el tema de las "buenas prácticas" como la documentación, la indentación correcta del código el uso de comentarios, que informan sobre una tarea o sub tarea organizadas en bloques de instrucciones, de tal forma que programar no se percibe por parte del estudiante como el simple hecho de "reunir" instrucciones, sino que una *"misión"*, consta de etapas dentro de un contexto claro y razonado. Aunque el mundo de Karel, es muy sencillo (calles que corren paralelas en sentido horizontal, y avenidas que lo hacen de forma vertical, secciones de pared y zumbadores), la propuesta de tareas creativas que despierten la imaginación, permitirán lograr el nivel de "*abstracción"* necesario si se brinda la adecuada contextualización del problema.

Notemos como logramos incentivar la capacidad de abstracción en los estudiantes mediante la propuesta del siguiente ejercicio: "*Un grupo de científicos requieren que Karel ingrese a un laberinto subterráneo en donde se sabe existen 5 valiosas joyas de una dinastía china (representadas por 5 zumbadores). En exploraciones previas, se tiene un plano aproximado del laberinto (se muestra en el mundo). Usted debe programar a Karel para que ingrese al laberinto y cuidando de no estrellarse contra una pared del laberinto, llegue a las joyas y las guarde en su mochila. Debido al ingreso de Karel, hay pequeños derrumbes que hacen que la forma original del laberinto se pierda un poco. Como precaución, al internarse en el laberinto - cada vez que Karel daba con un obstáculo al frente- colocaba un zumbador que emite un "beep" al estar junto a él. Usted debe programar a Karel para salir del laberinto, valiéndose del sonido que emite cada zumbador (SIN RECOGERLO).* La figura 5, describe la situación del problema.

Figura 5: Karel el robot – Misión de las joyas dinastía china en laberinto. Fuente: los autores.

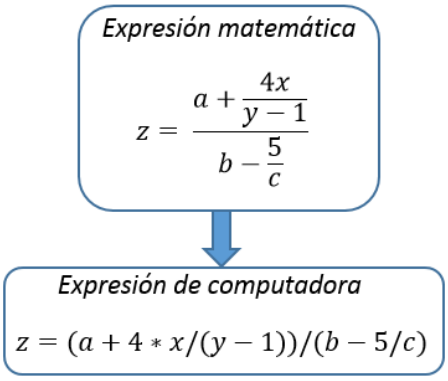


Karel posibilita la construcción del concepto de función de una manera intuitiva, puesto que se habla de "ampliar su vocabulario", de tal forma que se creen nuevas tareas (tareas de usuario) que permiten crear nuevas funcionalidades y reutilización de código.
Como se ha mencionado, no solo se tiene con Karel, a disposición un conjunto básico de instrucciones y la posibilidad de aplicar y comprender las estructuras de control de flujo, como son las condicionales e iteraciones; sino, que principalmente en la guía sistemática para los aprendizajes del estudiante, irá no solo asimilando nuevos conceptos, sino un verdadero entrenamiento que promueva habilidades de pensamiento como la composición, la abstracción, la búsqueda de patrones y la construcción de programas (algoritmos), todos ellos insumos valiosos en la promoción y fortalecimiento de pensamiento algorítmico.

En una tercera instancia se ha tomado en consideración, promover la comprensión de conceptos básicos de las matemáticas tomados en la consideración para ser programados; tales como el concepto de *"número par"* (la propuesta de un programa para determinar si un número es par, mínimamente requiere el concepto de *"divisibilidad por 2"*, de tal forma que previo a la construcción correcta de la estructura condicional –traducción del lenguaje natural a la abstracción como una expresión lógica-, el estudiante tenga clara sobre cual objeto se construye el condicional; si sobre el numero original, o sobre el residuo de éste al ser dividido entre dos).

Las relaciones jerárquicas de operadores y los operandos involucrados en una expresión matemática, son de necesaria comprensión para la traducción correcta de las expresiones matemáticas a las expresiones de computadora; que guarden el grado de coherencia y correspondencia entre la expresión matemática y su adecuado *"modelo de expresión computacional"*.

Figura 6: Traducción de expresiones matemáticas a expresiones de computadora. Fuente: los autores.



En ésta etapa, se realizó una minuciosa selección de ejercicios orientados a fortalecer las principales dificultades que presentan los estudiantes:

- tratamiento de cantidades porcentuales, y cálculo de descuentos o impuestos sobre cantidades
- comprensión de la lógica de construcción de los sistemas de numeración, necesarios en las tareas de descomposición de un número en sus cifras, o la composición de nuevas cantidades basadas en las cifras de un numero
- conocimientos básicos en Geometría plana y espacial
- comprensión de los modelos matemáticos que representan conceptos de algún área de la física (mecánica, electromagnetismo, ondas)
- comprensión de conceptos y habilidades en el tratamiento de expresiones algébricas y operaciones básicas

Como herramienta de software en este tercer momento de acercamiento a la tarea de programar, en la búsqueda de soluciones a problemas estructurados y no-estructurados, nos hemos servido del programa PseInt.

4. METODOLOGÍA DEL PROYECTO

El modelo planteado, en relación a otras propuestas similares como las que hemos descrito, consta de los componentes teóricos y metodológicos que le brindan solidez. Las dimensiones del marco conceptual que compete al modelo, son las siguientes: (1) un modelo pedagógico basada en las pedagogías activas, en que se busca la construcción de conceptos fundantes, en la medida en que dichos conceptos permiten contextualizar todo el constructo teórico de un curso de programación, (2) un conjunto de herramientas de software libre que permiten su acceso por parte de los estudiantes y materiales didácticos de soporte (de creación propia, como es el caso de las animaciones realizadas con el fin de exponer un concepto), (3) El uso de la plataforma Moodle, como un importante recurso para el seguimiento de los avances por parte del estudiante, (4) Un modelo de evaluación orientado a verificar las habilidades comunicativas y de argumentación de los estudiantes, de tal forma que sea adecuada descripción de las tareas a realizar y los resultados obtenidos.

4.1 Diseño Instruccional

El diseño instruccional es un proceso de *"arquitectura"* de las experiencias de aprendizaje, que va mucho más allá de la enseñanza. En este sentido el Diseño Instruccional realiza varios pasos, antes que cualquier enseñanza se lleve a cabo, enmarcando en mucho, la ciencia de cómo aprendemos las personas.

En consecuencia, la habilidad de evaluar las actitudes, las brechas de conocimiento y los objetivos de aprendizaje se hace esencial para este campo.

En cuanto a la claridad que se debe tener al momento de asumir un proceso de diseño, podemos destacar las siguientes:

- evaluar y definir las necesidades de formación de la población objetivo
- definir objetivos de aprendizaje claros y accionables, junto a la producción de contenido atractivo, en correspondencia con ellos
- conceptualizar gráficos, multimedia, interfaces de usuario, como productos de instrucción
- aplicar principios de buenas prácticas en atención a los requerimientos de tecnologías de aprendizaje
- desarrollar ejercicios tareas con el suficiente grado de creatividad, de tal manera que se pueda maximizar el potencial de aprendizaje de los estudiantes
- integrar recursos de soporte, herramientas que permitan la construcción de materiales multimedia (audios, animaciones, videos, capturas de pantalla, diversos escenarios de situaciones problema)
- idear métodos de evaluación que se orienten al fortalecimiento de habilidades comunicativas y argumentativas en los estudiantes

Algunas de las pautas consideradas en todo un proceso de Diseño Instruccional, pueden girar en torno a preguntas como:

¿Se transmite mejor este bloque de texto como un gráfico interactivo?

¿Es probable que mis estudiantes desarrollen este curso desde cualquier lugar por medio de un dispositivo móvil?

¿Se puede dividir un material básico en módulos de micro aprendizaje para que sean mejor comprendidos?

4.2 Marco Conceptual

La definición del marco conceptual se definió en términos de los ejes articuladores para la adquisición de competencias y habilidades en programación:

(1) modelo de ejecución de instrucciones en los componentes de Hardware de almacenamiento-comunicación-procesamiento, (2) modelo de la memoria, (3) modelo de descomposición de una tarea en sub-tareas, (4) abstracción de tareas en el mundo de Karel, (5) algoritmia.

Cada uno de los ejes mencionados constituye su propia conceptualización y habilidades de formación a diferentes niveles de profundización.

Figura 7: Modelo conceptual con los ejes articuladores de formación en competencias en Programación.
Fuente: los autores

5. RESULTADOS

Como indicadores principales, se definieron:

(1) la percepción que los estudiantes tienen del curso de programación, con la herramienta de creación de programas a ser ejecutados por el robot Karel, (2) la percepción de los estudiantes acerca de los materiales de instrucción empleados para el aprendizaje de conceptos, (3) la mejora en las habilidades de argumentación y comunicación en los estudiantes, al momento de sustentar tareas y pequeños proyectos de programación.

En cuanto al primer ítem, los estudiantes afirmar haber ganado confianza en el compromiso de creación de programas, y con el apoyo de una metodología orientadora que los estructura a tener en cuenta no solo los aspectos de la lógica de solución de un problema, sino "las buenas prácticas" que permiten escribir programas legibles y por tanto, de fácil comprensión. El uso de la herramienta Karel, es una forma muy amena de adquirir los fundamentos de programación y, lo más importante es la posibilidad de evidenciar las acciones producto de las instrucciones, o los bloques de instrucciones con que se construyen *"nuevas palabras"* para Karel, o el diseño de condicionales y repeticiones.

6. CONCLUSIONES

La enseñanza de la programación siempre estará sujeta a grandes retos, y toda propuesta debe tomar en consideración, en primera instancia el paradigma sobre el cuál se orientará la instrucción, pues como fue expuesto en el presente documento; no es lo mismo si la enseñanza se orienta desde el modelo estructurado, en cuyo caso son relevantes los conceptos de variable, funciones (como sub programas) y la adecuada construcción de expresiones de computadora; así como el diseño correcto de bloques de codigo y/o módulos. En tanto, que, si la enseñanza se propone desde un paradigma como el funcional, las consideraciones del paradigma imperativo pierden toda su importancia.

En la presente propuesta el modelo de referencia es el estructurado, en virtud que sigue siendo el modelo de iniciación adoptado por la Universidad de Pamplona. Un curso de iniciación a la programación, como el concebido como *"Pensamiento Computacional"* brinda unas grandes posibilidades de promover y mejorar en los estudiantes la mejor comprensión de los modelos matemáticos, como lenguaje de ciencias, en virtud de su potencial papel como computistas en equipos de trabajo de colaboración científica; y en principal medida, por el énfasis en *"Ciencias Computacionales"*, del programa de Ingeniería de Sistemas de la Universidad de Pamplona.

La pertinencia de una herramienta como el Robot Karel, se puede validar dado el grado de autoconfianza expresado por los estudiantes en el proceso de creación de un programa, en donde todas las recomendaciones en procura de la legibilidad de un programa, se reconoce como un elemento integrador necesario, más allá de un aspecto normativo.

RECONOCIMIENTO

Agradecer la participación activa en el desarrollo de las actividades del proyecto a los estudiantes integrantes del Semillero de Investigación en Inteligencia de Datos y Computación (GIIDAC) adscrito al programa de ingeniería de sistemas de la Universidad de Pamplona, campus de Villa del Rosario y en general a los estudiantes participantes del estudio del Programa de Ingeniería de Sistemas.

REFERENCIAS

Aguirre L, Chirinos D. (2017). *TIC en la educación informática y herramientas digitales*, Lima, Perú: Macro, ISBN: 978-612-304-545-6.

Basogain Olabe, X. et al. (2015). Pensamiento computacional a través de la programación: Paradigma de Aprendizaje. Revista de Educación a distancia (RED), Universidad de Murcia, España, (46), 3–7.

Briceño Guevara, O. et al. (2019). Diseño didáctico para el desarrollo de destrezas básicas de programación por medio del programa Scratch a estudiantes del grado quinto del colegio diocesano de Duitama. Revista Colombiana de Tecnologías de Avanzada (RCTA), 15(34), 4–7.

Cairó, O. (2005). *Metodología de la Programación*, México, D.F.: Alfaomega, ISBN: 970-15-1100-x.

De Zubiría Samper, J. (1999). *Tratado de Pedagogía Conceptual: Los Modelos Pedagógicos*, Santa Fe de Bogotá, D.C.: Fundación Alberto Merani para el desarrollo de la inteligencia, ISBN: 958-9405-04-5.

Ferreira, A. y Rojo, G. (2015). Enseñanza de la programación. Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología (TE&ET), Un. Nacional de Río Cuarto, Argentina, 1(1), 1–8.

Heidegger, M. (2005). *¿Qué significa pensar?*, Madrid: Editorial Trotta, Colección Estructuras y Procesos Serie Filosofía, Madrid, ISBN: 84-8164-788-8.

López, J. (2009). *Algoritmos y Programación Guía para docentes.* Segunda edición. Obtenido de http://eduteka.org/GuiaAlgoritmos.php

López M., Whalley, J., Robbins, P. y Lister, R. (2008). *Relationships between reading, tracing and writing skills in introductory programming.* Proceedings of the fourth International Workshop on Computing Education Resarch, Sidney, Australia, Septiembre 06-07, 2008).

Murcia Florián, J. (1999). *El camino del saber: Elementos teórico-metodológicos básicos del proceso investigativo*, Santa Fe de Bogotá, D.C.: Ediciones USTA, ISBN: 958-631-243-7.

Murcia Florián, J. (1997). *El proceso de conocimiento*, Santa Fe de Bogotá, D.C.: Ediciones USTA, ISBN: 958-631-244-5.

Ortiz, M. (2006). *Manual Karel el Robot.* Primera edición. Comité Veracruzano de Informática. Obtenido de: https://www.academia.edu/7940276/KAREL_EL_ROBOT

Schank, R. (1997). *Aprendizaje Virtual*, McGraw-Hill Interamericana Editores, México, D.F, ISBN: 0-7863-1148-7.

Timarán, R. et al. (2009). *Un nuevo enfoque en la enseñanza de la Programación*, Ruisnar, Editorial Universitaria Universidad de Nariño, ISBN: 978-958-9479-98-8.

Villalobos, J. y Calderón, N. (2009). *Proyecto cupi2: un enfoque multidimensional frente al problema de enseñar y aprender a programar*. Revista de Investigaciones UNAD, 8(2), 45-64.

SITIOS WEB

Sitio oficial de la Olimpiada Mexicana de Informática, para descarga del compilador de Karel https://www.olimpiadadeinformatica.org.mx/OMI/OMI/Material/Karel_el_Robot.aspx