Propuesta metodológica para la enseñanza de la programación en la Unidad Académica de Ingeniería de la UAGro

Proposed methodology for teaching programming in the Academic Unit UAGro Engineering

René Edmundo Cuevas Valencia*

Héctor Bautista Jiménez**

Juan Carlos Medina Martínez***

Fecha de recepción: 2 de abril de 2013
Fecha de aceptación: 5 de mayo de 2013

Resumen

El presente artículo, se desarrolla con la finalidad de describir una nueva propuesta metodológica, para la enseñanza de la Unidad de Aprendizaje de programación, en la Unidad Académica de Ingeniería, para promover la formación académica adecuada para los estudiantes inscritos en dicha Unidad de Aprendizaje de la carrera de Ingeniero en Computación. Como es sabido, la educación actual en esta institución se fundamenta en las Tecnologías de la Información y la Comunicación (TIC's), por ello la propuesta presentada se cimenta en el uso de herramientas educativas de esta modalidad que ha venido revolucionando la Educación en México desde hace casi dos décadas, por eso mismo dichas tecnologías deben ser aprovechadas por el estudiante y el do-

* Universidad Autónoma de Guerrero Unidad Académica de Ingeniería. Chilpancingo Gro., México. Correo electrónico: reneecuevas@uagro.mx

** Universidad Autónoma de Guerrero Unidad Académica de Ingeniería. Chilpancingo Gro., México. Correo electrónico: yoker_yoker@yahoo.com.mx

*** Universidad Autónoma de Guerrero Unidad Académica de Ingeniería. Chilpancingo Gro., México. Correo electrónico: jcmedina74@yahoo.com.mx

cente, para dar una mejor solución a cualquier problema que se le presente al momento de aplicar los conceptos de programación. Como parte de los resultados se pretende que el estudiante aplique la propuesta etapa por etapa para que se apliquen los resultados de una manera concreta y eficaz.

Palabras clave: TIC's, educación, programación, metodología para la enseñanza.

Abstract

This article, it develops with the aim of describing a new methodological proposal, for the teaching of unit of programming learning, in the Academic Unit of Engineering, to promote adequate academic training for students registered in the Unit of learning of the career of Computer Engineer. As is known, the current education in this institution is based on the Information Technology and Communication (TIC's), therefore the proposal is founded on the use of educational tools of this modality that has been revolutionizing education in Mexico from for almost two decades, by that same technologies must be utilized by both the student and by the teacher, to give a better solution to any problem that they are present at the time of applying the concepts of programming. As part of the results it is intended that the student to apply the proposal stage by stage to the implementation of the outcome of a practical and effective manner.

Key words: ICT, education, programming, teaching methodology.

1. Introducción

Desde hace mucho tiempo, la carrera de Ingeniero en Computación de la Unidad Académica de Ingeniería en la Universidad Autónoma de Guerrero, se ha visto en la necesidad desde sus inicios en los años 90´s, de crear ingenieros con gran habilidad y capacidad de programar de una manera excelente, dificultad que no se ha solucionado hasta la actualidad. El problema surge cuando el alumno egresado no cumple al ciento por ciento el perfil que debería cubrir respecto a la carrera de Ingeniero en Computación; dando como resultado una secuencia de problemáticas como conocimiento obsoleto o incompetente.

La investigación se refiere puntualmente a la problemática que se vive en la Unidad Académica de Ingeniería por el bajo índice de egresados con un buen perfil de Ingeniero en Computación, que se puede definir como un bajo índice de excelentes ingenieros.

Para analizar esta problemática, es necesario de mencionar sus causas. Una de ella es la re-

probación. Se entiende por reprobar como el hecho de no haber sido aprobado o apto del conocimiento necesario de una Unidad de Aprendizaje.

La reprobación se vive a diario en todas las Unidades de Aprendizaje, pero se denota más en la de programación de la carrera de Ingeniero en Computación, incluso en otras carreras que llevan esa Unidad de Aprendizaje. Los estudiantes lo viven por experiencia propia o de compañeros de clase.

Dado los motivos mencionados, son fundamentales para abordar la investigación, y definir una solución a dicho problema, por lo que se desarrolló una metodología que le permitiera al facilitador abordar de una manera más concreta y definida la Unidad de Aprendizaje de programación, de tal forma que el alumno aprenda bien a programar, para evitar el alto índice de reprobación.

2. Planteamiento

La educación en la Unidad Académica de Ingeniería es tradicional y típica, metodologías sin resultado que han usado los docentes llamados ahora facilitadores. Este problema se vive desde que surgió la carrera de Ingeniero en Computación en la Unidad Académica, por lo tanto surge la necesidad del docente, de mejorar dichas metodologías para obtener buenos indicadores en egresados con un verdadero alto porcentaje de conocimientos y habilidades, que les permitan poder competir en el ámbito laboral.

En torno a esta problemática que se vive en dicha institución educativa, ha surgido una propuesta metodológica que se basa en las TIC's, para mejorar el desempeño educativo del alumno a lo largo de su carrera, desde el primer semestre aprovechando esta metodología. Es importante que el facilitador descubra el talento o en su defecto, la habilidad para programar de cada uno de los estudiantes, con el fin de identificar los problemas de aprendizaje que tienen cada uno de ellos desde que se inicia a forjar como ingeniero.

La propuesta debe ser concreta, en la que se defina por partes de como el alumno debe ir progresando gradualmente el conocimiento de forma específica, de acuerdo a las etapas que se establezcan. Cada una de las etapas debe cumplir puntualmente un objetivo, de no ser así, el rumbo que tomará el conocimiento, será equivocado y perjudicial para el alumno, por lo tanto se deberá ser cuidadoso en la manera de concretar las ideas por cada etapa.

Como bien se explicó, que las etapas deben cumplir de manera perfecta un objetivo, el alumno puede concentrarse en el objetivo de la etapa siguiente. Acorde a las etapas que vayan pasando, se debe sufrir un cambio relativo en el que la dificultad debe subir y que a la vez, esté relacionado con la nueva etapa siguiente para facilitar la transición entre una etapa y otra. Para garantizar que se cumpla el ciclo de las etapas, todas deben haberse realizado de manera secuencial sin distinción alguna.

Al finalizar todas las etapas, el alumno puede presentar una respuesta bien definida de la solución al problema del que esté inmerso. Es importante que al finalizar la última etapa se verifique si la solución al mismo se ha cumplido, de no ser así, el alumno tiene el privilegio de regresar a alguna de las etapas que le permita solucionar su error.

El alumno inexperto en programación, va a adquirir un conocimiento más rápido y obteniendo un avance académicamente mucho mejor usando esta metodología, comparándola con la tradicional. Cabe señalar que la

propuesta debe enfocarse hacia personas que tienen poca o nula experiencia programando, es decir, hacia estudiantes que no tienen el conocimiento básico de programación.

3. Metodología propuesta

Se propone una metodología que contemple una solución gradual a un problema, esto significa que el alumno debe dar soluciones parciales hasta ofrecer una respuesta definitiva, concreta y consistente.

La metodología se plantea en cinco etapas para que el alumno deba dar una solución a cierto problema que tenga que resolver en un programa. Las cinco etapas de las que se habla, van desde la más básica y fácil, hasta la más completa y difícil de llegar, las cuales son:

- Nivel 1: Implica realizar un análisis y clarificar conceptos que permitan definir una solución general del problema.
- Nivel 2: Realizar una solución general que se base en un lenguaje natural. Se propone un lenguaje natural basado en UML usando los casos de uso.
- Nivel 3: Diseñar diagramas de flujo para la resolución del problema. Dicho diagrama debe basarse en el caso de uso que se realizó en el nivel 2, dicho diagrama debe ser más concreto que el caso de uso.
- Nivel 4: Hacer un algoritmo en un pseudocódigo (falso lenguaje) [1] más detallado basado en el diagrama de flujo hecho anteriormente.
- Nivel 5: Escribir un programa basado en cualquier lenguaje de programación (lenguaje formal) basado en el nivel 4.

Cada nivel debe cumplir las expectativas que tiene como objetivo, por consiguiente superar los del nivel anterior. La propuesta considera que los cinco niveles constituyen una pirámide, la cual empieza desde el nivel más básico que de acuerdo en la posición y área proporcionada, es el tiempo en el cual el alumno se tarda más en resolver dicho nivel. Se le llama la pirámide del tiempo, la que incluye todo lo anteriormente mencionado, tal como se muestra en la Figura 1.
Se debe destacar que del nivel 1 al nivel 5, se propone usar software libre adecuado, que le facilite al alumno dar respuesta al objetivo planteado.

3.1. Nivel 1

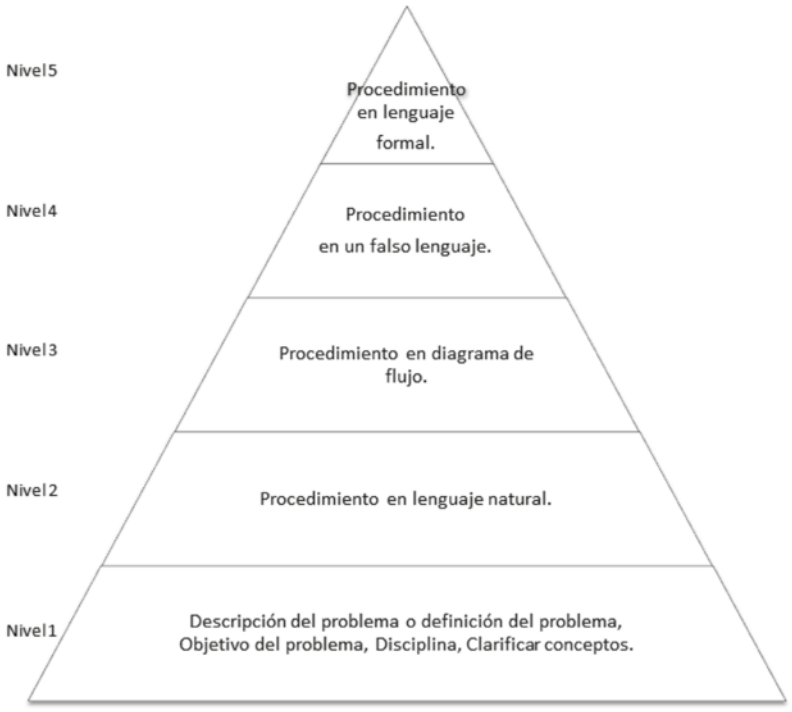
En el nivel uno se debe comprender, definir claramente y conceptualmente el problema; como consecuencia se tendrán que despejar todas las dudas al respecto.

En este nivel es importante que se defina el o los objetivos específicos sin ambigüedades, ya que es de suma importancia que se realice un análisis minucioso, para que se determine la solución del programa.

En la Figura 1, se muestra que el nivel 1 determina la disciplina y por ello se clarifican conceptos técnicos, que pudieran ser relevantes al momento de realizar tareas en niveles posteriores. Los conceptos deben coincidir con la disciplina de la que se esté tratando, por lo consiguiente, esto beneficia al alumno en no ser ambiguo o abordar otra temática que esté en otro campo o área.

El tiempo no debe escatimarse en ningún momento en el trayecto del análisis en este nivel. Entre mayor tiempo se tome, es menor el riesgo de cometer un error en niveles superiores.

Figura 1. Pirámide del tiempo.



Fuente: elaboración propia.

Para ejemplificar el nivel, se indica en un ejemplo:

- **Descripción del problema**: Se necesita un programa, que calcule y muestre en pantalla si un número introducido en el programa es primo o no lo es.
- **Objetivo del problema**: Calcular si un número introducido en el programa es primo o no.
- **Disciplina**: Matemáticas.
- **Clarificar conceptos**: Número primo. ¿Qué es un número primo?, ¿Cómo calcular un número primo?, ¿Cómo descubrir si es un número primo?, si hay excepciones ¿Cuáles son?

¿Qué es un número primo?: un número primo es cualquier número natural mayor a 1, tenga exactamente 2 divisores distintos, los cuales son el uno, y el número mismo [2].

¿Cómo descubrir si es un número primo?: la forma de descubrir si es un número primo, es ir dividiendo consecutivamente el número a identificar desde el 1 hasta el número a identificar.

¿Cómo calcular un número primo?: Número 3: 3/1=**3**, 3/2=1.5, 3/3=**1** -> El número 3 es primo porque tiene solo 2 divisores que son el 1 y el 3. Número 4: 4/1=**4**, 4/2=**2**, 4/3=1.3, 4/4=**1** -> El número 4 no es primo, porque tiene 3 divisores, los cuales son el 1, 2 y 4.

Si hay excepciones ¿Cuáles son?: Sí las hay, y son: no pueden ser número negativos, ni el número uno o el cero.

3.2. Nivel 2

En el segundo nivel, el alumno debe de dar solución parcial de manera general usando un lenguaje natural, tal como el lenguaje que habla en su vida diaria.

El alumno debe revisar y basarse en el nivel anterior, dejando claramente definidos los objetivos, la terminología y dudas para que se pueda analizar una posible solución. En el actual nivel se debe tardar menos tiempo para transitar al siguiente.

Para el alumno primerizo o inexperto, esta etapa es muy importante porque en este momento sus ideas empiezan a fluir, en el cual el panorama se amplía hasta el grado de analizar y comparar con detenimiento, si esas ideas de solución son las más adecuadas para darle seguimiento.

Es fundamental darle la debida importancia a este nivel, porque se define la lógica para la resolución del problema. La lógica es extremadamente sustancial para que cualquier problema sea resuelto correctamente, es decir, que si la lógica falla, el resultado será incorrecto y por ende, el programa no dará solución al problema original.

En la propuesta se contempla usar el Lenguaje Unificado de Modelado (UML) [3] el cual le permite al alumno escribir y expresar de forma general sus ideas para resolver dicho problema. En este nivel el alumno se torna con más libertad y autoridad para decidir cuál, o cuáles serán sus vías de solución.

El Lenguaje de Modelado Unificado es muy extenso, el cual implica que, hacer todos los diagramas no tendría gran relevancia para un programa relativamente sencillo, es por eso que únicamente se enfoca hacia el diagrama de casos de uso.

Figura 2. Representación de un diagrama de Casos de Uso

Fuente: elaboración propia.

Un caso de uso es una notación no una metodología [4], el cual denota gráficamente los pasos y este debe ser de fácil entendimiento; es precisamente el motivo de los casos de uso aunque se diga que no es de relevancia por el hecho de que los diagramas UML se enfoquen a sistemas orientado a objetos (ver Figura 2).

Aunque en los inicios del aprendizaje de la programación se utilice una programación procedural, el empleo de los casos de uso es vital ya que además agiliza el aprendiza-

je, también ayuda a que en algunos años, el alumno tenga los conocimientos y habilidades para desarrollar sistemas grandes, en el que descubra que las bases del conocimiento de programación se encuentran en el análisis y planeación de un proyecto apoyado en el UML.

El aprovechamiento del UML, concretamente de los casos de uso, el alumno centrará el objetivo del problema en una solución de una forma gráfica, combinado con el lenguaje natural de su idioma le facilitará enfocarse en el razonamiento del problema sin preocuparse de sintaxis o reglas gramaticales, es por ello que la propuesta en el nivel 2 se enfoca en los diagramas de casos de uso.

Para describir un caso de uso, se puede decir que son los principales medios para obtener la funcionalidad del sistema desde la perspectiva del usuario [4]. Los casos de uso están definidos por actores, casos y relaciones que se pueden dar entre estos mismos.

Un actor es un usuario del sistema, ya sea una persona o ente, que tenga interacción o en su defecto, otro sistema computacional que realice una acción o que proporcione un valor al propio sistema [4].

Un caso de uso es una representación general del comportamiento del sistema, definiendo los pasos y reglas que deberán ser llevados a cabo, incluso denotando descripciones y comentarios en el diagrama que represente el caso de uso [4].

Una relación dentro del diagrama de casos de uso puede utilizarse para relacionar la actividad entre un actor sobre uno, dos o más casos de uso, o asimismo, algo no muy común, entre dos actores. Estas relaciones pueden incluir un caso de uso dentro de otro, también puede extenderse hacia otro caso de uso, o en su defecto una simple relación.

En el problema que se resolvió en el nivel 1 se tuvo un objetivo, dicho que se puede resolver y llevar a cabo en el nivel 2 utilizando los casos de uso. En este segundo nivel se determina como va a interactuar el sistema con todos los actores y los casos que se puedan presentar, para ello se ejemplifica la propuesta en la Figura 3.

Figura 3. Diagrama de casos de uso para determinar un número primo

Fuente: elaboración propia.

En la Figura 3, se detalla un diagrama de casos de uso para determinar si un número es primo o no lo es. En el diagrama de caso de uso anteriormente expuesto, se expresa que cada caso explica de forma general una actividad, reflejado en un verbo en gerundio, en el cual recae la tarea concreta a realizar[2]. Dicho diagrama resume en todo momento el cómo y cuándo se debe realizar una tarea. En este contexto el nivel 2 será el origen del nivel 3.

3.3. Nivel 3

El nivel 3 plantea utilizar una ideología similar planteada al del nivel 2, debido a que en éste se propone utilizar diagramas en el modo gráfico que faciliten la transición de un nivel a otro aumentando la dificultad, pero a la vez asumiendo un nivel gradual que el alumno no perciba dificultades de manera brusca.

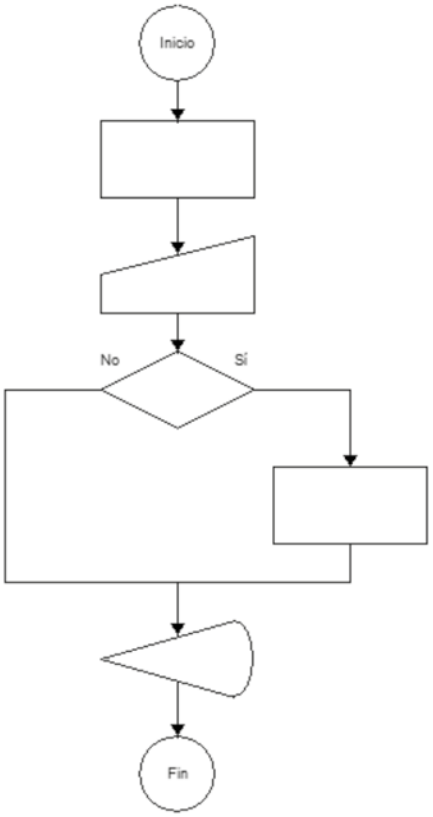
En el diagrama de flujo se pretende interpretar lo que se diseñó en el diagrama de casos de uso, de esta manera cada caso debe ser plasmado en pasos más concretos, dándole la facilidad al alumno de ver gráficamente el flujo de los datos.

Se concluyó que en el nivel 3 se implementarán los diagramas de flujo, ya que ofrecen la ventaja de una mayor comprensión del proceso a través de un dibujo, para seguir la secuencia de lo que se plasmó en el segundo nivel, debido a que el cerebro capta y reconoce más fácilmente los dibujos.

Los diagramas de flujo le permiten al alumno detectar y mejorar problemas en el proceso de un flujo. Dicho diagrama representa un algoritmo[3] que estructura concretamente los pasos de forma gráfica, para dar solución a un problema en particular. Cada diagrama de flujo es diferente con el simple hecho de tratar objetivos diferentes, pero todos siguen una notación similar, en el que un diagrama tiene un inicio y un final.

Se usan los diagramas de flujo para ver el panorama completo del flujo que conlleva la solución, para que en su momento se detecten errores que puedan ser corregidos a tiempo.

Figura 4. Representación de un diagrama de flujo



Fuente: elaboración propia.

La simbología es la misma para todos los diagramas de flujo, en este concepto unos símbolos denotan la entrada de datos, salida o procesamiento de información.

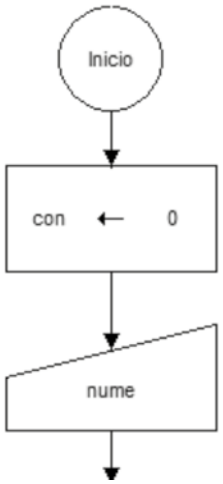
2 La herramienta propuesta para realizar los casos de uso es ArgoUML [5].

3 La herramienta propuesta para realizar los diagramas de flujo es FreeDFD [6].

El diagrama de flujo que se muestra en la Figura 4, es una muestra de cómo el flujo de datos puede cambiar. La parte interesante de los diagramas de flujo, es visualizar gráficamente el flujo de la información hasta concluir el ciclo del mismo.

El tratamiento de datos en todo proceso dentro del diagrama de flujo, tiene como particularidad relacionar dos o más datos, eso influye en utilizar los llamados operadores, los cuales son de tipo lógico, de relación, aritméticos y asignación.

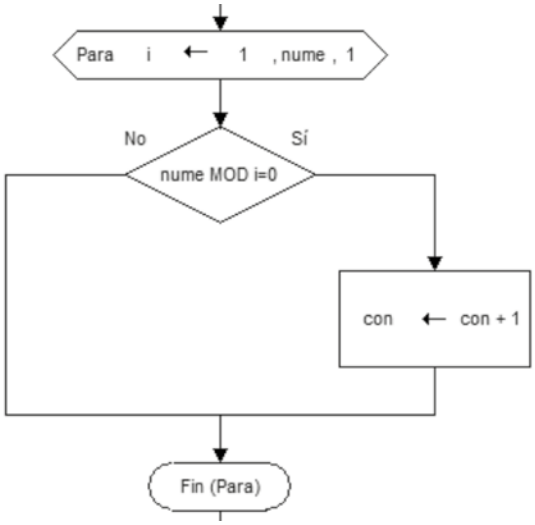
Figura 5. Primera parte del diagrama de flujo.



Fuente: elaboración propia.

Los operadores lógicos permiten relacionar dos o más variables que se relacionan mediante operaciones lógicas, en comparación de los operadores de asignación que asignan información. Los operadores de relación hacen preguntas sobre alguna situación sobre dos datos, por el contrario, los operadores aritméticos realizan operaciones matemáticas sencillas sobre dos o más datos.

Figura 6. Segunda parte del diagrama de flujo.



Fuente: elaboración propia.

El algoritmo que corresponde al diagrama de casos de uso que se ejemplificó en el nivel 2, se realizará por bloques, dándole solución concretamente a cada caso de uso, del diagrama en el nivel anterior.

El diagrama de flujo se dividió en partes o bloques para identificar cada caso de uso. En la Figura 5 detalla claramente el inicio del diagrama, posteriormente asigna la cantidad de cero a la variable con, y por último introduce un número al sistema. Esto concluye que se cumple el primer caso de uso del diagrama realizado en el segundo nivel.

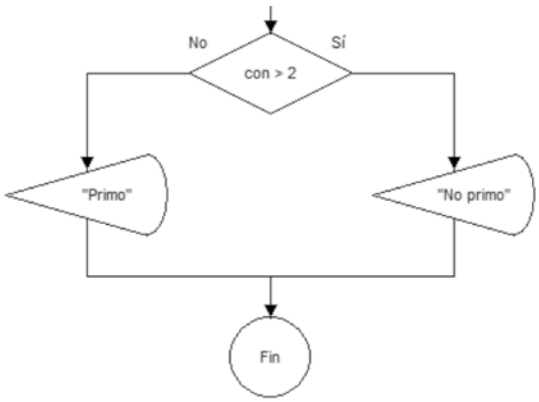
El segundo caso de uso del nivel 2, calcula si el número ingresado es primo, procedimiento que es mostrado en la Figura 6. Analizando el diagrama de flujo en este segundo bloque, es interesante observar que se hace un recorrido desde 1 hasta el número introducido, por consiguiente, en cada uno de éste se revisa si el resultado de la división es

cero, de esta manera se contabilizan las veces que el número es divisible.

Al finalizar las operaciones y comparaciones, se pregunta si las veces que el número fue dividido se realizó más de 2 veces, entonces muestra en pantalla que no es primo, en caso contrario muestra la leyenda en pantalla que es primo (ver Figura 7). Esta parte final del algoritmo de flujo es equivalente al último caso de uso del diagrama que se elaboró en el segundo nivel.

El diagrama de flujo completo se conforma uniendo las Figuras 5, 6 y 7 de forma vertical. El algoritmo definitivo es un método concreto, que resuelve el objetivo que se definió en el nivel 1. El programa propuesto para realizar dicha tarea es FreeDFD[4] [6].

Figura 7. Tercera parte del diagrama de flujo.



Fuente: elaboración propia.

3.4. Nivel 4

El nivel 4 plantea una penúltima etapa para llegar a concluir en una solución, que brinde servicio o resultados a un cliente, es decir que el cliente haga uso de un producto como tal.

El presente nivel es el que se encarga de definir secuencialmente los datos, al igual que en el diagrama de flujo, con la diferencia de que el algoritmo en pseudocódigo (falso lenguaje), no utiliza gráficos para representar la lógica. El algoritmo en pseudocódigo es una serie de instrucciones, que asemeja su estructura a un programa (por eso llamado falso lenguaje), por este motivo es importante desarrollarlo porque el último nivel se basa en este algoritmo, que prácticamente es el mapa o la documentación del programa.

El algoritmo en pseudocódigo es fundamental, ya que después de haberlo concluido, es relativamente fácil hacer un programa en cualquier lenguaje de programación basándose en dicho algoritmo.

Los diagramas de flujo tienen los mismos operadores que el pseudocódigo, lo que los diferencia es la forma de representar una instrucción, dado que en el diagrama se representa gráficamente y en el pseudocódigo con letras. Una de las ventajas que se tiene al usar los algoritmos en pseudocódigo, es que tiene la flexibilidad de escribir las órdenes en un lenguaje coloquial, es decir, en español.

El trabajo que se hace en los niveles anteriores, es significativo para llegar a elaborar un algoritmo bien estructurado en pseudocódigo, por este motivo, se muestra en la Tabla 1 el procedimiento para llegar a la solución del objetivo que se tiene escrito en el primer nivel.

El algoritmo en pseudocódigo es el reflejo casi real del código de un programa, tal y como se ejemplifica detalladamente y de manera legible el procedimiento en la Tabla 1. De esta demostración queda claro y convin-

4 La herramienta libre, propuesta para realizar los diagrama de flujo más fácil y rápido.

cente que el algoritmo en pseudocódigo debería plantearse en el nivel 4.

Al finalizar el nivel 4, el tiempo que se demoraría un alumno en programar secuencialmente el algoritmo en pseudocódigo, es relativamente menor que si se hubiese hecho directamente sin haber seguido el camino de la propuesta que se plantea en este artículo.

En la Tabla 1, se ve reflejado el trabajo que se ha seguido durante todos los niveles anteriores, por consiguiente el resultado que se escriba en el próximo nivel (código del programa) será estrictamente basado en el algoritmo de pseudocódigo. El programa que se propone para la realización de algoritmos en pseudocódigo es PSeInt [7], ya que es una herramienta libre y gratuita para escribir algoritmos en pseudocódigo.

Tabla 1. Algoritmo en pseudocódigo del nivel 4

Algoritmo del primo
1: **Proceso** primo 2: con <- 0; 3: **Leer** nume; 4: **Para** i<-1 Hasta nume Con Paso 1 **Hacer** 5: **Si** nume MOD i = 0 **Entonces** 6: con<-con+1; 7: **FinSi** 8: **FinPara** 9: **Si** con > 2 **Entonces** 10: Escribir "No primo"; 11: **Sino** 12: Escribir "Primo"; 13: **FinSi** 14: **FinProceso**

Fuente: elaboración propia.

3.5. Nivel 5

El último paso propuesto es el nivel 5, en el que se transforma un algoritmo en pseudocódigo a un programa, o en su defecto un sistema computacional.

La transición de un pseudocódigo a un lenguaje de programación es muy fácil, el problema surgirá si no se sabe correctamente la sintaxis del lenguaje en el que se requiera escribir. El resultado del nivel 5, es producto de todo el esfuerzo que se realizó en los cuatro niveles anteriormente descritos.

El lenguaje de programación que se toma en cuenta en esta etapa para ser el más adecuado y correcto para personas que se les hace difícil la programación es Python.

Se prefiere Python por muchos factores, como son:

- Es un lenguaje similar a un algoritmo en pseudocódigo.
- Es multiplataforma.
- Los programas son muy compactos.
- El código es entendible.
- Es un lenguaje cómodo, rápido y flexible.
- Es un lenguaje limpio y ordenado.

Para comparar el resultado del algoritmo en pseudocódigo con el programa desarrollado, se debe realizar completamente dicho programa. Es importante para el alumno identificar cómo evoluciona el proyecto desde el nivel uno hasta el nivel 5, así mismo dando resultando el código en el lenguaje python (ver Tabla 2).

Tabla 2. Código del programa primo en python

Programa primo

con=0
nume=input("Escriba un número: ")
for x in range(1,int(nume)+1):
if int(nume)%x==0:
con=con+1
if con > 2:
print("No primo")
else:
print("Primo")

El resultado es magnífico, si se compara con el algoritmo en pseudocódigo, ya que son menos líneas de código generado en dicho lenguaje. Por tal motivo, al concluir se demuestra que es factible el uso de Python como primer lenguaje de programación, o en caso contrario, como un lenguaje que impulse el conocimiento.

4. Resultados

Los resultados propuestos a mediano y largo plazo son; que el estudiante se autoevalúe provocando que se motive y mejore su conocimiento y habilidades en un par de años. Es importante disminuir el índice de reprobación a lo largo de la carrera de Ingeniero en Computación en la Unidad de Aprendizaje de programación.

Se espera que en un máximo de seis meses en el proceso de enseñanza-aprendizaje de los maestros, se instruyan en el uso de las TIC's, para aplicarlas en los estudiantes.

Se estima que en un máximo de cuatro años, la Unidad Académica de Ingeniería sobresalga académicamente egresando estudiantes de calidad, con conocimientos, habilidades, fortalezas y destrezas que les permitan competir en el ámbito laboral.

Es valioso no distraerse en ningún momento en la aplicación de la propuesta, de lo contrario ningún resultado esperado será favorable o confiable.

5. Referencias

[1] Juan Carlos López García. Segunda edición, 2007, 2009. Facultad de Ingeniería. UNAM. *Diagramas de flujo, pseudocódigo y pruebas de escritorio*.

[2] Miguel Ángel Abánades Astudillo. *Los Números Primos: de Euclides a Internet*. Ingeniería Técnica en Informática de Sistemas Centro de Estudios Superiores Felipe II (UCM).

[3] Geoffrey Sparks, Sparx Systems, Australia. *El Modelo Dinámico*. Introducción al UML. http://www.craftware.net/es/descargas/modelo_dinamico.pdf

[4] Geoffrey Sparks, Sparx Systems, Australia. *Una Introducción al UML*. Introducción al modelado de sistemas de software usando el Lenguaje Unificado de Modelado (UML). http://www.exa.unicen.edu.ar/catedras/modysim/teoria/casos_de_uso_a.pdf

[5] CollabNet, 2001 - 2009. http://argouml.tigris.org/.

[6] Nelson Castillo Izquierdo, Eduardo Daza Castillo, Fabián Cárdenas Varela. FreeDFD: Editor e intérprete de diagramas de flujo. http://freedfd.freaks-unidos.net

[7] Pablo Navora, PSeInt. http://pseint.sourceforge.net/

[8] Raúl A. Trejo, Rubén D. Santiago, Lourdes Quezada, Francisco Delgado, 2003. Instituto Tecnológico y de Estudios Superiores de Monterrey. *La Enseñanza De La Programación Dentro De Un Modelo De Integración Curricular: La Experiencia Del Proyecto Principia*. http://lsm.dei.uc.pt/ribie/docfiles/txt20031212172724TCI15.pdf

[9] Yolanda Martínez Treviño, 2005. Tecnológico de Monterrey *En busca de una nueva forma de enseñar a programar*. http://www.mty.itesm.mx/rectoria/dda/rieee/pdf-05/28(DTIE).YolandaMtz..pdf

[10] Ariel Ferreira Szpiniak, Guillermo A. Rojo. Universidad Nacional de Río Cuarto. *Enseñanza de la programación*. http://teyet-revista.info.unlp.edu.ar/files/No1/09_Ensenanza_de_la_programacion.pdf.