Asociación Colombiana de Facultades de ingeniería

Revista Educación en Ingeniería
http://www.educacioneningenieria.org

Dificultades de aprender a programar

Jorge Iván Fuentes-Rosado & Melquizedec Moo-Medina

Academia de Sistemas y Animación, Instituto Tecnológico Superior Progreso, Yucatán, México. jfuentes@itsprogreso.edu.mx, mmoo@itsprogreso.edu.mx

Resumen— Dentro de las competencias de cualquier ingeniero solicitadas por la industria, sin importar su área de especialidad, es la codificación en un lenguaje de programación. Un ingeniero debe ser capaz de codificar sus ideas ya sea para hacer experimentos y simulaciones de sus propuestas de solución, así como crear soluciones de software. El objetivo es determinar los obstáculos y clasificarlos para poder generar estrategias que faciliten el desarrollo de la competencia del desarrollo de software. El proyecto se realizó con estudiantes de segundo semestre de las ingenierías en electromecánica e ingeniería en sistemas computacionales. Se solicitó que los estudiantes describieran las diferentes problemáticas con las que se enfrentaban y cómo fueron resolviéndolas. Estas descripciones se utilizaron para determinar y clasificar los obstáculos que se les presentaron. Es interesante descubrir que aunque los estudiantes pertenecían a programas de ingeniería diferentes las problemáticas y obstáculos que tuvieron fueron muy similares.

Palabras Clave— Programación, ingeniería en sistemas, ingeniería en electromecánica, dificultades.

Recibido: 18 de agosto de 2016. Revisado: 17 de enero de 2017. Aceptado: 22 de junio de 2017

Difficulties from learning to computer programming

Abstract— Industries are in need of people able to code computer programs, no matter the specialty area. An engineer must be able to portray ideas, test them and create new software applications. The objective of this research is to determine the barriers and sort them in order to create strategies to help and facilitate the development of coding skills in students of different engineering programs. The project was done with students of the second semester of computer systems and electromechanical engineering programs. Students were asked to describe the different barriers they had faced and how they had solved that situation. These descriptions were classified. It is important to highlight that regardless of the students' background, they presented similar barriers.

Keywords— Computer programming, computational systems engineering, electromechanical engineering, barriers.

1. Antecedentes

Este artículo inicia parafraseando a dos grandes mentes del siglo XXI: “Aprender a escribir programas exprime tu mente, y ayuda a pensar mejor, crea una manera de pensar sobre cosas que creo es útil en todos los dominios”-Bill Gates [1]. “Nuestra política en Facebook es literalmente contratar tantos ingenieros talentosos que podamos encontrar. No hay suficiente gente entrenada y que tengan estas habilidades hoy”-Mark Zuckerberg [1]. En primera instancia Bill Gates, Fundador de Microsoft, menciona los beneficios que tiene el aprender a programar, incluso sin mencionar el área de ingeniería, Bill Gates dice que los beneficios de programar crea nuevas formar de pensar y solucionar problemas. Por su parte Mark Zuckerberg, fundador de Facebook, reclama la necesidad de buenos y talentosos programadores, haciendo notoria la falta de ellos.

De acuerdo al U.S. Bureau of Labor Statistics (BLS) para el 2024 los trabajos en Tecnologías de la Información (TI), en comparación a 2014, se incrementarán en un 17% [2], respecto a estas predicciones Mark Lassoff, presidente de learntoprogram.tv, dijo: “No hay suficientes personas para llenar estos puestos de trabajo, el mercado de trabajo se está moviendo más rápido que la capacitación en las universidades”. El reto que la universidad presenta actualmente es preparar a más gente a la velocidad que el mundo de las TI se mueve y requiere.

1.1. Habilidades para el desarrollo

La programación es parte esencial e integral de cualquier programa de ingeniería. Los estudiantes de ingeniería en sus últimos semestres tienen que enfrentar tareas de solución de problemas empleando esta competencia. El contar con buenas habilidades de desarrollo les ayudará a dar solución a estos problemas fácilmente. Es importante que los estudiantes de ingeniería y tecnología aprendan programación básica en sus primeros años de su preparación universitaria [3].

Aprender programación no es como adquirir otro conocimiento. No es un proceso algorítmico, es decir, no es como el cálculo diferencial donde se aprende el procedimiento o fórmula se aplica repetidas veces. No es de memorización, es decir, no es como aprenderse una lista de fechas importantes y repetirlas. Para aprender a programar no basta con aprender las palabras reservadas de un lenguaje para poder aplicarlo. Aprender a programar consiste en plasmar, mediante un lenguaje de programación, la forma de solucionar un problema. Cada problema se soluciona de manera distinta y cada programador lo resuelve de una forma diferente. Es allí donde

Como citar este artículo: Fuentes-Rosado, J.I. and Moo-Medina, M., Dificultades de aprender a programar. Educación en Ingeniería, 12(24), pp. 76-82, Julio, 2017.

radica la dificultad de aprender a programar; de tener un problema y crear una solución.

1.2. Temarios de estudio

En las dos ingenierías donde se desarrolló este proyecto, se lleva una materia de fundamentos de programación que se encuentra orientada al perfil. En ingeniería en sistemas, el objetivo de la asignatura es Analizar, diseñar y desarrollar soluciones de problemas reales utilizando algoritmos computacionales para implementarlos en un lenguaje de programación [4]. Por otra parte, el objetivo de Introducción a la programación en la ingeniería en Electromecánica es diseñar e implementar estrategias y programas para el control de los dispositivos en los sistemas electromecánicos. Diseñando interfaces gráficas con manipulación de puertos de computadoras, a través de lenguajes de programación [5]. Ambos temarios representan el primer acercamiento a la programación en ambas ingenierías, pero sus objetivos son totalmente diferentes, mientras que en ingeniería en sistemas se busca desarrollar el sentido analítico de los estudiantes, enfocándose más en el diseño y desarrollo; en electromecánica se busca la creación de interfaces gráficas de usuario que permita la comunicación con puertos de la computadora. En el desarrollo temático de la materia Fundamentos de programación de la Ingeniería en Sistemas Computacionales está compuesta por cinco unidades, la segunda unidad es donde el diseño y análisis inicia con los temas de diagramas de flujo y pseudocódigos, y la codificación en la unidades subsecuentes, siendo arreglos bidimensionales el último de los temas solicitados. En introducción a la programación de Ingeniería electromecánica está compuesto por cuatro unidades, no solicita el diseño mediante pseudocódigo o diagramas de flujo sino que inicia directamente con la codificación de soluciones, en la última unidad se solicita estudiar los temas relacionados con la comunicación de los puertos y las interfaces gráficas. En las dos ingenierías donde se desarrolló este proyecto, se lleva una materia de Fundamentos de Programación que se encuentra orientada al perfil. En la Ingeniería en Sistemas, el objetivo de la asignatura es analizar, diseñar y desarrollar soluciones de problemas reales utilizando algoritmos computacionales para implementarlos en un lenguaje de programación [4]. Por otra parte, el objetivo de Introducción a la Programación en la ingeniería en Electromecánica es diseñar e implementar estrategias y programas para el control de los dispositivos en los sistemas electromecánico, diseñando interfaces gráficas con manipulación de puertos de computadoras, a través de lenguajes de programación [5]. Ambos temarios representan el primer acercamiento a la programación en ambas ingenierías, pero sus objetivos son totalmente diferentes, mientras que en ingeniería en sistemas se busca desarrollar el sentido analítico de los estudiantes, enfocándose más en el diseño y desarrollo; en electromecánica se busca la creación de interfaces gráficas de usuario que permita la comunicación con puertos de la computadora. En el desarrollo temático de la materia Fundamentos de Programación de la Ingeniería en Sistemas Computacionales está compuesta por cinco unidades, la segunda unidad es donde el diseño y análisis inicia con los temas de diagramas de flujo y pseudocódigos, y la codificación en la unidades subsecuentes, siendo arreglos bidimensionales el último de los temas solicitados. En introducción a la programación de Ingeniería electromecánica está compuesto por cuatro unidades, no solicita el diseño mediante pseudocódigo o diagramas de flujo sino que inicia directamente con la codificación de soluciones, en la última unidad se solicita estudiar los temas relacionados con la comunicación de los puertos y las interfaces gráficas

1.3. Algoritmo para solucionar problemas en ingeniería

Todos los problemas que tienen solución tienen estrategias que pueden ser empleadas para solucionarlos. De manera general existen algoritmos para la solución de problemas. Como la que presenta Moore, en su libro, Matlab para Ingenieros [6]. Él plantea un algoritmo básico para la solución de problemas en las disciplinas de ingeniería, ciencias y programación. Es una serie de cinco pasos que inician desde el entendimiento del problema y termina con la validación de los resultados. El algoritmo es el siguiente:

1. Planteamiento del problema
   - Si el problema no está claro, es poco probable que se pueda resolver
   - Elaborar diagrama o bosquejos del problema
2. Establecer los valores de entrada y salida para verificar el algoritmo y la solución
   - Es importante incluir las unidades tanto de los valores de entrada y de salida.
   - Si hay constantes, incluirlas
3. Diseñar el algoritmo para resolver el problema. Desarrolla una prueba de escritorio.
   - Identificar ecuaciones que relacionen los valores conocidos como incógnitas.
   - Trabajar una versión simplificada del problema, a mano o con calculadora.
4. Resolver el problema.
   - Codificar el problema, empleando un lenguaje de programación.
5. Validar la solución.
   - ¿Los resultados tienen sentido?
   - ¿Los resultados del programa coinciden con los obtenidos a mano o con la calculadora?

En el desarrollo de software existe diversas formas crear soluciones de software. Una metodología de desarrollo consiste en una serie de fases o pasos por los que el problema transita, desde la obtención de requerimientos o necesidades de los usuarios hasta la liberación del software final. Una de estas metodologías es el modelo de cascada, que está compuesta por 5 fases [7]:

1. Requerimientos. Entender el problema a resolver, ser capaz de expresarlo en lenguaje formal sin ambigüedades.
2. Diseño. Diseño del algoritmo y Diseño de las pruebas
3. Análisis. Analizar la funcionalidad del algoritmo diseñado, probar en papel junto con el algoritmo y los datos de pruebas.
4. Codificación. Escribir en un lenguaje de programación el algoritmo
5. Pruebas. Realizar las pruebas pertinentes para asegurar su funcionalidad
6. Liberación. Entregar al cliente el software final. [7]

Ya sea con una metodología general o una metodología propia de la ingeniería de software es importante entender el

problema, describir la solución, implementarla y probarla. Esta estrategia básica de solución puede ser empleada o enseñada a los estudiantes al momento de analizar los problemas que se les presente al momento de codificar.

1.4. Objetivo

El objetivo del proyecto es determinar y clasificar las dificultades que tienen los estudiantes de las ingenierías en sistemas y electromecánica al momento de crear rutinas de software en la solución de problemas comunes para proponer estrategias que incrementen el aprovechamiento de los mismos estudiantes en las materias relacionadas al desarrollo de software.

1.5. Preguntas de investigación

Las preguntas de investigación que siguió este proyecto fueron:

- ¿Cuál es el proceso, de los estudiantes, empleado para generar rutinas de software?
- ¿Qué problemáticas presentan los estudiantes al momento de solucionar un problema que requiere de una rutina de software?
- ¿Cómo resolvieron los estudiantes las problemáticas presentadas?
- ¿Cómo afecta el aprovechamiento de la programación en las asignaturas futuras?

1.6. Justificación

Todo ingeniero debe programar. El mercado laboral exige que todo ingeniero tenga la competencia de crea aplicaciones, tal vez, no al grado de un ingeniero en software, pero sí que pueda realizar rutinas que le apoyen en su trabajo. La competencia del desarrollo de software se adquiere en los primeros semestres de las ingenierías; en ingeniería electromecánica en el primer semestre en la materia Introducción a la Programación; mientras que en ingeniería en sistemas en primer semestre en la materia Fundamentos de Programación y en Segundo Semestre en la materia Programación Orientada a Objetos.

El índice de reprobación en estas materias es elevado y se desconoce la causa fundamental. Entre las propuestas que el instituto ha intentado ha sido cambiar a los maestros que imparten la clase, teniendo los mismos resultados, alto grado de reprobación. Otra de las estrategias que el instituto ha implementado son los cursos de regularización o asesorías y los resultados no han variado. Un punto importante a observar es que aquellos alumnos que aprueban no cuentan con la competencia completamente desarrollada, incluso piensan en sólo aprobar la materia sin considerar aprender a programar.

2. Metodología

La hipótesis que se tuvo en este proyecto fue: Las dificultades de los estudiantes al aprender a programar pueden ser identificadas para poder generar y clasificarlas para generar estrategias que las mitiguen y se logre incrementar el aprovechamiento en las materias relacionadas con la programación al fortalecer dicha competencia.

Tabla 1
Muestra de participantes por carrera.

Entrevistados	Tamaño de la Muestra
Ing. En Sistemas Computacionales	21
Ing. En Electromecánica	24
Total	45

Fuente: Autoría propia

Este proyecto se llevó a cabo mediante un diseño no experimental transeccional descriptivo cuantitativo, aplicado a través de un solo examen al finalizar el primer semestre del año 2016 Febrero-Junio.

La muestra fue tomada de los segundos semestres de las ingenierías en sistemas computacionales, y electromecánica, todos los estudiantes que participaron en la encuesta únicamente han llevado el curso introductorio a la programación.

Los resultados obtenidos se pueden observar en la Tabla 1, donde el tamaño total de la muestra es de 45.

El proyecto se inició en el mes de Febrero de 2016, los estudiantes, todos ellos, de segundo semestre y habiendo aprobado la materia introductoria de programación.

Se tomaron el 100% de los estudiantes que asistieron a clases un día al azar.

El test aplicado contó con tres preguntas abiertas y seis problemas a resolver relacionados con los siguientes temas: Arreglos, Ciclos, Manipulación de Cadenas, Traducción de Fórmulas, Archivos. Las tres preguntas abiertas fueron: ingeniería que cursan, los lenguajes que han estudiado y el lenguaje que emplearían en la prueba. Algunos estudiantes, aquellos por el gusto nato para programar aprenden lenguajes de una manera autónoma, por lo que existía la posibilidad de resolver los problemas con el lenguaje que se sintieran más cómodos.

En la Tabla 2 se puede apreciar los reactivos que les fueron solicitados a los estudiantes, al igual de los conceptos que se requieren aplicar para solucionarlos.

Tabla 2
Ejemplo de reactivos
Fuente: Autoría propia

Enunciado del Problema	Conceptos
Realiza un programa que calcule el promedio de 10 número e imprima el resultado	Ciclos Promedio
Realiza un programa que invierta una cadena de texto introducida por el usuario	Manipulación de Cadenas Ciclos
Realiza un programa que ordene una lista de 10 números	Ciclos anidados Arreglos
Realiza un programa que calcule el resultado de la siguiente función $z = \frac{\sum_{i=1}^{5} x_i * \sum_{i=1}^{5} y_i - \sum_{i=1}^{5} x_i y_i}{\sum_{i=1}^{5} x_i^2 + \sum_{i=1}^{5} y_i^2}$ Fuente: Autoría propia	Ciclos Sumadores Potencias Traducción de fórmulas matemáticas Arreglos
Realiza un programa que calcule la transpuesta de una matriz de 3X4, la transpuesta de una matriz es convertir cada fila en columna	Matrices Ciclos
Realiza un programa que abra un archivo (input.txt) y elimine todas las vocales de él y guarde el resultado en otro archivo de salida (output.txt)	Ciclos Administración de Archivos Cadenas

Fuente: Autoría propia

Tabla 3
Lenguajes de programación utilizados

Ingeniería	C#	JAVA	C/C++
Sistemas Computacionales	1	21	0
Electromecánica	0	0	25

Fuente: Autoría propia

Tabla 4
Lenguaje a utilizar para la prueba

Ingeniería	C#	JAVA	C/C++
Sistemas Computacionales	0	21	0
Electromecánica	0	0	25

Fuente: Autoría propia

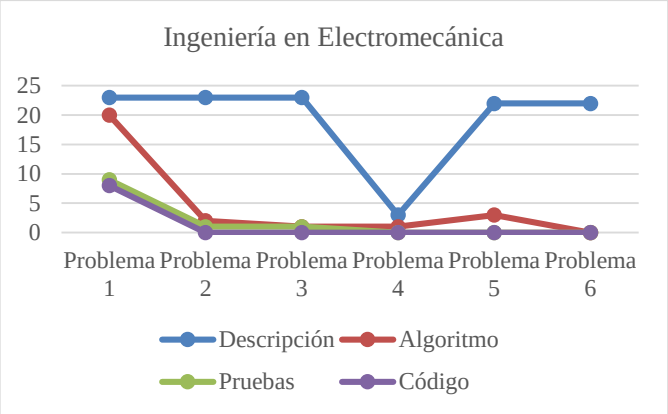

Figura 1. Resultados de la prueba aplicada a estudiantes de la Ingeniería en Electromecánica
Fuente: Autoría propia

Aplicando los algoritmos de solución de problemas en la ingeniería, cada programa solicitado debería estar acompañador de cuatro elementos, el primero de ellos fue la descripción personal, es decir, se solicitó a los estudiantes que escribieran con sus palabras cada uno de los problemas planteados, esto con el objetivo de validar si el problema fue entendido. Después de haber entendido el problema a resolver, se le pidió al estudiante realizar un algoritmo o diagrama de flujo que dé solución al problema planteado. El tercer elemento, los estudiantes debieron crear tres ejemplos de prueba para validar su solución y por último codificar la respuesta en el lenguaje de su preferencia. Con el objetivo de entender el proceso de solución, también se pidió a los estudiantes que escribieran los motivos por los cuales no podían realizar algunos de los apartados.

Esta prueba fue realizada en una sesión de tres horas consecutivas, con descansos cada que el estudiante lo requería. El uso de internet no fue permitido.

3. Discusión de resultados

En la encuesta se preguntó a los estudiantes que lenguajes han estudiado. En la Tabla 3 se muestra los resultados obtenidos.

No es de extrañarse que los estudiantes únicamente conozcan un lenguaje, sólo han aprobado un curso de programación y siguen utilizando el lenguaje que el profesor de la asignatura les enseñó. Los estudiantes de ingeniería en sistemas, presentan curiosidad en conocer otros lenguajes de programación y por ello, en este caso, uno de ellos ya iniciaba con la programación en C#.

No hubo mucha diferencia entre el lenguaje empleado en la solución de los problemas y los lenguajes de programación empleados. Los estudiantes expresaron que emplearían el lenguaje estudiado en su último curso. En la Tabla 4 se presentan los resultados estadísticos obtenidos

3.1. Resultados de ingeniería en electromecánica

En la ingeniería electromecánica, se resolvieron los siguientes programas. En la Fig. 1**¡Error! No se encuentra el origen de la referencia.** se observa los resultados por problema y por requerimiento de solución

La mayoría de los estudiantes de la ingeniería en electromecánica pudieron describir los ejercicios solicitados con sus propias palabras, es decir, entendieron el enunciado del problema. En sus diagramas o algoritmos se pudo notar que el concepto del ciclo no lo tienen entendido en su claridad, debido a que en el problema donde se les solicitó calcular el promedio de diez números, el algoritmo que prevaleció fue crear diez variables y leer cada una de ellas para realizar el cálculo solicitado. Dentro de los comentarios sobre el problema uno destacaba que era un ejemplo que ya habían realizado en clase.

Dentro de los comentarios observados y las soluciones inconclusas o erróneas, planteadas por los estudiantes se puede notar que los estudiantes, aunque entendían el problema no lograron crear un algoritmo para resolverlo, es decir, no lograban solucionar el problema. Otro de los problemas presentados por este grupo de estudiantes fue no lograr dividir el problema para solucionarlo, algunos estudiantes plantearon algoritmos de tres pasos para la solución de los problemas, por ejemplo, 1. Leer la cadena, 2. Invertir la cadena, 3. Imprimir resultados.

Los casos de validación fueron ignorados, los estudiantes olvidaban que tenían que ponerlo, y no reconocieron la importancia de ellos para realizar sus pruebas.

Aunque fueron pocos los estudiantes que llegaron a la codificación de la solución, y de únicamente dos problemas. Los códigos observados seguían fielmente los algoritmos planteados.

En el caso de la ingeniería en sistemas los resultados pueden ser observados en la Fig 2

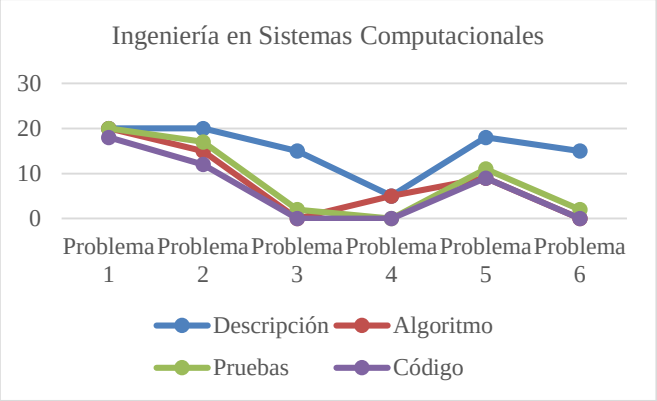

Figura 2. Resultados de la prueba aplicada a estudiantes de la Ingeniería en Sistemas Computacionales.
Fuente: Autoría propia

Las descripciones de los problemas no tuvieron dificultades, a excepción del problema cuatro, la traducción de la fórmula. Los estudiantes de la ingeniería en sistemas expresaron que era una fórmula compleja, que no conocían los símbolos o que no han estudiado cálculo integral. En cálculo integral se estudia el concepto de la sumatoria.

Lahtinen, Ala-Mutka y Järvinen establecen que la diferencia entre un programador experto y un programador novato es el grado de optimización y atajos que utiliza al momento de programar [8]. Dentro de los diseños de las soluciones a los problemas, al igual que los estudiantes de electromecánica, este grupo no tenía muy claro el uso de los ciclos. En el caso del problema cuatro, la fórmula, los diseños estaban basados en calcular cada uno de los elementos de manera individual con los cinco datos requeridos. Aunque la solución es correcta, el planteamiento muestra que es un programador novato. Sin embargo, la capacidad de un ingeniero debe ser solucionar problemas con las herramientas que tenga, y esta solución aunque no es la más elegante, es una solución práctica.

En el tema de la codificación de los problemas, los estudiantes externaron que el tema de administración de archivos no lo conocían. Atribuyo que no fue presentado en clase. Otro de los comentarios al momento de codificar fue el desconocimiento, los estudiantes, externan no conocer el lenguaje o como traducir sus ideas a código. Un comentario que llamó la atención del autor fue "No me acuerdo", este comentario se repitió en varios estudiantes, lo que me lleva a pensar que los estudiantes intentan aprenderse las rutinas, y no aprenden a desarrollarlas por si solos.

En forma general ningún estudiante logro terminar los seis problemas planteados, aunque los estudiantes de sistemas hicieron más codificación que los estudiantes de ingeniería electromecánica. También es de notar que el lenguaje estudiado por los estudiantes de sistemas es JAVA, mientras que el estudiante que estudiaron en electromecánica fue C/C++.

4. Conclusiones

A nivel global el desarrollo de nuevas tecnologías requiere de más desarrolladores de software, debido a que vivimos en un mundo altamente digital que sigue creciendo a pasos enormes. La demanda de aplicaciones de software dejó de ser una necesidad empresarial o industrial y se volvió parte de la vida diaria. Retos actuales como Internet de las Cosas hace que surjan nuevas necesidades de aplicaciones. Áreas como la salud requiere de software especializado, no sólo para la administración, sino aplicaciones que controlan los equipos de alta tecnología que realizan diagnósticos más oportunos o trasplantes electrónicos que dan una mayor calidad de vida a sus portadores. México requiere capacitar a más gente en disciplinas como el Desarrollo de Software e ingenierías tecnológicas para crear la tecnología que el país y el mundo necesita y convertirse es un productor de esta tecnología y no meramente consumidor.

Programar es una habilidad muy útil y puede ser una recompensante carrera. Programadores novatos sufren de un amplio rango de dificultades y deficiencias. Los estudiantes deben descubrir la necesidad de seguir un algoritmo para poder solucionar problemas. Es importante entender qué hay que hacer, establecer un plan de acción, establecer las pruebas de validación, codificar y por último hacer las pruebas. Los estudiantes no realizaban el diagrama de flujo porque no reconocen la importancia del diseño o el plan de acción. Los estudiantes directamente intentaban codificar sin haber entendido el problema o pensando en la solución. Es importante que el facilitador de la materia motive a los estudiantes a realizar los diseños previos al intento de codificación.

Dentro de los comentarios que los participantes escribían sobre las problemáticas que se les presentó durante el desarrollo de la prueba fueron clasificadas de la siguiente manera:

- Fobia a los problemas "complejos": "La fórmula es muy complicada de hacer. No he estudiado cálculo integral.". El estudiante antes de intentar solucionar el problema, en su primera percepción del problema, si ve algo que lo impacte, tiende a no solucionarlo. Se debe enseñar al estudiante de una manera práctica la estrategia divide y vencerás, la cual consiste en dividir el problema en pequeños subproblemas que pueda solucionar de una manera más sencilla.
- Lógica incompleta: Al no lograr dividir el problema en subproblemas no consiguen establecer los pasos necesarios para llegar a una solución completa. Los facilitadores podrían favorecer la lógica al sugerir la solución de acertijos y permitir que los estudiantes solucionen con tus recursos los problemas
- Desconocer el lenguaje: "No sé cómo leer archivos" Los estudiantes entienden el problema, saber cómo solucionarlo pero no logran codificarlo debido a que ignoran las palabras reservadas o librerías del lenguaje que podrían aplicar para realizar el software correspondiente. A manera de estrategia, los estudiantes podrían crear sus propias guías de bolsillo con las palabras reservadas y funciones que emplean.
- Desconocer las herramientas del entorno de desarrollo (IDE): "No corre como yo esperaba y no pude encontrar el error". Los estudiantes escribían sus códigos pero no lograban corregir los errores de lógica que se les presentaban en el código. Los errores de sintaxis eran eliminados por el IDE, es decir, no hubieron errores tales como olvidar los puntos y comas, la falta de declaración de variables, o los tipos de variables. Si los estudiantes supieran como realizar un debuggeo, correr el programa paso a paso empleando el entorno de programación, hubieran podido encontrar los errores faltantes.
- Falta de motivación: "No sé hacerlo". Aunque los estudiantes reconocen que el desarrollo de software es importante, hay quienes expresan que no tiene el gusto en realizarlo. El estudiante debe reconocer que una de las ventajas de programar es el desarrollo profesional. Un estudiante desmotivado no realizará las prácticas, y lamentablemente para aprender a programar hay que programar.
- Administración del tiempo: "No me alcanzó el tiempo". Los estudiantes dentro de sus comentarios expresaron que el tiempo no fue suficiente para resolver el examen. Para un trabajo futuro se considerará dar más tiempo para la solución.

La programación en el caso de sistemas computacionales está ligada con las materias siguientes: Programación Orientada a Objetos, Métodos Numéricos, Simulación, Estructura de Datos, Tópicos avanzados de programación, bases de datos, ingeniería de software. Si el estudiante no desarrolla las

competencias debidas se verá afectado en el aprovechamiento de las materias subsecuentes. La ingeniería en electromecánica en el plan de estudios actual no cuenta con ligaduras para la materia de fundamentos de programación, además el autor sugiere más cursos de programación para esta ingeniería, para que los estudiantes logren desarrollar la lógica requerida.

Referencias

[1] CODE, Leaders and trend-setters all agree on one thing, 2015. [En línea]. Available at: https://code.org/quotes.

[2] United States Deparment of Labor, Software Developers, 15 04 2016. [En línea]. Available at: http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm.

[3] Sun, W. and Sun, X., Teaching computer programming skills to engineering and technology students with a modular programming strategy. American Society for Engineering Education, 2011.

[4] Tecnológico Nacional de México, Temario de Fundamentos de Programación Ingeniería en Sistemas, México, 2010.

[5] Tecnológico Nacional de México, Temario de Introducción a la Programación Ingeniería Electromecánica, Mexicali, 2010.

[6] Moore, H., Matlab para ingenieros, Pearson Educación, 2007.

[7] Sommerville, I. y Alfonso, M., Ingeniería de Software, Pearson Educacion, 2009.

[8] Lahtinen, E., Ala-Mutka, K. and Järvinen, H., A study of the difficulties of novice programmers, SIGCSE Bull, 2005, pp. 14-18.

[9] Ko, A., Myers, B. and Aung, H., Six learning barriers in end-user programming systems. In: Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing, 2004.

J.I. Fuentes-Rosado, recibió el título de Lic. en Ciencias de la Computación en 2006 de la Universidad Autónoma de Yucatán en México, el título de MSc. en Ciencias con Especialidad en Sistemas Inteligentes en 2008 del Instituto Tecnológico de Estudios Superiores de Monterrey, México. Inició sus labores docentes en el Instituto Tecnológico Superior Progreso en agosto de 2010, y es profesor de tiempo completo de asignatura C desde 2012. Ha fungido como líder de la línea de investigación "Desarrollo de Tecnologías de la Información y Comunicación". Sus intereses investigativos incluyen: aprendizaje máquina para procesos de optimización, innovación educativa, gráficos por computadora, procesos automáticos de optimización de código.
ORCID: 0000-0002-3079-2323

M. Moo-Medina, recibe el titulo Ing. en Sistemas Computacionales en 2003 por el Instituto Tecnológico de Mérida en Yucatán, México y en 2014, el título de MSc. en Tecnologías de Información por la Universidad Interamericana para el Desarrollo. Ha trabajado como líder de proyectos en Tecnologías de la Información para Infraestructura de la Red del Instituto Tecnológico Superior Progreso y como líder de Proyecto para el desarrollo de laboratorio de Diseño y Animación Digital en el 2014 y 2015. Actualmente realiza proyectos de investigación enfocados al desarrollo de software en diferentes áreas de trabajo y a las publicaciones de artículos desde el Instituto Tecnológico Superior Progreso, desempeñando el puesto de profesor de tiempo completo e investigador.
ORCID: 0000-0003-3578-862X