“Estudio de la problemática presente en el diseño de algoritmos por computadora”

M. en C. Isaías Pérez Pérez
M. en C. América Fuentes Gálvez
M.C.C. Silvia Soledad Moreno Gutiérrez

Instituto de Ciencias Básicas e Ingeniería
Escuela Superior Tlahuelilpan
Universidad Autónoma del Estado de Hidalgo
Tel. Cel. 044-771-114-8569
e-mail: isaiaspp7@hotmail.com

Área de conocimiento:
Programación de computadoras-Computación Educativa

RESUMEN

Es conocido que en la mayoría de los estudiantes, su iniciación en los temas sobre programación de computadoras suele ser regularmente, problemática (Joyanes Aguilar, 1991, p. XI). Esto se debe principalmente a que no entienden la definición de los problemas que pretenden solucionar (Lozano R., 1992, p. 13), y en segundo lugar, su incapacidad de poder identificar de manera eficiente los elementos implicados para su solución (entradas-procesos-salidas). El presente trabajo presenta los avances alcanzados del estudio que fue presentado inicialmente en abril de 2008, titulado: *“Estudio sobre la problemática en los enunciados de los problemas de programación”*, sustentado en la aplicación y análisis de un instrumento de evaluación, previamente diseñado, en estudiantes universitarios de la asignatura de programación, con el fin de identificar en que medida éstos poseen los elementos conceptuales necesarios para el diseño de algoritmos de computadora, así como en los errores de diseño que frecuentemente incurren.

PALABRAS CLAVE:
Programación, problemática, algoritmo, diseño, método.

INTRODUCCIÓN

En la enseñanza de las asignaturas que abordan el diseño de algoritmos para computadora, es un problema muy conocido en los estudiantes, la gran dificultad de que lleven y logren finalmente, el desarrollo adecuado y correcto de algoritmos de diversa índole. Esto se debe principalmente a dos razones: la primera, a que dicho proceso de diseño no es una labor sencilla, y segunda, a que los estudiantes de estos temas no poseen o tienen malos hábitos para el desarrollo de esta tarea.

Para atacar en gran medida esta problemática, desde los años 70’s, surgieron las técnicas de la programación estructurada desarrolladas por Wirth, Dijkstra, Hoare, Bohm y Jacopini, entre otros, y que actualmente se recogen y aplican en lo que se conoce como el “Método de Resolución de Problemas por Computadora” (Joyanes Aguilar, 1993, p. XV).

ESTADO DEL ARTE Y PROBLEMÁTICA AFRONTADA

La enseñanza de las asignaturas de programación, actualmente es un tema obligado en los cursos universitarios relativos a las TIC's. Es de vital importancia su estudio, que reconocidos autores, han identificado el crucial papel que juegan estos temas en la educación de las ciencias computacionales; tal es el caso de Luis Joyanes Aguilar (1993), reconocido profesor y autor español, cuando menciona:

"Para el estudiante universitario, las asignaturas de programación se convierten en asignaturas clave en su formación, cuyo éxito o fracaso influirá decisivamente en el resto de sus estudios. La vital importancia de ésta asignatura obliga a un replanteamiento y estudio profundo en el que deberán intervenir esencialmente las modernas técnicas de programación y las características mas notables de los lenguajes de programación" (Joyanes Aguilar, 1993, p. XV).

Las modernas técnicas de programación se ven hoy reflejadas primordialmente en el "Método de Resolución de Problemas por Computadora" (Joyanes Aguilar, 1993 y Watkins, 1984), el cual consta de 3 fases principales: entrada de datos, proceso (algoritmo) y salida de resultados, visualizando a la definición de algoritmos como el paradigma clásico de un sistema.

El citado método consta de las siguientes fases:

1ra. Fase:	Análisis del problema
2da. Fase:	Diseño del algoritmo
3ra. Fase:	Resolución del problema con computadora

La primera de ellas, el análisis del problema, trata sobre el definir y acotar adecuadamente el problema a resolver, en correspondencia con una solución que se puede construir por computadora; a continuación se procede a definir las entradas y salidas que deberá tener el algoritmo a diseñar.

La segunda fase, el diseño del algoritmo, aborda la definición de los elementos particulares que deberá llevar el proceso del algoritmo; esto se realiza inicialmente como una jerarquización de los pasos del algoritmo a diseñar (Diseño Descendente); después se trata de representar de manera más detallada cada una de las instrucciones o pasos del algoritmo, de manera parecida al pseudocódigo (Refinamiento de los Subproblemas). Finalmente, se utiliza alguna de las técnicas de diagramación de algoritmos, para hacer una representación grafica de él, con la finalidad de lograr un mejor entendimiento y una adecuada documentación del mismo (Representación de Algoritmos).

La tercera fase, resolución del problema con computadora, trata sobre la codificación del algoritmo en un lenguaje de programación específico, su compilación y ejecución, así como de la depuración de los errores que pudiese tener. Esta fase es la que construye un producto de software como tal.

El presente estudio aborda de manera inicial, la segunda etapa del método mencionado, debido a que en el artículo titulado: *"Estudio sobre la problemática en los enunciados de los problemas de programación"*, presentado en abril de 2008, se abordo aspectos relacionados con la primera etapa del método citado. Entre los problemas que se presentan en esta segunda fase, se puede destacar lo que menciona Letvin Lozano (1992), autor colombiano, al respecto:

"Los estudiantes creen que en llevar a cabo las técnicas de representación de algoritmos (diagramas de flujo, pseudo código, tablas N-S, etc.) radican todos sus problemas, pero en realidad el hecho es que no entienden correctamente los enunciados de los problemas a los que se enfrentan" (Lozano R., 1992, p.13).

Tomando como hipótesis del estudio, lo mencionado por Lozano (1992), se procedió a llevar a cabo una investigación de campo, utilizando un test de evaluación escrito, basado en el método de resolución de problemas por computadora, en su segunda fase (diseño del algoritmo), el cual fue aplicado a estudiantes de las materias de programación, de las carreras de Sistemas Computacionales y de Ingeniería Industrial, impartidas en el Instituto de Ciencias Básicas e Ingeniería, perteneciente a la Universidad Autónoma del Estado de Hidalgo. En el citado test se les proporcionó a los estudiantes el planteamiento del problema, la entrada de datos, salida de resultados, así como el tipo de datos a utilizar en el diseño del algoritmo, con el fin de conocer en que medida los estudiantes logran el diseño de algoritmos satisfactorios y correctos.

METODOLOGÍA O TÉCNICA USADA

La metodología del estudio es la siguiente:

- Selección de dos problemas tipo a plantear.
- Desarrollo del test de evaluación, presentando los dos problemas previamente escogidos
- Aplicación del test a grupos de estudiantes de las asignaturas de programación
- Recolección de los resultados obtenidos en el test
- Análisis y generación de las conclusiones, derivadas de los resultados obtenidos

Los dos problemas tipo se escogieron en base a dos enfoques específicos: el primero (problema A), presenta un enfoque didáctico tradicional, ya que es un programa con características propias de un proceso de cálculo aritmético (dado el radio de una esfera, calcular su área y volumen). Este estilo de problemas generan estructuras de diseño de algoritmos que los estudiantes generalmente memorizan, ya que siempre presentan el mismo arreglo: a) entrada de datos; b) proceso, el cual consta por lo regular, de una serie de operaciones aritméticas interrelacionadas; y c) salida de resultados. Hay que aclarar que en este tipo de algoritmos, por lo general, las entradas se diferencian notablemente de las salidas. Por otro lado, el segundo problema (problema B), el cual propone el diseñar un algoritmo que intercambie los valores de tres variables numéricas, es también un tipo de problema conocido en la enseñanza de estos temas, pero la construcción del proceso del algoritmo, requiere de dos aspectos principales: a) las entradas y salidas implican usar las mismas variables numéricas, lo cual produce una confusión al identificar las entradas de las salidas; y, b) el intercambio de los valores entre las variables, requiere de una compresión bastante precisa de este proceso, ya que se tiende a presuponer la solución de dicha operación, sin estar seguro de los detalles a considerar (por ejemplo, el de perder uno de los valores de las variables usadas).

En cada uno de los problemas presentados, se le presenta al estudiante las tres etapas que comprende la fase de diseño de algoritmos: a) Diseño Descendente, el cual se aborda mostrando una lista, en donde el estudiante selecciona la secuencia de subproblemas, expresados de manera verbal, y que le permitan diseñar más adelante el algoritmo; b) Refinamiento de los Subproblemas, en donde se muestra una lista en donde el estudiante, de nueva cuenta, selecciona la secuencia de subproblemas, expresados ahora de forma más abstracta o simbólica (de manera aritmética o algebraica), y que se utilizaran para el diseño más detallado del algoritmo; y c) Representación de Algoritmos, en el cual se solicita que el estudiante represente el algoritmo, por alguna técnica como el diagrama de flujo, el pseudocódigo, las tablas N-S u otro que considere adecuado, basándose en la secuencia de subproblemas seleccionados en los dos pasos anteriores. Este último paso obliga al estudiante llevar a cabo una abstracción mucho más profunda del algoritmo a diseñar.

RESULTADOS EXPERIMENTALES

La población de estudio a la que se le aplicó el mencionado test de evaluación desarrollado, fue de 66 estudiantes de la asignatura de programación estructurada.

En cuanto a la recolección de los resultados obtenidos en el test, esta se llevo a cabo a través de la construcción de una matriz, donde se evaluaban cada uno de los 66 estudiantes encuestados, en cada de los tres pasos de la fase de diseño de algoritmos: Diseño Descendente (DD), Refinamiento de los Subproblemas (RS) y Representación de Algoritmos (RA), registrando si habían acertado o en que habían fallado. La figura 1 muestra los resultados concentrados.

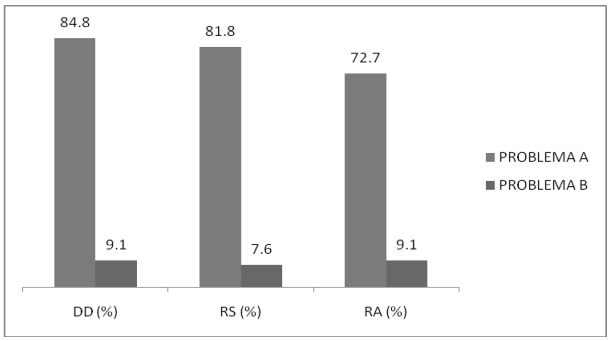

Fig.1. Porcentajes de aciertos de los estudiantes encuestados, en cada uno de los pasos de la etapa de diseño de algoritmos

Aunque es evidente la diferencia en la cantidad de aciertos entre ambos problemas (aprox. 70%), lo cual se debe principalmente, a los estilos diferentes de ambos problemas, lo cual produce estas diferencias tan marcadas; además, se puede observar que existe una tendencia de los aciertos a la baja, según se va aumentando el grado de abstracción en el diseño de los algoritmos, lo que indica que al avanzar cada vez en esta fase, la

construcción de los algoritmos presenta más errores (ver figura 2).

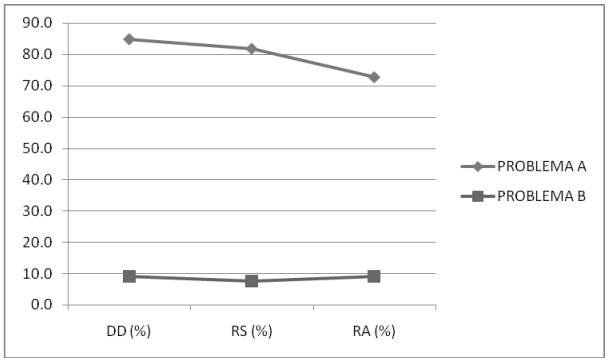

Fig. 2. Tendencias en porcentajes del nivel de aciertos de los estudiantes encuestados, en cada uno de los pasos de la etapa de diseño de algoritmos

En cuanto a los porcentajes de errores, cometidos por los estudiantes en cada uno de los pasos de la fase de diseño de algoritmos, siendo aproximadamente para el Diseño Descendente de 15%, para el Refinamiento de los Subproblemas de 18% y para la Representación de Algoritmos de 27%, hay que decir que cada uno de estos pasos, pretende definir de manera cada vez más abstracta el cuerpo del algoritmo a diseñar; por tanto, en cada uno de estos intentos, se puede observar tres subpartes constituyentes: la entrada de datos, el proceso y la salida de resultados. Esto permitió clasificar en porcentajes los errores que se cometían en cada una de estas subpartes, constituyentes de cada paso del diseño del algoritmo. La grafica 3 muestra las tendencias encontradas.

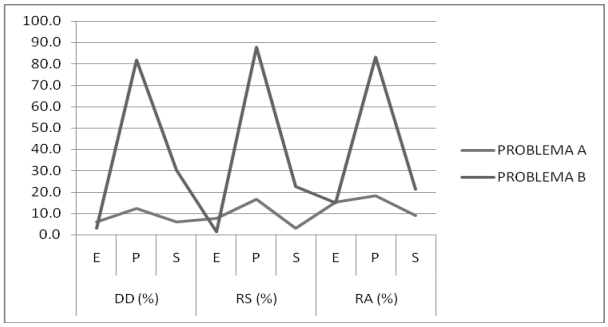

Fig. 3. Porcentajes de errores cometidos por los estudiantes encuestados, en cada uno de los subpartes de cada uno de los pasos del diseño de algoritmos

Es evidente notar que el mayor porcentaje de errores cometidos por los estudiantes, al tratar de definir un algoritmo de un problema especifico, es sin lugar a dudas el diseño del proceso del algoritmo (ver figura 4), seguido por los errores al definir las salidas de resultados y por último, los errores cometidos con respecto a la entrada de los datos. Esto es debido a que el proceso afecta de manera natural a la salida de los resultados, lo cual no sucede con la entrada de datos, siendo, al contrario, esta la que afecta al proceso.

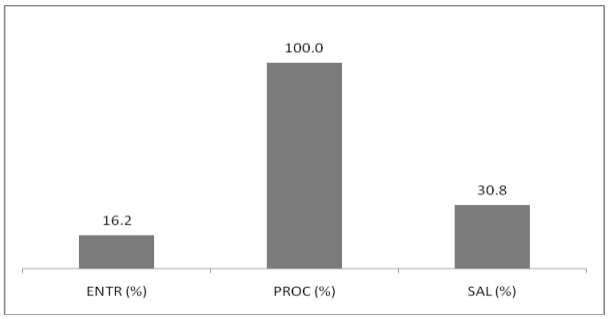

Fig. 4. Comparativa del nivel promedio de errores encontrados en los pasos del diseño de algoritmos, cometidos por los estudiantes encuestados

Finalmente, el hecho de que los estudiantes tienen problemas con el diseño de algoritmos, debido a tienen dificultades con la representación de los mismos, puede desmentirse de una manera categórica, al ver que los estudiantes presentan altos índices de dominio de estas técnicas de representación, como son el diagrama de flujo, el pseudocódigo y las tablas N-S (ver figura 5), además de que siguen adecuadamente en más de un 70%, las reglas establecidas para ellas (ver figura 6). Este dominio de las técnicas no parece ayudar a disminuir el problema que presentan los estudiantes, de no entender correctamente los enunciados de los problemas a los que se enfrentan. Se puede concluir de forma experimental, que la aseveración de Letvin Lozano (1992), es acertada en grado importante.

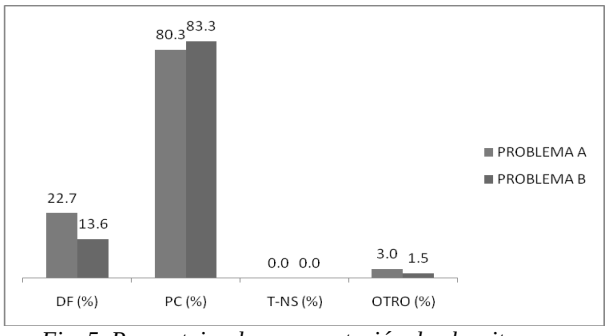

Fig. 5. Porcentajes de representación de algoritmos, por diversas técnicas de representación de algoritmos

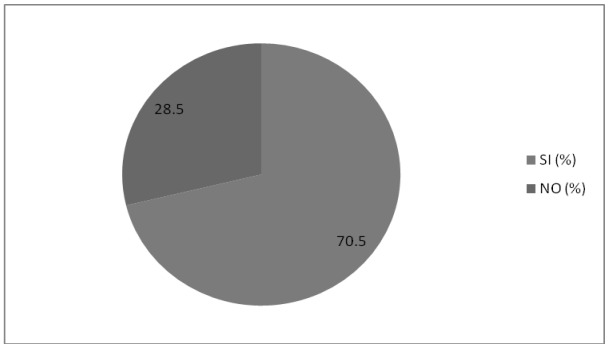

Fig. 6. Porcentaje en que se siguen las reglas de las técnicas de representación de algoritmos, por parte de los estudiantes

CONCLUSIONES Y TRABAJOS FUTUROS DE INVESTIGACIÓN

La primera conclusión que se puede apreciar, después de analizar los resultados obtenidos en el test, es que los estudiantes parecen haber aprendido "de memoria", el diseño de algoritmos comunes, que son enseñados como problemas ejemplo en las clases de las asignaturas de programación. Al aprenderse la estructura de la solución de estos problemas modelo, se pueden reproducir en problemas parecidos, lo cual inhibe la aplicación del verdadero diseño de los algoritmos, problema muy común, cuando los estudiantes se enfrentan al desarrollo de un programa, cuya estructura es diferente a las que conoce.

Es importante observar tres aspectos que han sido revelados por la presente investigación: primero, según se va aumentando el grado de abstracción en el diseño de los algoritmos, la construcción de los mismos presenta más errores; segundo, al parecer el mayor porcentaje de errores cometidos por los estudiantes, al tratar de definir un algoritmo, es el diseño del proceso del algoritmo, seguido por los errores al definir las salidas de resultados y por último, los errores cometidos con respecto a la entrada de los datos; y tercero, el conocimiento y aplicación adecuada de las técnicas de representación de algoritmos, no parece ayudar sustancialmente a resolver el problema que presentan los estudiantes, sobre su ineficiente capacidad para desarrollar algoritmos que reflejen correctamente las soluciones solicitadas.

Lo antes mencionado, puede converger en una conclusión importante: la aplicación del Método de Resolución de Problemas por Computadora, en sus dos primeras etapas (análisis del problema y diseño del algoritmo), parece carecer de algunas mejoras en sus procedimientos, técnicas y herramientas, que le permitan al que lo aplique, lograr avanzar de forma positiva en el nivel de abstracción del diseño de algoritmos; esto se vera reflejado de manera principal en el adecuado y correcto diseño de su proceso, así como su correcta articulación con las entradas de datos y salidas de resultados. Es necesario lograr enriquecer de manera teórica y practica los procedimientos, técnicas y herramientas del actual Método de Resolución de Problemas por Computadora, por lo menos en sus dos primeras etapas; es posible que para lograr ello, sea necesario generar una nueva propuesta de éste método, que recoja las mejoras necesarias sobre los aspectos relevantes en el diseño de algoritmos. Finalmente, hay que señalar que las técnicas de representación de algoritmos son sólo parte de la compleja estructura metodológica pendiente por definir y ajustar en futuras investigaciones.

REFERENCIAS

Lozano, R., L. (1992). *"Diagramación y programación estructurada y libre"*. Editorial Mc Graw-Hill. Tercera edición. México.

Joyanes Aguilar, L. (1991). *"Metodología de la programación. Diagramas de flujo, algoritmos y programación estructurada"*. Editorial Mc Graw-Hill. Primera edición. México.

Joyanes Aguilar, L. (1993). "*Fundamentos de programación. Algoritmos y estructuras de datos"*. Editorial Mc Graw-Hill. Primera edición. México.

Watkins, R. (1984). *"Solución de problemas por medio de computadoras"*. Editorial LIMUSA. Primera reimpresión.México.