PROPUESTA METODOLÓGICA PARA DESARROLLAR UN PROGRAMA CON PROGRAMACIÓN ESTRUCTURADA A PARTIR DEL PARADIGMA FUNCIONAL

A METHODOLOGICAL PROPOSAL TO DEVELOP STRUCTURED BASED PROGRAMS USING FUNCTIONAL PARADIGM

Omar Iván Trejos Buriticá
Ingeniero de Sistemas, Ph.D,, Profesor Titular, Facultad de Ingenierías,
Investigador Grupo Informática, Universidad Tecnológica de Pereira, Pereira, Colombia
omartrejos@utp.edu.co

Fecha de recepción: 11 de marzo de 2013
Fecha de aprobación: 21 de enero de 2014

RESUMEN

El presente artículo es producto de un Proyecto de Investigación[1] desarrollado en el área de la programación de computadores y su relación con los procesos de aprendizaje que se involucran alrededor de ella. En este artículo se plantea una metodología para el desarrollo de programas que se basa en la programación estructurada, pero tomando como base los conceptos centrales del Paradigma Funcional para su desarrollo. Se busca con esta propuesta metodológica enfrentar y solucionar los tres grandes problemas de la programación e igualmente, se persigue la simplificación de la programación estructurada a partir de la utilización de funciones. Se acude a una concepción general del programa a realizar y se va desglosando poco a poco, al tiempo que se construyen funciones que cumplan con los micro-objetivos planteados. Se ha encontrado que, para los estudiantes, es una forma muy sencilla y simplificada de concebir tanto el paradigma estructurado como el paradigma funcional y que su relación facilita la comprensión de los conceptos asociados a dichos paradigmas.

Palabras Clave. Aprendizaje, metodología, programación estructurada, programación funcional.

[1] Proyecto de Investigación "Desarrollo de contenidos y metodología para un curso de Introducción a la Programación basado en el paradigma de Programación Funcional para estudiantes de primeros semestres de Ingenierías utilizando actividades y técnicas de Active Learning" aprobado por el Consejo de la Facultad de Ingenierías e inscrito en la Vicerrectoría de Investigaciones, Innovación y Extensión de la Universidad Tecnológica de Pereira Código 6-12-14.

ABSTRACT

This article is a product of a research Project developed in the Computer Programming area and its relation with learning process included in it. You can find a methodology for developing programs based on structured programming but using core concepts from functional paradigm. We try to solve the three fundamental problems of computer programming and to simplify of the structured programming using functions as the central concepts. We use a general overview of the program to develop and we build functions each time you can find a micro-objective to develop. For the students, this methodology is a very easy way to understand both structured and functional programming paradigms and their relation simplify the understanding of the concepts of these paradigms.

Keywords. Learning, methodology, Structured Programming, Functional Programming.

1. INTRODUCCIÓN

El estudio de la programación de computadores y el respectivo desarrollo de contenidos que posibiliten el aprendizaje se ha constituido, con el paso del tiempo, en uno de los caminos a través de los cuales se madura la calidad formativa en los programas de Ingeniería de Sistemas. En este sentido se hace recurrente el estudio y profundización del paradigma estructurado de programación, no solo porque ha sido uno de los paradigmas sobre los cuales se ha basado gran parte del desarrollo de la industria del software, sino porque ha sido el que posiblemente más ha avanzado en cuanto a formas de representación, definición de fronteras y construcción de aplicaciones.

Sin quitarle mérito a los demás paradigmas que sirven como base para el desarrollo de aplicaciones de software, el paradigma estructurado pareciera seguir siendo un paso obligado en el proceso de apropiación, asimilación y aplicación de los conceptos que giran alrededor de la programación de computadores. Por su parte, el paradigma funcional hace un aporte muy importante en el pensamiento, asociado a la programación de computadores y tiene que ver con la relevancia que le concede a la organización estructural de un programa a partir del concepto de función.

Una función es el núcleo de trabajo del paradigma funcional y su utilización, aplicación y depuración que facilitan enfrentar los tres grandes problemas que se presentan al momento de construir un programa y que se explican en un numeral posterior de este artículo. Este artículo se justifica desde la óptica de buscar mejores caminos para simplificar la aplicación del paradigma estructurado en el desarrollo de programas de computador, al tiempo de establecer relación con el paradigma funcional y capitalizar los conceptos centrales del paradigma funcional para que la programación estructurada se haga más aplicable, asimilable y evaluable.

El problema a resolver radica en la búsqueda de caminos que, sin perder la esencia de un paradigma como el estructurado, facilite su aprendizaje así sea a partir de la apropiación y aplicación de conceptos de otros paradigmas. La enseñanza de la programación y más importante, el aprendizaje de la misma pareciera que fueran por caminos diferentes y no siempre el refinamiento de aquella ha implicado el mejoramiento de ésta. Por tal razón, se quiere con este artículo, proponer uno de esos caminos que enfrente y resuelva el problema planteado.

Alrededor del tema de la programación de computadores se han escrito una muy buena cantidad de artículos y libros; pocos de ellos han buscado caminos de simplificación de los paradigmas asociados a la programación y muy pocos han intentado establecer relaciones entre los paradigmas de manera que uno de ellos ayude a la simplificación de otro. Autores como Van Roy [11], Joyannes, Trejos, Schildt, Jamsa y Kernigham [6] se han distinguido por el refinamiento de textos alrededor de la programación dentro del marco de un determinado paradigma. Lo innovador de este artículo es que se propone un camino que no solo simplifica el paradigma estructurado y al tiempo el paradigma funcional, sino que permite que el funcional sirva de base para el desarrollo de programas construidos bajo el paradigma estructurado.

Pareciera obvio que el estudio y profundización de un área del conocimiento implicara una alta aproximación al modelo matemático que le subyace, a sus fuentes conceptuales y a los elementos de juicio que de él se derivan [7]; por esta razón, este artículo acude a la relación entre los dos elementos que distinguen conceptualmente los paradigmas citados. En cuanto al paradigma estructurado se acude a las estructuras básicas y en cuanto al paradigma funcional, se acude al concepto de función.

En este artículo, no se aborda la aproximación matemática dado que lo que se propone es una metodología que simplifique lo aplicativo en relación con el paradigma estructurado a partir del paradigma funcional; aunque no se descarta que en próximos artículos se planteen relaciones entre los modelos matemáticos que subyacen a cada uno, entre sus fuentes conceptuales y entre los elementos de juicio que se derivan de ambos paradigmas.

Para el desarrollo de este artículo debió acudirse a la formulación de los tres grandes problemas de programación, las teorías de aprendizaje significativo [2] y aprendizaje por descubrimiento [4] así como a los planteamientos fundamentales de los paradigmas estructurado y funcional.

De la misma manera, se acudió a la experiencia del autor de este artículo y a su búsqueda permanente de caminos que simplifiquen el aprendizaje de la programación como producto de su formación Doctoral y de su experiencia como autor de 10 libros de programación y varios artículos de investigación científica. Igualmente se acudió a la voluntad de los estudiantes de aceptar y apropiar la metodología propuesta, de

permitir el monitoreo del avance de dicha metodología y de compartir sus inquietudes.

El método utilizado en la puesta en escena de esta propuesta metodológica consistió en el planteamiento de enunciados, luego de cubrir teóricamente los temas pertinentes, y desarrollarlos en presencia de los estudiantes, algunas veces bajo el uso de instrumentos tradicionales (tablero y marcador) y otras veces con la utilización de computador y videobeam como instrumentos modernos pero siempre bajo la misma línea metodológica. La metodología fue aplicada en los grupos de las asignaturas Programación I (programación funcional) y Programación II (programación estructurada) de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira desde el I semestre de 2011 hasta el I semestre de 2013 y se sigue aplicando.

El autor de este artículo parte de cuatro hipótesis para la formulación de la presente propuesta metodológica: a) Siempre es posible encontrar caminos que relacionen el conocimiento previo con el nuevo conocimiento dentro del área de la programación de computadores. b) Todo paradigma de programación puede apoyarse en otro que simplifique su comprensión. c) La programación es un área simple y sencilla SOLO sí así lo quiere presentar el Docente. d) Si el estudiante apropia plenamente el concepto de función, tendrá dominado por completo tres paradigmas: El paradigma funcional, el paradigma estructurado y el paradigma orientado a objetos. Estas son hipótesis originales del autor, basadas eso sí, en estudios de las teorías de aprendizaje significativo y aprendizaje por descubrimiento.

Para finalizar, esta introducción se plantea que el presente artículo está organizado bajo el formato internacional IMRYD y por ello se inicia con una introducción que presenta las características generales del artículo, se presenta la teoría que sobre la cual se fundamenta, se comparte la metodología tanto en su formulación como en su aplicación, se presentan unos resultados, se formulan algunos elementos de juicio para nutrir la discusión y finalmente, se plantean algunas conclusiones y el soporte bibliográfico que sirvió de base para este artículo.

2. TEORÍA

2.1. Aprendizaje Significativo

Bajo el nombre de aprendizaje significativo se conoce la teoría formulada por el Dr. David Paul Ausubel, quien planteó la gran importancia que tiene el significado de lo que se aprende dentro de un proceso de aprendizaje. Se llama Aprendizaje Significativo porque su fundamento es el concepto de significado, es decir, ¿Para qué sirve determinado conocimiento? La búsqueda de significado es una de las habilidades cognitivas de alto nivel innatas para el cerebro; cualquier evento, símbolo o situación que no sea claro por el cerebro inmediatamente genera una reacción de éste en la búsqueda de lo que significa, bien a través de los patrones que tenga almacenados, bien a través de la información útil que haya recibido a través de los sentidos o bien, a través de

la información simple (de uso poco frecuente) que haya guardado en la memoria a corto plazo [1].

De acuerdo a esto, el significado se convierte en el norte que permite que la información que llega al cerebro a través de los sentidos adopte una condición de patrón, información útil o información latente [2]. Un patrón es un modelo informacional (o de conocimiento) que se usa con frecuencia y que llega a orientar ciertas decisiones del comportamiento humano basado en su significado. La información útil constituye ese conjunto de conocimientos que se usan con alta frecuencia y que, normalmente, tiende a ser un patrón sin serlo. La información latente es la información que llega a través de los sentidos y que el cerebro aún no ha clasificado como información útil, bien porque no le ha encontrado significado o bien porque no es de uso frecuente.

El Dr. Ausubel formula en su teoría que el aprendizaje se basa en tres fundamentos: el conocimiento previo, el nuevo conocimiento y la actitud del estudiante. El conocimiento previo es el conjunto de saberes que el estudiante tiene cuando se inicia un nuevo proceso formal de aprendizaje, como cuando se inicia un curso. La teoría del aprendizaje significativo establece que siempre que el ser humano se enfrenta a un proceso de aprendizaje, existe un conjunto de conocimientos que pueden considerarse como conocimiento previo, o sea, lo que el ser humano ya sabe antes de empezar a aprender. De acuerdo a esto, el Dr. Ausubel establece que "Si me preguntaran qué es lo más importante en el aprendizaje yo diría que es lo que el alumno ya sabe" [2] y con esto le está dando alta prioridad al conocimiento previo adquirido antes del inicio de un proceso de aprendizaje.

Según esto, el nuevo conocimiento lo constituye el conjunto de saberes que el alumno aún no había tenido oportunidad de acceder a ellos, que aún no se había formalizado desde alguna de las ciencias o que aún no se había difundido. Para el alumno puede considerarse como nuevo e innovador todo aquello que a un determinado alumno no haya llegado por ninguno de los medios o por ninguno de los sentidos (Piaget, 1986). Es de aclarar que lo nuevo solo es nuevo para determinada persona, así no sea necesariamente reciente.

En consonancia con la teoría, la actitud del estudiante se define desde dos fronteras: La motivación y la capacidad que tenga el estudiante para relacionar conocimiento previo y nuevo conocimiento. En cuanto a la motivación, ésta puede definirse como el ánimo y la voluntad que el estudiante pone para acceder a nuevos saberes y nuevos conocimientos y para incorporarse, de manera voluntaria, en un determinado proceso de aprendizaje. La motivación es la clave para que el cerebro busque todos los caminos posibles para establecer relaciones entre el conocimiento previo y el nuevo conocimiento. La capacidad para desarrollar estas relaciones se da cuando la motivación es lo suficientemente sólida como para que el mismo cerebro busque todos los caminos posibles que las posibiliten.

Con lo cual el modelo que estableció el Dr. Ausubel para fundamentar su teoría del aprendizaje significativo determina que "… el ser humano aprende mucho más fácil todo aquello que tiene significado para él" [2] y parte del concepto de significado como esencia de dicha teoría basado en el conocimiento previo, el nuevo conocimiento y la actitud del estudiante.

2.2. Aprendizaje por Descubrimiento

Otra teoría aceptada por el mundo, y que ha posibilitado caminos más expeditos para el aprendizaje, se conoce como la teoría del aprendizaje por descubrimiento formulada por el Dr. Jerome Seymour Bruner quien, a diferencia de Ausubel, propone que "…el ser humano aprende mucho más fácil todo aquello que descubre" [4] con lo cual propone un panorama que prioriza la fascinación y lo insólito como fundamento para que el proceso de aprendizaje ocupe el espacio de la memoria a largo plazo.

Según su autor, el Dr. Bruner, actualmente profesor emérito de la Universidad de Nueva York, como descubrimiento se puede calificar a la "fascinación" que se traduce simplemente en la motivación que puede tener uno mismo para intentar explicarse lo insólito [4]. En el proceso de aprendizaje, como se plantea en la teoría del aprendizaje por descubrimiento, se involucran tres procesos casi simultáneos: De una parte se tiene la "adquisición" de nueva información. En segundo nivel, y casi al tiempo del primero, se presenta el proceso de manipular el conocimiento para hacerlo adecuado a nuevas tareas que el autor calificó como "transformación" y finalmente, está el proceso que permite comprobar si la manera con que hemos manipulado la información es adecuada a la tarea; etapa que el autor de la teoría calificó como "evaluación" [4].

De acuerdo a esto, en relación con la teoría de Aprendizaje Significativo puede definirse que el concepto de "descubrimiento" corresponde a la chispa que abre el camino para encontrar el significado, ubicando el conocimiento en el nivel de un patrón que permite que la memoria a largo plazo asimile, apropie y, eventualmente, aplique [4]. Con esto, se puede establecer una relación íntima entre el concepto de "significado" de la teoría de aprendizaje significativo y el concepto de "descubrimiento" de la teoría del aprendizaje por descubrimiento.

Finalmente, vale la pena incluir en esta fundamentación teórica sobre el aprendizaje que "Todo conocimiento puede ser objeto de aprendizaje, es decir, todo se puede aprender" [3] y que, por tanto, siempre existirán caminos expeditos para encontrar significado a lo nuevo y propiciar su descubrimiento por parte de las personas que se involucran en un proceso de aprendizaje.

2.3. Programación Funcional

De acuerdo a lo estudiado, la programación funcional se deriva del paradigma funcional, un modelo matemático basado en el cálculo Lambda que posibilita la construcción de soluciones simples basadas en funciones como núcleo básico de la programación [11]. La función constituye el elemento principal a partir del cual se construye una

solución que luego se revierte en un programa y que cuenta con características como paso de argumentos, nominación polimórfica, recursión, omisión de declaraciones y retornos automáticos.

Según la teoría, el paradigma de programación funcional aborda la construcción de soluciones a partir de tres conceptos básicos: Simplificar el objetivo a alcanzar, facilitar las pruebas de escritorios y reusar lo construido. Dado que el objetivo resulta ser lo más importante en la construcción de un programa, el paradigma de programación funcional posibilita no solo la clarificación del mismo, sino la simplificación en los frecuentes casos en los cuales el objetivo resulta tener un cierto nivel de complejidad.

En consecuencia, la realización de las pruebas de escritorio sigue siendo el mecanismo excelso para la comprobación y detección de errores en la construcción de un programa. Las pruebas de escritorio exigen el seguimiento lineal de las instrucciones que se han incorporado dentro de un programa y la verificación de sus resultados. En tiempos modernos, las pruebas de escritorio se han facilitado a partir de la incorporación de los botones Debug en los IDE (Integrated Development Environment) que son los ambientes de desarrollo que se han construido para facilitar la tarea de transcripción, digitación, edición, compilación y ejecución: Todo en uno [12].

Por lo tanto, el método tradicional de la realización de las pruebas de escritorio utiliza papel y lápiz y, cuando un programa se basa en funciones, realizarlo posibilita que las pruebas sean mucho más confiables, que éstas sean menos agotadoras y que, por lo tanto, sean mucho más ágiles; factores de gran impacto e importancia en el proceso de construcción de soluciones que pueden ser implementadas a través de un computador [8].

Según esto, reusar lo construido es una tendencia que se ha fortalecido con la irrupción del paradigma funcional dado que cuando se vuelve a utilizar lo que ya se tiene hecho, se logra acudir a fragmentos de código (funciones) que no solo ya han funcionado apropiadamente sino que también ya han sido probadas en tiempo de ejecución. De otra parte, la reutilización del código (a nivel de funciones) permite hacer un óptimo uso del tiempo y la labor de programar se vuelve mucho más eficiente. A partir de la aparición del concepto de librerías y bibliotecas (tanto estándares como de usuarios), la reutilización del código se convirtió en una estrategia usada con frecuencia, de forma que se pueda acudir a funciones eficientes y que, así mismo, pueda ser la programación basada en este paradigma.

Son estos tiempos los que le exigen que el desarrollo de software y con él, la construcción de programas incluya un ingrediente de alta importancia como es el buen uso del tiempo y para ello, la programación funcional con su filosofía de construcción de soluciones a partir de funciones, proporciona el camino preciso y perfecto para que así se cumpla, sin desconocer que otros paradigmas brindan herramientas que también posibilitan caminos eficientes de solución. En la actualidad, la tendencia a construir

soluciones basadas en funciones eficientes ha posibilitado no solo que se fortalezca la programación funcional sino también la programación estructurada y la programación orientada a objetos, que son las tendencias modernas que marcan el avance tecnológico en la actualidad.

2.4. Programación Estructurada

Así se define al primer paradigma formal de programación de computadores que se conoció como la programación estructurada, dado que es un modelo de programación que se basa en la máquina de estados de Von Newmann y se fundamente en tres estructuras básicas. Antes de la programación estructurada se acudía a una "técnica" conocida como programación libre en la cual cada programador hacía sus programas como a bien tuviera.

Sin embargo, el estudio exhaustivo de los programas realizados a partir de la llamada programación libre permitió ir encontrando que todos los programas hacían uso de tres estructuras específicas y a partir de allí, tomando los conceptos matemáticos de la máquina de estados de Von Newmann y de la máquina de Turing, se configuró un paradigma que, luego de más de sesenta años de haber sido formulado, sigue teniendo alguna vigencia y, dado el tiempo de refinamiento, no solo perdura en algunas expresiones tecnológicas sino que ha abierto la puerta para que otros paradigmas irrumpan en el mundo de la programación de computadores, solucionando apropiadamente lo que el paradigma estructurado no había podido solucionar.

Como su nombre lo indica, la programación estructurada se basa en unas estructuras básicas que en cantidad son tres y en definición corresponden a la secuencia de instrucciones, los condicionales y los ciclos [12]. Este tipo de programación también se conoce como programación imperativa aunque este concepto es compartido con otros paradigmas.

De acuerdo a esto la estructura de secuencia establece que una instrucción se ejecuta completamente luego de la anterior y antes de la siguiente y con ello, determina la precedencia de ejecución de las instrucciones lo cual le hace merecedor, a este paradigma estructurado, de lo puramente imperativo. La determinación de esta estructura permitió que muchas tareas se pudieran hacer de manera específica utilizando completamente la capacidad del computador y sus sistemas de procesamiento electrónico de forma que las tareas capitalizaran las altas velocidades que para tal fin se involucran.

Con el avance de la tecnología y la aparición de técnicas de programación como los threads (hilos) se ha podido entender que esta estructura en un ambiente puramente imperativo puede llegar a tener utilidad pero en otros ambientes (tal vez distribuidos o multiprocesados) impiden la realización de varias tareas al tiempo como sucede modernamente con los procesos multihilos que son los que permiten, por ejemplo, la ejecución simultánea de varias ventanas en el sistema operativo Windows [9].

Por otro lado, la estructura de decisión (o condicional) permite que se pueda escoger

uno de entre dos caminos lógicos dependiendo de una condición. Dicha condición se escribe en términos de operadores relacionales y booleanos y se evalúa a partir de los valores Verdadero o Falso que se pueden originar como respuesta de su revisión. Aunque no es necesario que todo condicional tenga de manera explícita los dos caminos, la estructura de decisión posibilita que siempre se puedan tener dos posibles opciones frente a una misma situación que pueda ser escrita en términos de los operadores mencionados. La estructura de decisión es, posiblemente, la única que se ha mantenido vigente desde que fue planteada como estructura básica y se ha extrapolado hacia otros paradigmas como es el caso de la programación funcional y la programación orientada a objetos con las mismas características que la han hecho útil en la programación estructurada.

Finalmente, la estructura cíclica o iterativa permite que se pueda ejecutar un conjunto de varias instrucciones tantas veces como una condición lo posibilite, de manera que su evaluación facilite la ejecución de las tareas iterativas que se hayan propuesto. La estructura cíclica es la que más ha evolucionado en cuanto a su concepción y características desde que fue formulada como una de las estructuras básicas. Actualmente, se puede hablar de dos formas de procesos cíclicos: Los ciclos formales de la programación estructurada (hacer hasta, repetir mientras, hacer para, etc.) y los ciclos llamados recursivos que, si bien no corresponden a las características de los ciclos estructurados, de todas formas mantienen el espíritu de ser formas de lograr que un conjunto de instrucciones se repitan de manera finita dependiendo de una condición [10]. Otra forma de construir ciclos son los ciclos no estructurados, heredados de la programación libre, pero estos ciclos se salen del contexto del presente artículo.

Tomando como base la programación estructurada como principal ejemplo del paradigma imperativo surgieron en el mercado una gran cantidad de lenguajes que, incluso hoy a más de cincuenta años de su aparición, siguen vigentes. Tal es el caso de Fortran, Cobol, Pascal, Logo y C que han logrado mantener en el mercado la esencia de la programación estructurada bien ampliándola, bien mejorándola, o mejor aún, evolucionando hacia otro paradigma pero partiendo de los elementos que caracterizan el paradigma imperativo.

2.5. Los tres grandes problemas de la programación

En referencia con el tema de programación, como elemento central de un perfil técnico profesional, la programación de computadores históricamente ha presentado tres grandes problemas que subyacen a todos los paradigmas y a todas las expresiones tecnológicas y posibilidades que proporcionan los lenguajes de programación y que, siendo comunes, han obligado a que el pensamiento humano trate de resolverlas en bien, no solo, de la programación como eje temático sino del desarrollo de aplicaciones y de las relaciones entre tecnología y sociedad.

Los tres grandes problemas de la programación son: a) La necesidad de simplificar el

objetivo de un programa, b) La ausencia de mecanismos que simplifiquen las pruebas de escritorio y c) La reutilización del código de un programa. En relación con el primer problema, se ha demostrado que cuando se tiene un problema complejo y éste, por su complejidad, se subdivide en pequeños problemas más simples, la solución también se simplifica pues se acude a resolver los problemas pequeños y no a enfrentar, como un todo, el gran problema original.

Esto facilita una de las grandes dificultades que tiene la programación como es la simplificación de las soluciones, dado que es mucho más sencillo programar resolviendo problemas simples que resolviendo problemas complejos, eso sí, con la claridad de que el vínculo de relación entre varias soluciones simples permiten resolver problemas de alta complejidad.

El segundo problema no es menos importante, dado que la herramienta más fuerte que tiene el programador para constatar que sus soluciones realmente resuelven un problema determinado es la prueba de escritorio que consiste en una simulación de lo que haría el computador con un determinado código y de los resultados que arrojaría, solo que dicha simulación se puede hacer bien desde el papel ("en frío") o bien aprovechando los recursos de depuración (debug) que brindan entornos integrados de desarrollo (IDE – Integrated Development Environment) de los lenguajes de programación.

La realización de pruebas de escritorio que implican demasiado tiempo, demasiadas iteraciones y demasiadas instrucciones, terminan siendo bastante agotadoras y con el agravante de que dejan en tela de juicio su confiabilidad. Esta es una de las razones por las cuales, en la actualidad, muchos programadores acuden pocas veces a la prueba de escritorio como mecanismo excelso para verificar que su solución resuelve un determinado problema.

La tercera dificultad hace referencia a la necesidad de reutilizar código dado que cuando se escribe un programa es muy posible que gran parte de él sea útil en la construcción de otras soluciones. Por este motivo, todo el tiempo que un programador pueda ahorrarse en desarrollo será tiempo altamente valioso dado que el código que pueda reutilizar en otros programas le ayudará a que su labor de programación sea más eficiente frente a la única variable que, en programación de computadores, no es posible reponer o adquirirle repuesto y que se trata del tiempo.

2.6. El concepto de función

La función constituye el elemento central, la célula, de la programación. Es la base para que los paradigmas coexistan y se puedan comprender unos desde la óptica de otros. La función se define como un pequeño conjunto de instrucciones que tiene un nombre específico, puede recibir parámetros y puede retornar un valor en cumplimiento de un pequeño objetivo. Nótese que se destacan los conceptos "pequeño conjunto de instrucciones" y "pequeño objetivo" para determinar la característica principal de una función puesto que no hacer hincapié en

este adjetivo podríamos confundir, tanto en la teoría como en la práctica, el concepto de función con el concepto de programa [11].

La función es la base de la programación funcional dado que éste debe su nombre a la organización de un programa a partir de la alta utilización de funciones y, particularmente, de todas las posibles relaciones que entre ellas se puedan dar. En la programación estructurada pasa algo parecido, si bien las soluciones se pueden construir en un solo bloque de instrucciones, la utilización de funciones permite que los objetivos a resolver a partir de las estructuras básicas que involucra el paradigma imperativo se haga más sencillo si se parte del concepto de función como núcleo central y cuya forma de hacerlo es, precisamente, el objetivo de este artículo.

En la programación orientada a objetos la función tiene un nombre propio: Se conoce como método, una de las dos partes principales de la declaración de cualquier clase y, por ende, de la utilización de los objetos concebidos como instancias de las clases definidas. Las funciones como métodos en la programación orientada a objetos son una de las dos partes que conforma un objeto: Los atributos y los métodos.

De manera que la función, concebida desde esta perspectiva, termina siendo la base para la comprensión de los paradigmas más importantes en el mundo moderno y también para el establecimiento de relaciones entre dichos paradigmas como se pretende demostrar en este artículo.

3. METODOLOGÍA

3.1. Descripción

La metodología propuesta va a ser ejemplificada a través del siguiente enunciado: Construir un programa que permita leer un número entero positivo de tres dígitos y determinar si la suma de sus dígitos es un número par. Este enunciado será resuelto tomando como base el lenguaje C para la construcción de las funciones. Lo primero que debemos hacer es tomar el enunciado y desglosarlo de la siguiente forma:

- construir un programa que permita
- leer un número entero positivo de tres dígitos y
- determinar si la
- suma de sus dígitos
- es un número par

En segundo lugar, se procede a dejar las palabras y elementos claves que sean suficientemente descriptivos y que puedan configurarse como pequeños objetivos de este gran objetivo. De esta manera, el desglose quedaría de la siguiente forma:

- Función que lee un entero positivo de tres dígitos, lo valida y pasa a la función respectiva
- Función que calcula la suma de los dígitos de un número de tres dígitos

- Función que determina si un número es par

A este reacomodamiento del desglose del enunciado se le adicionan los soportes que se necesiten para que cada función cumpla su objetivo, de la siguiente forma:

- Función que lee un entero positivo de tres dígitos, lo valida y pasa a la función respectiva
  - Función que determina si un número es positivo
  - Función que determina si un número es de tres dígitos
- Función que calcula la suma de los dígitos de un número de tres dígitos
  - Función que obtiene el 1º dígito de un número de tres dígitos
  - Función que obtiene el 2º dígito de un número de tres dígitos
  - Función que obtiene el 3º dígito de un número de tres dígitos
- Función que determina si un número es par

De esta manera, queda construido el esquema general de documentación del programa a desarrollar, ahora solo falta adicionarle el código. Así pues, llevado a los términos del lenguaje C, el esquema de documentación quedaría de la siguiente forma:

```
// Librerías y prototipos
// Función que lee un entero positivo de tres dígitos, lo valida y pasa a la función respectiva
// Función que determina si un número es positivo
// Función que determina si un número es de tres dígitos
// Función que calcula la suma de los dígitos de un número de tres dígitos
// Función que obtiene el 1º dígito de un número de tres dígitos
// Función que obtiene el 2º dígito de un número de tres dígitos
// Función que obtiene el 3º dígito de un número de tres dígitos
// Función que determina si un número es par
```

Nótese, que aparece como primera línea un nuevo comentario que no se había tenido en cuenta y que hace referencia a las librerías y prototipos, algo muy propio del lenguaje C y que es ineludible por la sintaxis misma del lenguaje. Cabe aclarar que este programa se ha escrito y ejecutado en el compilador DevC++ versión 4.9.9.2. Ahora empezamos a construir las funciones que corresponden a cada comentario. Las vamos escribiendo y las vamos documentando.

```
// Librerías y prototipos de funciones
#include <iostream.h>

int positivo (int);
int numde3dig (int);
int suma3dig(int);
int pdn3dig(int);
int sdn3dig(int);
int tdn3dig(int);
int par(int);
```

```
// Función que lee un entero positivo de tres
   dígitos, lo valida y pasa a la función respec-
   tiva
   int main( )
   { int num;
   //  Declaración variable num tipo entero
   cout<<"Escriba un entero positivo de 3 di-
   gitos...>";  / Muestre aviso de petición
   cin>> num;
   // Lea un entero y guárdelo en num
   if ( positivo(num) && numde3dig(num))    //
   Si el número es positivo y es de 3 digs
   suma3dig(num);
   // llamar a la función que suma los 3 dígitos
   else cout<<"Error en el dato ingresado ";
   // sino mostrar aviso de error
   }

//  Función que determina si un número es po-
   sitivo
   int positivo (int n)
{
   // Inicio de función
   if (n > 0 )
   // Si el número recibido es positivo
   return(1);
   // retorne VERDADERO
   else
   // sino
    return(0);
   // retorne FALSO
}                                          / /
   Fin de función

//  Función que determina si un número es de
   tres dígitos
   int numde3dig (int n)
{
   // Inicio de función
   if (n>=100 && n<=999)
   // Si el número es de 3 dígitos
   return(1);
   // retorne VERDADERO
   else
   // sino
   return (0);
   // retorne FALSO
}
```

Nótese que, para el llamado de las funciones positivo( ) y numde3dig( ) se aprovecha el hecho de que en lenguaje C el valor 0 es FALSO [5] y cualquier valor diferente de cero es VERDADERO. De esta forma, se simplifican los llamados y los retornos de las funciones. Con este conjunto de instrucciones hemos resuelto el primer pequeño objetivo. De la misma manera resolvemos los otros pequeños objetivos.

```
// Función que calcula la suma de los dígitos
   de un número de tres dígitos
   int suma3dig(int n)
   {
   // Inicio de la función
   int r;
   // Declaración de variable local
   r=pdn3dig(n)+sdn3dig(n)+tdn3dig(n);
   // Se obtiene la suma de los tres dígs
   cout<<"La suma de sus dígitos es "<<r;
   // Se muestra la suma de los tres dígis
   if (par(r))
   // Si dicha suma es par
   cout<<"La suma de los dígitos es par";// Se
   muestra el aviso correspondiente
   else
   // si no es par
  cout<<"La suma de los dígitos no es par"; //
   también se avisa
}  // Fin de la función
```

```
// Función que obtiene el 1º dígito de un número de tres dígitos
int pdn3dig(int n)
{ // Inicio de la función
return(n/100);
// Retorna el 1o dígito de un num de 3 digs
} // Fin de la función

// Función que obtiene el 2º dígito de un número de tres dígitos
int sdn3dig(int n)
{ // Inicio de la función
return((n/10)%10); / /
Retorna el 2o dígito de un num de 3 digs
} // Fin de la función

// Función que obtiene el 3º dígito de un número de tres dígitos
int tdn3dig(int n)
{ // Inicio de la función
return(n%10); / /
Retorna el 3o dígito de un num de 3 digs
} // Fin de la función

// Función que determina si un número es par
int par(int n)
{ // Inicio de la función
if(n%2==0)
// Si el valor recibido es par
return(1);
// retorne VERDADERO
else
// Sino
return(0);
// Retorne FALSO
}
// Fin de la función
```

Puede notarse que, en la medida en que se va construyendo cada función, se va documentando, con el ánimo de que se sepa claramente lo que hace cada instrucción [6]. Con esta propuesta metodológica construir un programa dentro del paradigma estructurado a partir del concepto de función se vuelve muy sencillo pues todo lo que se necesita es desglosar apropiadamente el enunciado e ir desarrollando cada una de las funciones que resulten en cumplimiento de sus pequeños objetivos.

3.1.1. Aplicación

Durante el desarrollo del contenido de la asignatura Programación II (Programación Estructurada) en Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira en el período establecido, se adoptó la siguiente mecánica para la aplicación de esta propuesta metodológica. Primeramente, se agotó el tiempo que debía destinarse para la fundamentación teórica y que, teniendo en cuenta que el semestre tiene 16 semanas, corresponde aproximadamente a las primeras 4 semanas dependiendo del ritmo de aprendizaje que tenga el curso y de la manera como interactúen el estilo de enseñanza del Docente con el estilo de aprendizaje de la mayoría de los alumnos.

Durante este período, hablando en términos del contenido de la asignatura Programación II, se sirven los conceptos relacionados con lo puramente metodológico en cuanto a la relación con el desarrollo de programas a la luz de la programación

estructurada y se abordan los temas fundamentales que permiten establecer un enlace entre el paradigma como modelo matemático y el lenguaje de programación en el cual se quiere hacer efectivo dicho paradigma. Estos temas iniciales incluyen el concepto de variable, el uso y jerarquía de los operadores así como su clasificación, las dos primeras estructuras (secuencias y decisiones) y algunos recursos propios de los lenguajes de programación que permiten aprovechar las bondades del paradigma.

Seguidamente, se adopta la metodología expuesta para que los estudiantes vayan apropiando la idea de construir soluciones a problemas, partiendo de la concepción de las funciones como elemento central y, sobre este concepto, poder resolver problemas de alta complejidad partiendo de la simplicidad que ofrece la función como concepto central.

De allí en adelante, las tres sesiones semanales de programación se distribuyen de la siguiente forma: En la primera sesión semanal se presenta la teoría asociada a un nuevo recurso y se resuelven algunos enunciados basados en la metodología de construcción de programas a partir de funciones; la segunda sesión se destina para realizar talleres en donde el estudiante no solo interactúe con los conceptos vistos en la primera sesión, sino que se vea enfrentado a la utilización de las funciones como concepto central en el marco de la construcción de un programa y en la tercera sesión, se lleva lo que se hizo tanto en la primera como en la segunda sesión, a la sala de computadores para que el estudiante vea la ejecución de lo que programó en frío. Este esquema de trabajo funciona maravillosamente siempre y cuando no haya interrupciones en el desarrollo de las sesiones (paros, cese de actividades, asambleas de estudiantes) y toda vez que la tercera sesión coincida con la sesión práctica.

Dado que la temática de la programación estructurada es suficientemente amplia se puede contar por lo menos con un nuevo tema cada semana, ponerlo en práctica "en frío" en la misma semana a través de la realización de un taller preparado para tal fin y hacerlo efectivo en el computador en esa misma semana. Sabiendo que pueden ser muchos los factores que influyen en el desarrollo normal de las actividades académicas es importante tener en cuenta que debe mantenerse el ritmo de un modelo que he llamado TETAPRAS (Teoría – Ejercicios, Taller, Práctica en la Sala) para que se afiancen los conocimientos desde las tres aristas mencionadas: La exposición magistral, la realización de ejercicios por parte del Docente, la realización de talleres por parte del estudiante con la asesoría y acompañamiento del Docente y la puesta en funcionamiento de los programas en el computador.

4. RESULTADOS

La aplicación de esta metodología ha generado unas reacciones muy positivas en los estudiantes de la asignatura Programación II (Programación Estructurada) en Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira durante el periodo de investigación establecido. Esta metodología se sigue utilizando en las clases que imparte el autor de este artículo en las asignaturas que sirve. Durante los cuatro primeros semestres se realizaron pruebas en paralelo con dos grupos: Uno de ellos adoptando esta propuesta metodológica y el otro sin adoptarla. Los resultados cuantitativos del promedio de las notas definitivas de la asignatura Programación II se muestran en la Tabla 1.

Tabla 1. Promedio de Notas

Descripción	2011		2012	
	I Sem	II Sem	I Sem	II Sem
Aplicando la metodología	4.3	4.5	4.4	4.6
Sin aplicar la metodología	3.2	3.8	3.6	3.6

Ahora bien, el análisis cualitativo del rendimiento académico de los estudiantes en cuanto a la apropiación de la programación, su aplicación y la incorporación del concepto de función como núcleo básico para el desarrollo de cualquier programa ha arrojado los resultados que se exponen en la Tabla 2. Es de anotar que estos resultados cualitativos no se basaron en las notas sino en diferentes factores que van más allá de lo puramente cuantitativo: Nivel de las preguntas que hacían, asimilación del concepto de función, aplicación y construcción de funciones, capacidad de explicación de dudas por parte de los mismos compañeros, etc.

Tabla 2. Análisis Cualitativo del rendimiento académico

Descripción	2011		2012	
	I Sem	II Sem	I Sem	II Sem
Aplicando la metodología	Muy alto	Muy alto	Alto	Muy alto
Sin aplicar la metodología	Alto	Regular	Regular	Alto

Para el análisis cualitativo del rendimiento académico de los cursos, se utilizó una escala de valoración de la siguiente manera: Pésimo, Muy bajo, Bajo, Regular, Alto, Muy Alto, Excelente. Se hicieron algunas subdivisiones para poder aproximarse más a la realidad dado que, los análisis cualitativos pueden llegar a tener mucho de subjetivo y se ha tratado de ser lo más objetivo posible dentro de la aplicación de esta metodología.

5. DISCUSIÓN

Debe anotarse que las tablas que se presentan en este artículo corresponden al resumen de una cantidad de datos recogidos durante la experiencia y que, en lo cuantitativo, incluyen notas de los dos exámenes parciales y del examen final, notas de talleres, notas de quices, notas de ejercicios y notas de otras actividades calificables [13]. En lo cualitativo, la valoración del rendimiento académico y la asimilación de los estudiantes incluyen conversaciones personalizadas, entrevistas directas, opiniones grupales, conceptos espontáneos y la utilización de una cuenta de correo anónima a través de la cual los estudiantes pueden manifestar libremente su opinión.

También debe dejarse en claro que, se trató de que todas las pruebas que se hicieran en ambos cursos fueran similares debido a que solo así se podían recoger resultados que fueran confiables tanto en lo cuantitativo como en lo cualitativo. Es claro que esta metodología se ha puesto a prueba durante dos años completos (4 semestres) y por ello podría pensarse en una avanzada investigativa más profunda para extender el tiempo de investigación. Sin embargo, a juicio del autor, se consideran suficientes los resultados recogidos, puesto que los procesos de evaluación y análisis se han realizado con la mayor rigurosidad posible a nivel investigativo.

La relación entre los cursos donde se aplicó la metodología y aquellos en los que no es significativamente alta. Tanto los resultados cuantitativos como los cualitativos son superados notoriamente por los cursos en donde se impartió el contenido de la asignatura basado en la metodología propuesta. Esto podría significar tres cosas: Primero, que la metodología propuesta pareciera ser efectiva y pareciera facilitar el aprendizaje de construcción de programas imperativos basados en el concepto de función como núcleo central de desarrollo.

Segundo, no se descarta que sea posible refinar la metodología tradicional hasta lograr resultados como los que se han logrado con la aplicación de la metodología propuesta; lo que se necesitará es desarrollar toda una investigación que posibilite el aprendizaje de la programación

estructurada por caminos que sean más sencillos y digeribles para el estudiante, como el que se expone en la metodología propuesta en este artículo.

Tercero, sigue abierta la posibilidad de refinar mucho más la metodología propuesta a fin de que se puedan lograr resultados mucho más altos, por encima de las notas o de los conceptos cualitativos, en lo que corresponde al aprendizaje de la programación y particularmente, en lo que compete a la asimilación del paradigma de programación estructurada así como las posibles relaciones que se pueden establecer con el paradigma de programación funcional y particularmente, con el concepto de función como núcleo central de trabajo y base para el desarrollo de programas. A partir de esta metodología, pareciera que los estudiantes encuentran un camino más llano para asimilar las herramientas y recursos con que cuenta el lenguaje de programación que, en este caso, corresponde al lenguaje C en su sabor DevC++.

Resulta para el estudiante ser muy satisfactorio encontrar relación entre lo que está aprendiendo (programación estructurada) y los conceptos que ya tiene como conocimiento previo (programación funcional), [3] para el caso de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira). Esta situación genera automáticamente una alta motivación y constituye la base para que su actitud, tal como lo establece Ausubel [2] sea exitosa en el proceso de aprendizaje significativo.

De otra parte, en la medida en que el estudiante va descubriendo las relaciones posibles entre lo que ya sabe y lo que está comenzando a aprender, en la medida en que encuentra por sus propios medios la aplicación de nuevos conceptos y su conexión con conceptos anteriores, el proceso de aprendizaje se hace muchísimo más expedito y más sencillo tal como lo establece Bruner [4].

6. CONCLUSIONES

Es posible realizar procesos de evaluación cuantitativa y cualitativa efectivos y confiables en las asignaturas de Ingeniería que involucran programación de computadores.

Se hace necesario que la valoración de las evaluaciones se haga tanto desde lo cuantitativo como desde lo cualitativo.

Cualquier propuesta metodológica es necesario complementarla con actividades. intraclase y extraclase que fortalezcan el proceso autónomo de aprendizaje por parte del estudiante y que le permitan compenetrarse con él mismo fijando y revisando sus propias metas.

La comunicación directa con los estudiantes y la aceptación de las opiniones al res-

pecto de determinada metodología es lo que permite que ella sea refinada al punto de convertirse en base para procesos exitosos de aprendizaje.

Pareciera que siempre se hace necesario realizar pruebas en paralelo para tener una aproximación mayor a los efectos de la aplicación de determinada metodología.

Vale la pena profundizar en la investigación educativa en los campos como la Ingeniería.

Deben refinarse los métodos que permita realizar investigaciones con resultados cuantitativos y cualitativos de manera que los procesos investigativos además de ser rigurosos sean confiables.

Puede considerarse a la función, como concepto, el núcleo central de tres paradigmas de programación.

La apropiación del concepto de función simplifica la aplicación de los conceptos que giran alrededor de la programación estructurada.

Toda metodología que se propone para mejorar los procesos de aprendizaje en Ingeniería, son susceptibles de mejorarse [7].

Siempre existirán caminos más simples para compartir el conocimiento con los estudiantes; hallarlos solo dependerá de la capacidad investigativa del Docente y de la vocación que él tenga en cumplimiento de sus funciones.

Todo resultado cuantitativo debe complementarse con un análisis cualitativo.

La teoría del aprendizaje significativo y la teoría del aprendizaje por descubrimiento facilitan fuertemente los procesos de aprendizaje en la formación de Ingenieros.

BIBLIOGRAFÍA

[1] Aamodt, S. (2009). Entra en tu cerebro. Ediciones B. Barcelona. 335p.

[2] Ausubel, P. (1986). Sicología Educativa: Un punto de vista cognoscitivo. México, Trillas. 1187p.

[3] Bransford, J. (2003). How people learn. National Academic Press. Washington, D.C. 385p.

[4] Bruner, J. (1963). El proceso de la Educación. México, Hispanoamericana. 152p.

[5] Joyanes, L. (2005). Programación en C: metodología, algoritmos y estructuras de datos. McGraw Hill Interamericana. España. 256p.

[6] Kernigham, B. (1988). Lenguaje C. Prentice Hall. México. 132.

[7] Medina, J. (2010). Los 12 principios del cerebro. Grupo Editorial Norma. Santafé de Bogotá. 328p.

[8] Small, G. (2010). El cerebro digital. Editorial Urano. Barcelona. 254pp.

[9] Trejos B, O. (2005). Fundamentos de Programación. Editorial Papiro. Pereira. Colombia. 120p.

[10] Trejos B., O. (2002). La esencia de la Lógica de Programación. Centro Editorial Universidad de Caldas. Manizales. Colombia. 365p.

[11] Van Roy, P. (2003). Concepts, Techniques and Models of Computer Programming. Swedish Institute of Computer Science. Estocolmo. Suecia. 939p.

[12] Van Santen, R. (2010). 2030 Technology that will change the world. Oxford University Press. 304p.

[13] Verlee, L. (1986). Aprender con todo el cerebro. Ediciones Martínez Roca. Barcelona. 242p.