Programación multiparadigma como estrategia de aprendizaje de los lenguajes de programación en ingeniería de sistemas*[1]

[Multiparadigm programming as a strategy for learning programming languages in systems engineering]

Ricardo TIMARÁN PEREIRA[2], Javier JIMÉNEZ TOLEDO[3], Anivar CHAVES TORRES[4]

Recibo: 20.08.2012 - Aprobación: 08.11.2012

Resumen

Para el desarrollo de software se cuenta con varios paradigmas de programación, cada uno provisto de sus metodologías, técnicas y herramientas y orientado a un determinado campo o a un conjunto de problemas, y por ende, ninguno es suficiente por sí mismo para solucionar todos los problemas que se puedan suscitar. En este artículo se presentan los resultados del proyecto de investigación que tiene como objetivo la apropiación y aplicación del modelo de programación multiparadigma con el entorno Mozart-Oz para el desarrollo de software en el programa Ingeniería de Sistemas de la Universidad de Nariño. Esta investigación

* Modelo para citación de este artículo de investigación científica y tecnológica: TIMARÁN PEREIRA, Ricardo; JIMÉNEZ TOLEDO, Javier & CHAVES TORRES, Anivar (2012). Programación multiparadigma como estrategia de aprendizaje de los lenguajes de programación en Ingeniería de Sistemas. En: Ventana Informática. No. 27 (jul.-dic., 2012). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 41-54. ISSN: 0123-9678

1 Artículo proveniente del proyecto *Apropiación y aplicación del modelo de programación multiparadigma con el entorno Mozart-Oz en el desarrollo de software en el programa Ingeniería de Sistemas de la Universidad de Nariño*, ejecutado en el periodo junio de 2010 a junio de 2012, e inscrito en el grupo de investigación GRIAS, de la Universidad de Nariño.

2 Doctor en Ingeniería, Master of Science en Ingeniería, Especialista en Multimedia e Ingeniero de Sistemas y Computación. Director grupo de investigación GRIAS, Profesor Asociado, Departamento de Sistemas, Facultad de Ingeniería, Universidad de Nariño (Pasto, Nariño, Colombia). Correo electrónico: ritimar@udenar.edu.co

3 Especialista en Docencia Universitaria, Ingeniero de Sistemas, Profesor hora cátedra, Departamento de Sistemas, Facultad de Ingeniería, Universidad de Nariño (Pasto, Nariño, Colombia). Correo electrónico: javierjx@gmail.com

4 Especialista en Docencia Universitaria, Ingeniero de Sistemas, Profesor hora cátedra, Departamento de Sistemas, Facultad de Ingeniería, Universidad de Nariño (Pasto, Nariño, Colombia). Correo electrónico: anivarchaves@yahoo.com

se realizó en tres fases en las que se estudian y evalúan la programación estructurada y orientada a objetos, la programación funcional y la programación por restricciones, con el fin de desarrollar en los estudiantes las competencias específicas en la solución de problemas utilizando estos modelos y entorno.

Palabras Clave: *Programación Multiparadigma, Entorno de Desarrollo Mozart-Oz, Aprendizaje de Lenguajes de Programación.*

Abstract

For software development has several programming paradigms, each equipped with their methodologies, techniques and tools aimed at a particular field or set of problems, and therefore, none is sufficient by itself to solve all problems that can inspire. This paper presents the results of the research project that aims at the appropriation and application of multiparadigm programming model with the Mozart-Oz environment for software development in the Systems Engineer program at the Universidad of Nariño. This research was conducted in three phases in which structured and object-oriented programming, functional programming and constraints programming was studied and evaluated, in order to develop in students the specific skills to solve problems using these models and environment.

Keywords: *Multiparadigm Programming, the Mozart-Oz Development Environment, Learning Programming Languages*

Introducción

Cada uno de los modelos de programación se ha desarrollado en algún momento histórico y se ha orientado a la solución de un conjunto de problemas para los cuales se aplica adecuadamente. El ingeniero, analista o desarrollador encuentra que la realidad es muy compleja y que en la práctica requiere combinar los diferentes paradigmas de programación con el fin de construir una solución de software de alto nivel.

Uno de los inconvenientes que siempre se encuentra presente es el tener que apropiar nuevos lenguajes de programación cuando es necesario recurrir a dar solución a un problema con la aplicación de un paradigma de programación diferente, debido a que en muchas ocasiones el lenguaje de programación con el que se ha trabajado no tiene soporte para abordar un concepto diferente al afrontar algunas metodologías de implementación de software que harían más fácil y robusto un desarrollo.

Ante esta realidad se ha propuesto la programación multiparadigmática o multiparadigma. Esta se basa en la idea de combinar varios paradigmas en un mismo marco de programación, según Fernández (2005), como resultado de la coexistencia de los paradigmas orientado a objetos, procedural, declarativo y funcional buscando mejorar la producción en el desarrollo de proyectos, de acuerdo con Gewali & Minor (2006) y Pérez & López (2007). La combinación de paradigmas de programación pretende que en cada subprograma se pueda utilizar un paradigma diferente, aquel en el cual sea más natural programar. Actualmente se cuenta con entornos de desarrollo que permiten aplicar cualquier modelo de programación sin tener que aprender o utilizar otros lenguajes, ejemplo de ello es Mozart-Oz (Mozart, s.f.), de igual en otros lenguajes se dispone de librerías o módulos adicionales para cumplir con este objetivo (Porkoláb & Zsók, 2006).

Apropiar un lenguaje de programación multiparadigma en la formación de los ingenieros de sistemas llevaría a fortalecer el estudio de las diferentes metodologías de programación en lugar de dedicar basto tiempo a la enseñanza del lenguaje formal para luego aplicar a un paradigma determinado. Además, los estudiantes de Ingeniería de Sistemas desarrollarían la competencia de integrar los diferentes paradigmas con el fin de construir mejores productos de software.

Bajo la concepción anterior, se tendría una gama mayor de expertos con grandes destrezas en un lenguaje que con el paso del tiempo permitiría una mayor apropiación (en este caso sería un lenguaje multiparadigma) y capaces de abordar una solución de software de acuerdo a las verdaderas necesidades de implementación y no acorde a la especialidad en un lenguaje de programación específico con el cual se han identificado.

En este orden de ideas, Van Roy & Haridi (2004) proponen una metodología de enseñanza de la programación en la que se integran los paradigmas de programación más comunes como son: programación orientada a objetos, programación lógica, programación funcional, programación concurrente, programación por restricciones, utilizando el entorno de programación multiparadigma Mozart-Oz.

En este artículo se presentan los resultados del proyecto de investigación que tiene como objetivo la apropiación y aplicación del modelo de programación multiparadigma con el entorno Mozart-Oz para el desarrollo de software en el programa Ingeniería de Sistemas de la Universidad de Nariño. Esta investigación se realizó en tres fases en las que se estudian y evalúan la programación estructurada y orientada a objetos, la programación funcional y la programación por restricciones (Apt, 2003), con el fin de desarrollar en los estudiantes las competencias específicas en la solución de problemas utilizando estos modelos y entorno.

El resto del artículo se organiza de la siguiente manera. En la sección 1, se presenta los conceptos básicos sobre la programación multiparadigma. En la sección 2, se describe la metodología utilizada en la investigación. En la sección 3, se presentan los resultados de cada una de las fases de la investigación y la discusión de resultados y finalmente, en la última sección se presenta las conclusiones y trabajos futuros.

1. Fundamento teórico

Un paradigma de programación, de acuerdo con Appleby & Vandekopple (1998), es una colección de modelos conceptuales que orientan el proceso de diseño y desarrollo de programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión y alta rentabilidad.

Los paradigmas de programación representan un enfoque particular o filosofía para la construcción del software, por lo que no es mejor uno que otro; sino que cada uno tiene ventajas y desventajas, existiendo situaciones donde un paradigma resulta más apropiado que otro. Aunque en teoría los modelos de programación pueden o no tener lugares comunes entre sí, en la práctica difícilmente un modelo se implementa de forma exclusiva. Cuando los paradigmas se entremezclan en un mismo proyecto para contribuir a una mejor solución se dice que dicho proyecto o solución es multiparadigma, por lo que el uso de distintos modelos, según se adapten mejor a las diversas partes de un problema, resulta más natural y permite expresar de forma más clara y concisa la solución.

Un lenguaje de programación multiparadigma es aquel que soporta más de un paradigma de programación. El objetivo de diseño de estos lenguajes es permitir a los programadores utilizar la mejor herramienta para cada trabajo, admitiendo que ningún paradigma resuelve todos los problemas de la forma más fácil y eficiente posible.

«Oz es un lenguaje de programación multi-paradigma (puede usarse en forma procedural, funcional, con restricciones lógicas u orientado a objetos) que soporta programación en soft-real time, concurrencia, distribución y programación reactiva» (Clúa, 2004), caracterizado por tener una semántica formal simple y una implementación eficiente. Su principal fortaleza es el desarrollo de aplicaciones por restricciones y la programación distribuida, lo cual fácil programar aplicaciones abiertas y tolerantes a fallas en el lenguaje. Para programación con restricciones, Oz introduce la idea de espacios de computación, los cuales permiten búsquedas definidas por el usuario y estrategias de

distribución que son ortogonales al dominio de restricciones. Oz es un lenguaje orientado a la concurrencia, término introducido por Joe Armstrong, el principal diseñador del lenguaje Erlang, según reportan Armstrong et al., (1996).

La plataforma de desarrollo Mozart es un sistema de programación avanzado diseñado para aplicaciones inteligentes y distribuidas, que surge como resultado de un proceso extenso de investigación en lenguajes de programación, inferencia tomando como base las restricciones, computación distribuida y el diseño de interfaces humano-computador, de acuerdo con Mozart (s.f) y Vargas (2004).

Otros lenguajes de programación como C++ cuentan también con componentes que le acondicionan la capacidad para trabajar con varios de los paradigmas en forma simultánea. Uno de estos módulos está representado por las librerías de GECODE, que le proporcionan al código la posibilidad de trabajar con Restricciones, de acuerdo con Schulte, Tack & Lagerkvist (s.f).

El lenguaje Java también cuenta con paquetes de clases adicionales que permiten trabajar con otros paradigmas, tal es el caso del paquete CHOCO (Choco, s.f.) que permite codificar en Java programas que requieren de las características de restricciones.

2. Metodología

En esta investigación se pretendió apropiar la programación multiparadigma utilizando el entorno de desarrollo (Mozart) y el lenguaje (OZ) en tres fases: en la primera fase se estudia la programación estructurada y orientada a objetos; en la segunda fase, la programación funcional y en la tercera fase, la programación por restricciones.

La investigación se desarrolló bajo al enfoque cuantitativo aplicando el método empírico-analítico. El diseño experimental desarrollado se basó en una pre-prueba, post-prueba y un grupo experimental. El diseño corresponde a un cuasi-experimento en el cual el grupo experimental lo conformaron 31 estudiantes del programa de Ingeniería de Sistemas de la Universidad de Nariño que matricularon en los semestres VIII, IX y X los cursos de Programación Avanzada I, Programación Avanzada II y Seminario de Computación e Informática respectivamente.

En cada fase se evalúan los resultados y habilidades que van desarrollando estos estudiantes. Las fases I y II son la base para que en la fase III los estudiantes afronten sin dificultades un nuevo paradigma. Las características generales de estos estudiantes se presentan en la tabla 1.

Tabla 1. Caracterización estudiantes grupo experimental

Pregunta	Respuesta	%
Género	Femenino	34.4
	Masculino	65.6
Estado Civil	Soltero	96.9
	Unión Libre	3.1
Tiene Hijos	Si	9.4
	No	90.6
Vive con sus padres	Si	81.3
	No	18.2
Edad	21	9.4
	22	18.8
	23	25.0
	24	12.5
	26	12.5
	27 o más	15.6
Estrato Socioeconómico	1	25.0
	2	40.6
	3	31.3
	4	3.1
Municipio de procedencia	Capital	65.6
	Otro municipio del departamento	34.4
Ha repetido asignaturas de programación	Si	37.5
	No	62.5
Trabaja	Si	34.4
	No	54.6
Fortaleza de programación	1	9.4
	2	6.3
	3	50.0
	4	34.4

La técnica de recolección de datos fue la encuesta. En cada fase del proyecto, para determinar los conocimientos sobre programación se utilizó cuestionarios y para establecer la aplicación de dichos fundamentos se propuso ejercicios de programación.

De acuerdo a la tabla 1, el género predominante es el masculino, el estado civil de la mayoría es soltero, así como manifiestan no tener hijos, viven con sus padres y no han repetido materias. El estrato socioeconómico predominante del grupo es 2 (bajo medio) y el municipio de procedencia de los estudiantes es la capital del departamento.

En el grupo experimental se encuentra que solo el 34,4% combina el estudio con el trabajo y el resto de estudiantes se dedica únicamente

al estudio de la Ingeniería de Sistemas, además la gran parte del grupo establece en 3.0 su fortaleza de programación.

3. Resultados y discusión

3.1 Fase 1. Programación imperativa

El objetivo de la primera fase del proyecto de investigación *Apropiación y Aplicación del modelo de programación multiparadigma con el entorno Mozart-Oz en el desarrollo de software en el programa Ingeniería de Sistemas de la Universidad de Nariño,* fue el apropiar y aplicar el paradigma imperativo (programación estructurada y orientada a objetos) con Mozart-Oz en el curso de Programación Avanzada I del VIII semestre, con el fin de desarrollar las competencias en la solución de problemas utilizando este modelo y entorno. El curso se programó para 16 semanas con una intensidad semanal de 4 horas y distribuido en cuatro unidades cuyas temáticas se muestran en la tabla 2. Cada sesión de clase fue planeada y concertada con los estudiantes, en el cual se establece los temas a tratar, su objetivo, los contenidos, las actividades de aprendizaje, las estrategias de evaluación y el tiempo de dedicación presencial e independiente.

En general, el tratamiento experimental generó buenos resultados en el manejo de las temáticas estudiadas durante el semestre académico a tal punto que los resultados de la post-prueba del grupo experimental, después del tratamiento experimental aplicado, es superior para los ítems evaluados con respecto a los encontrados del mismo grupo en su fase de pre-evaluación. Los buenos resultados se deben a que los estudiantes ya conocen el paradigma imperativo, pero con entornos y lenguajes de desarrollo diferentes en cursos de semestres anteriores del componente de programación del programa de Ingeniería de Sistemas. El curso de Programación Avanzada I con Mozart-Oz afianzó estos conocimientos.

3.2 Fase 2. Programación funcional

La segunda fase de esta investigación tenía como objetivo la apropiación del modelo de programación funcional con Mozart-Oz. Esta fase se llevó a cabo en el curso Programación Avanzada II del IX semestre del plan de estudios de Ingeniería de Sistemas, cuyo prerrequisito es Programación Avanzada I. Este curso tiene una intensidad de 64 horas con dedicación de 4 horas semanales.

La evaluación preliminar mostró que los estudiantes tenían algunos conocimientos teórico-prácticos sobre cinco modelos de programación:

estructurado, orientado a objetos, orientado a eventos, funcional y lógico. Los dos modelos más conocidos son el estructurado y el orientado a objetos, el primero es conocido por el 87,5% de los estudiantes y el segundo por la totalidad. El 37,5% manifiestan conocer los fundamentos teóricos del modelo funcional y el mismo porcentaje dicen conocer el modelo orientado a eventos, mientras que el modelo lógico sólo es conocido por el 25% de los estudiantes.

Tabla 2. Contenidos de los cursos de Programación Imperativa y Programación por Restricciones

Programación Imperativa		Programación por Restricciones	
Unidad	**Temas**	**Unidad**	**Temas**
1. Paradigmas de programación	- Concepto de paradigma de programación - Tipos de paradigmas de programación	1. Análisis matemático	- Introducción a problemas No Polinómicos - Complejidad de Algoritmos, Casos posibles Mejor (Ω) Peor (O) y Medio(θ) - Análisis de soluciones por medio de algoritmos convencionales
2. Programación estructurada	- Entradas/salidas - Condicionales - Ciclos - Arreglos - Matrices - Listas	2. Modelamiento CSP	- Modelamiento de un CSP - Implementación de un CSP en utilizando un motor de restricciones - Implementación de aplicaciones bajo restricciones utilizando ejercicios conocidos. (SendMoreMoney, grocery, etc.)
3. Interfaz Gráfica de Usuario	- Elementos básicos - Modo gráfico con qtk - Especificaciones declarativas de widgets	3. Motores de Restricciones	- Codificación de aplicaciones con Mozart (Oz).
4. Programación Orientada a Objetos	- Fundamentación - Orientación a Objetos - Clase, Objeto e Identidad - Encapsulamiento - Herencia - Polimorfismo - Relación	4. Propagación y Distribución	- Árboles de Búsqueda - Heurísticas de Distribución - Análisis de propagadores

Considerando que el modelo funcional no se aborda en ninguno de los cursos de programación, se indagó con aquellos que habían manifestado conocerlo y se encontró que habían formado parte de un grupo que en el año 2009 participó en una investigación sobre la enseñanza de la programación funcional con Scheme (Timarán et al., 2009a; 2009b y

2010). Mientras que quienes conocen el modelo de programación lógica es porque lo han estudiado para solucionar ejercicios en los cursos de sistemas expertos e inteligencia artificial.

En la post-evaluación, el 100% de los estudiantes manifestaron conocer los fundamentos teóricos y aplicar técnicas de los modelos de programación: estructurada, orientado a objetos y funcional. Todas las ventajas del modelo de programación funcional fueron reconocidas por los estudiantes, el 57% señaló la transparencia referencial y el 43% la capacidad de trabajar con datos potencialmente infinitos y el 28,6% marcó que los programas son fáciles de entender. El 71,4% reconoció que la programación funcional forma parte del paradigma declarativo. De esta manera, se evidencia que el curso en cuestión contribuyó significativamente al conocimiento del paradigma funcional y al mejoramiento de la capacidad para solucionar problemas de programación.

Cabe destacar que al comenzar el curso de programación funcional los estudiantes manifestaron su deseo de continuar estudiando y aplicando el modelo y lenguaje de programación que más conocían y presentaron resistencia a comenzar a programar desde una perspectiva diferente. Esto confirma que los estudiantes se inclinan y tienden a conformarse con el primer modelo y lenguaje que aprenden. De esto se colige que es muy importante la decisión sobre cuál modelo y qué lenguaje utilizar en los primeros cursos de programación y más importante aún es crear en los estudiantes la disposición mental para aprender y aplicar el enfoque multiparadigma.

3.3 Fase 3. Programación por restricciones

El objetivo de la tercera fase de esta investigación fue el apropiar y aplicar la programación por restricciones con Mozart-Oz en el curso de Seminario de Computación e Informática del X semestre del programa de Ingeniería de Sistemas. El curso se programó para 16 semanas con una intensidad semanal de cuatro horas y distribuido en cuatro unidades cuyas temáticas se muestran en la tabla 2.

Con el fin de determinar el conocimiento previo sobre conceptos de programación con restricciones, se diseñó una prueba diagnóstica compuesta por 10 preguntas sobre los temas más importantes de este modelo de programación y se aplicó al grupo experimental.

De acuerdo con los resultados de la pre-prueba de la tabla 3, se evidencia que el grupo de estudiantes obtiene porcentajes altos en las columna de respuestas incorrectas y no contestadas en aquellas preguntas que tienen que ver con los conceptos de la nueva temática de programación por restricciones tales como *Prunning*, *Branch* and *Prune* y *BoundConsistency*, por desconocimiento de estos temas. Al

finalizar el semestre se realizó la post-evaluación, aplicando el mismo instrumento de la pre-evaluación para determinar los conocimientos adquiridos por el grupo experimental durante el curso de programación por restricciones.

Tabla 3. Resultados pre-evaluación y post-evaluación del curso de programación por restricciones

N°	Pregunta	% correctas		% incorrectas		% sin respuesta	
		Pre	Post	Pre	Post	Pre	Post
1	Un paradigma de programación es…	82.5	91.3	13.3	8.7	4,2	0.0
2	Narrowing es una técnica utilizada para…	2.1	78.6	13.8	10.2	84.1	11.2
3	Pruning es una técnica utilizada para…	3.0	81.3	8.2	6.1	88.8	12.6
4	Branch and Prune es…	0.0	71.9	3.2	28,1	96.8	0.0
5	Un lenguaje de programación por restricciones es…	30.5	87,2	11.1	12,8	58.4	0.0
6	Un espacio de búsqueda es…	39,6	73.3	34.5	26.7	25.9	0.0
7	Un procedimiento es…	41.1	82.0	44.7	18.0	14.2	0.0
8	Una función objetivo es…	10.3	68.9	60,5	21.6	29.2	9.5
9	Una propagador es…	23.4	77.7	62,2	22.3	14.4	0
10	BoundConsistency es…	5.1	58.8	0.0	36,5	94.9	4.7

Al establecer la relación entre los resultados de la pre-evaluación y la post-evaluación del grupo experimental se obtuvieron los resultados de las preguntas correctas, incorrectas y sin respuesta (figura 1). Comparando los resultados de la pre-evaluación y post-evaluación de los estudiantes del grupo experimental, se evidencia que la columna de respuestas correctas cuenta con porcentajes altos en la mayoría de ítems, y en ninguno de los casos desmejoraron el resultado.

Las preguntas cuyo porcentaje mejoró considerablemente fueron: La pregunta 2 que evalúa el concepto de Narrowing que subió del 2,1% en la pre- evaluación al 78.6% en la post evaluación; de igual manera la pregunta 3 sobre Pruning subió del 3% al 81,3%; la pregunta 4 que pregunta sobre Branch and Prune subió del 0% al 71,9% y la pregunta 10 sobre BoundConsistency subió del 5,1% al 58,8%.

El tratamiento aplicado al grupo experimental afianzó los conocimientos sobre los conceptos evaluados en la pre-prueba sobre programación por restricciones y por lo tanto disminuyo notablemente el porcentaje de preguntas incorrectas en la post-prueba especialmente en los conceptos evaluados en las preguntas 7, 8 y 9. La pregunta 10 en la pre-evaluación aparece como 0% debido a que el 94.4% del grupo no la contestó.

De igual manera, se puede afirmar que después del curso de Programación por Restricciones, los estudiantes del grupo experimental redujeron en su totalidad la incertidumbre que tenían sobre algunos conceptos de programación al dejar muy pocas preguntas sin responder.

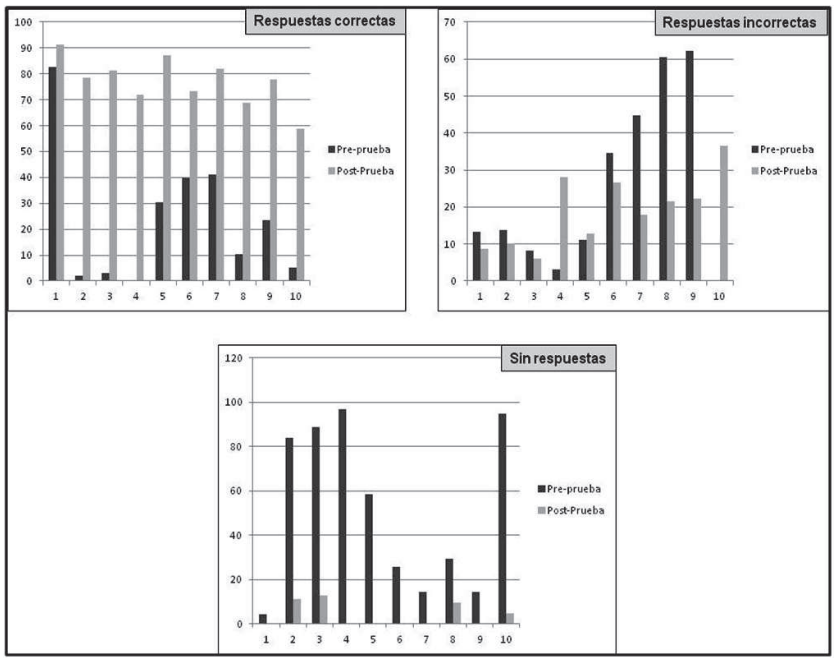

Figura 1. Relación entre las respuestas de la pre-prueba y post-prueba

El proceso de evaluación del curso de Programación por Restricciones fue permanente atendiendo a tres criterios: conocimiento de las temáticas, aplicación apropiada en la solución de problemas y participación activa en el desarrollo del curso. Los resultados finales de los estudiantes del grupo experimental se muestran en la tabla 4.

Tabla 4. Resultados finales del curso Programación por Restricciones

Estudiantes		Evaluación	
Cantidad	**%**	**Categoría**	**Rango**
6	19.4	Excelente	5.0
6	19.4	Muy bueno	4.5-4.9
8	25.8	Bueno	4.0-4.4
9	29.0	Satisfactorio	3.0-3.9
2	6.5	Malo	2.5-2.9
0	0.0	Insuficiente	0.0-2.4

De acuerdo a los resultados finales de la evaluación del curso de programación por restricciones, el 64.6 % de los estudiantes obtuvieron una nota de nivel superior, el 29% obtuvieron una nota de nivel medio y únicamente el 6.5% obtuvo una nota de nivel bajo. En general el

promedio del curso es de nivel superior con una nota cuantitativa de 4.1/ 5.0. Esto demuestra que los estudiantes del grupo experimental apropiaron con suficiencia la programación por restricciones.

3.4 Discusión de resultados

Teniendo en cuenta los resultados de cada fase de esta investigación, se puede afirmar que el tratamiento experimental aplicado en cada una de ellas, generó buenos resultados en el manejo de las temáticas estudiadas durante los tres semestres académicos a tal punto que los resultados de las post-pruebas del grupo experimental, después del tratamiento experimental aplicado, son superiores para los ítems evaluados a los resultados encontrados en las pre-pruebas.

Analizando los resultados de cada fase, la primera, permitió a los estudiantes afianzar los conocimientos sobre la programación estructurada y orientada a objetos que ya tenían en otros cursos, pero esta vez con un nuevo entorno y lenguaje de desarrollo como lo fue Mozart-Oz en miras de lograr en el estudiante la utilización de varios paradigmas de programación en la solución de problemas.

La segunda fase complementó este proceso en los estudiantes, al ser estos capaces de solucionar un problema utilizando o bien la programación funcional o bien la estructurada u orientada a objetos o combinación de estos en el mismo entorno de desarrollo Mozart-Oz.

Las dos primeras fases facilitaron a los estudiantes el aprendizaje y aplicación de otro modelo de programación como lo fue la programación por restricciones para la solución de problemas utilizando el entorno de desarrollo Mozart-Oz que les es familiar desde la primera fase de esta investigación.

Al finalizar este proyecto, los estudiantes reconocieron que el conocer un modelo de programación multiparadigma, les permitió comprender y concientizarse de que es más fácil programar cuando se selecciona el modelo de programación de acuerdo al problema planteado, de tal forma que el diseño del mismo se hace de forma natural y de que los diferentes paradigmas de programación no son únicos ni opuestos y que por el contrario, son diversos y complementarios en el desarrollo de software.

4. Conclusiones y trabajos futuros

Los resultados de esta investigación constituyen un aporte al conocimiento de la programación multiparadigma y proporcionó información específica que será útil para definir la orientación de otras materias de los

componentes de Programación e Ingeniería del software del programa de Ingeniería de Sistemas de la Universidad de Nariño. Adicionalmente, permitirá que los estudiantes de este programa aprendan utilizar adecuadamente los paradigmas de programación y las herramientas disponibles para mejorar su desempeño en la construcción de software.

Esta investigación fue la primera propuesta, en el ambiente regional, para investigar, aplicar y difundir el modelo de programación multiparadigma. Esto hace que la Universidad de Nariño se presente como pionera en la investigación de modelos avanzados de programación, el estudio de lenguajes no convencionales y la preparación de sus estudiantes para resolver problemas complejos utilizando un lenguaje multiparadigma.

Como trabajos futuros están el realizar un proyecto de construcción de software real utilizando la programación multiparadigma e involucrar el aprendizaje de otros paradigmas de programación dentro del componente de programación del programa de Ingeniería de Sistemas, que en este momento no se estudian.

Bibliografía

APPLEBY, Doris & VANDEKOPPLE, Julius (1998). Lenguajes de Programación: Paradigma y Práctica. México D.F. (México): McGraw-Hill. 492 p. ISBN: 970-10-1945-8.

APT, Krzysztof (2003). Principles of Constraint Programming, Cambridge (UK): Cambridge University Press. 404 p. ISBN: 0-521-82583-0.

ARMSTRONG, Joe; VIRDING, Robert; WIKSTROM, Claes & WILLIANS, Mike (1996). Concurrent Programming in ERLANG. 2. ed. Alvsjo (Sweden): Prentice Hall. 188 p. ISBN: 0-13-508301-X.

CHOCO (s.f.). Java Library for Constraint Satisfaction Problems [on line]. Paris (France): GIP RENATER. http://www.emn.fr/z-info/choco-solver/ [consult: 15/01/2012].

CLÚA, Osvaldo (2004). 75-62 Técnicas de Programación Concurrente II 2004: Mozart-oz [en línea]. Buenos Aires (Argentina): Facultad de Ingeniería de la Universidad de Buenos Aires <http://materias.fi.uba.ar/7562/2004/Mozart-Oz.pdf> [consulta: 20/03/2012]

FERNÁNDEZ, Antonio J. (2005). Programación Declarativa con Restricciones. En: Revista Iberoamericana de Inteligencia Artificial. Vol. 9, No. 27 (jul). Madrid (España): Asociación Española para la Inteligencia Artificial (AEPIA). p. 73-100. ISSN: 1137-3601.

SCHULTE, Christian; TACK, Guido & LAGERKVIST, Mikael Z. (s.f.). Gecode [on line]. Kista (Sweden): Gecode <http://www.gecode.org/presentations/INFORMS%20-%20Gecode.pdf> [consult: 15/01/2012]

GEWALI, Laxmi & MINOR, Jhon (2006). Multi-Paradigm Aproach for Teaching Programming. In: The International Conference on Frontiers in Education: Computer Science & Computer Engineering, FECS 2006, (26-29/06/2006), Las Vegas, Nevada (USA): World Academy of Science. Proceedings, p. 141-146. ISBN: 1-60132-008.

MOZART (s.f.). The Mozart Programming System [on line]. Saarbruecken (Germany): Universitaet des Saarlandes. <http://www.mozart-oz.org/> [consult: 23/05/2011].

PÉREZ, Yanina & LÓPEZ, Lidia (2007). Multiparadigma en la Enseñanza de la Programación. En: IX Workshop de Investigadores en Ciencias de la Computación, WICC 2007, (3-4/05/2007), Trelew (Argentina): Universidad Nacional de la Patagonia San Juan Bosco,

Facultad de Ingeniería - Sede Trelew, Departamento de Informática. Memorias, p.743-747. ISBN: 978-950-763-075-0.

PORKOLÁB, Zoltán & ZSÓK, Viktória (2006). Teaching Multiparadigm Programming Based on Object- Oriented Programming. In: 10th Workshop on Pedagogies and Tools for the Teaching and Learning of Object-Oriented Concepts,TLOOC Workshop, at ECOOP 2006, (3-7/08/2006), Nantes (France): the Computer Science Department of the Ecole des Mines de Nantes (EMN), the University of Nantes (UN). Final Reports, p 146-.156. ISBN: 978-3-540-71772-0.

TIMARÁN, Ricardo; CHAVES, Anivar; CHECA, Juan; ORDOÑEZ, Hugo & JIMÉNEZ, Javier (2009a). Introducción a la Programación con Scheme. San Juan de Pasto (Colombia): Centro de Publicaciones de la Universidad de Nariño. 205 p. ISBN: 978-958-9479-97-1.

TIMARÁN, Ricardo; CHAVES, Anivar; CHECA, Juan Carlos; COLUNGE, Constanza; JIMÉNEZ, Javier & ORDOÑEZ, Hugo (2009b). Un Cambio de Paradigma en la Enseñanza de Fundamentos de Programación en Ingeniería de Sistemas. En Revista Educación en Ingeniería. No. 7 (jun). Asociación Colombiana de Facultades de Ingeniería ACOFI. p 120-128. ISSN: 1900-8260.

TIMARÁN, Ricardo; CHAVES, Anivar; CHECA, Juan Carlos; COLUNGE, Constanza; JIMÉNEZ, Javier & ORDOÑEZ, Hugo (2010). Un Nuevo Enfoque en la Enseñanza de la Programación. San Juan de Pasto (Colombia): Centro de Publicaciones de la Universidad de Nariño. No. 197 p. ISBN: 978-958-9479-98-8.

VAN ROY, Peter & HARIDI, Seif (2004). Concepts, Techniques, and Models of Computer Programming. London (England): The MIT Press. 900 p. ISBN: 958-8162-70-X.

VARGAS, Julio (2004). Programación Concurrente con Restricciones en Mozart. En: Revista Scientia et Technica. Año X, No. 24 (may). Pereira (Colombia): Universidad Tecnológica de Pereira. p. 279-284. ISSN: 0122-1701.