Conciencia Tecnológica No. 37, Enero - Junio 2009

Investigación en el Aula: Uso de un Nuevo Método para la Enseñanza de la Programación Concurrente

Investigación

M.C.C. R. Leticia Villagómez Parra [1], Dr. Rodolfo Trejo Vázquez [2]

[1] Universidad del Valle de México, Campus Aguascalientes. Departamento de Tecnociencias, Blvd. Juan Pablo II Núm. 1144, Col. Loma Bonita, C.P. 20207, Aguascalientes, Aguascalientes. E-mail: letyvp@prodigy.net.mx

[2] Instituto Tecnológico de Aguascalientes, Departamento de Ing. Química y Bioquímica. Av. López Mateos 1801 Ote., C.P. 20256, Aguascalientes, Aguascalientes. E-mail: dr_trejo2004@yahoo.com.mx

Resumen

El objetivo de este trabajo es presentar una nueva metodología de enseñanza de la Programación Concurrente/Paralela, desarrollada por los autores para el curso de Programación Concurrente, que se imparte en la UVM, campus Aguascalientes. En la nueva metodología se hace una justificación del uso de Java, se proponen programas de desarrollo para facilitar el aprendizaje de la programación concurrente como son el NetBeans y JCreator, ya que los alumnos no conocen el lenguaje ni tienen experiencia previa en esta actividad. Además se presenta como introducción una máquina virtual y se instala el sistema operativo Linux Mandriva para que los alumnos se familiaricen con los conceptos de procesos e hilos, fundamentales en la programación concurrente. Con este material se busca que el alumno tenga menos dificultades para enfrentarse a la complejidad del manejo del paralelismo. Para probarlo, se compararon los resultados de aprendizaje en dos grupos de alumnos; uno, usando la nueva metodología y el otro la metodología tradicional. Al final del curso se realizaron dos exámenes, teórico y de programación. Los resultados mostraron que los alumnos que cursaron la materia con el nuevo método lograron dominar más tanto el material teórico como el práctico, así como el software de desarrollo.

Palabras clave: Programación Concurrente, Java, sistemas operativos, hilos, procesos.

Abstract

The objective of this work is to show a new methodology, developed by the authors for teaching Concurrent/Parallel programming, at the Universidad del Valle de México, campus Aguascalientes. The new methodology contains a justification of the use of Java, some development programs are proposed for facilitating the concurrent programming learning, like NetBeans and JCreator, because the students do not know the language and do not have experience in this activity. As an introduction a virtual machine is presented and it is installed the operative system Linux Mandriva Is installed in order to let the students get acquainted with the concepts of processes and threads, which are essentials in concurrent programming. With this material, it is intended that the students have less difficulties at handling the complexity of parallelism. In order to confirm this, learning results of two groups were compared, one used the new methodology and the other the traditional method. At the end of the courses, two exams were applied. One of them was theoretical and the other practical (involving programming). The results showed that the students exposed to the new method obtained a higher grade in both exams and in the software development.

Keywords: Concurrent Programming, Java, Operative Systems, Threads, Processes.

Introducción

Un programa concurrente es aquel que maneja múltiples tareas como si todas estuvieran ejecutándose a la vez [1]. En un sistema de computación con múltiples procesadores se pueden ejecutar varias tareas al mismo tiempo, cada una en su propio procesador. En un sistema con un solo procesador, o donde hay más tareas que procesadores, el sistema puede pasar de una tarea a otra, aparentando que se están ejecutando todas a la vez.

La programación concurrente está relacionada con la programación paralela, pero enfatiza más la interacción entre tareas. Las claves de la programación concurrente son la correcta secuencia de interacciones o comunicaciones entre los procesos y el acceso coordinado de recursos que se comparten por todos los procesos o tareas.

En los últimos años el procesamiento en paralelo ha emergido como una poderosa y nueva rama de la computación. Los avances en tecnología de hardware han hecho posible construir máquinas masivamente paralelas así como en red. Sin embargo, construir aplicaciones paralelas requiere de tareas complejas. Esto es debido al hecho de que los programadores

han contribuido con sincronización de procesos, condiciones de competencia, comunicación entre procesos y equilibrio de carga de trabajo [3]. En general, las dificultades son magnificadas por alguna posible incompatibilidad del hardware y el software.

Las tareas de programación concurrente, que en los sistemas operativos se conocen como comunicación entre procesos, a menudo son implementadas usando hilos (threads), que representan una secuencia simple de instrucciones ejecutadas en paralelo con otras secuencias. Los hilos permiten dividir un programa en dos o más tareas que corren simultáneamente, por medio de la multiprogramación. Los hilos pueden compartir acceso a memoria con otros hilos, y pueden tener los medios para coordinar sus actividades con los otros hilos, con el sistema de computación o con el usuario para lograr el objetivo del programa.

Escribir programas concurrentes requiere de visualizar la solución de un problema de una forma muy diferente a la secuencial, que está fuera de la experiencia de los alumnos. Lograr esta nueva perspectiva puede ser difícil. Es complejo manejar un paquete de muchos hilos, y tal vez como resultado su documentación se convierta en un estilo complicado. Como resultado, programar con hilos es un tema reservado a los programadores destacados. De éstos, no todos aprenden a usar hilos elegantemente o de forma confiable.

En el plan de estudios de la materia no se contempla un prerrequisito importante para cursarla, como sería una materia de Sistemas Operativos, donde el alumno conociera la función y planificación de procesos. Este curso lo estudian posteriormente en el octavo semestre de la carrera, y Programación Concurrente la cursan antes, en el tercer semestre.

La literatura existente a menudo dificulta el aprendizaje a los alumnos, que tienen corta experiencia en programación. Esto se debe a que se omite una introducción al uso de los procesos o a que no se presentan los detalles de los lenguajes de programación ni las plataformas más idóneas para los programas.

La metodología de seguir literalmente el plan de estudios de la materia sin preocuparse por impartir el curso construyendo sobre la base de lo que los estudiantes ya conocen y si carecen de algunas bases necesarias para cursar la materia, homogeneizar el grupo para lograr un aprendizaje significativo, trae consigo la gran desventaja de que los alumnos se quedarán con grandes huecos en el conocimiento y estudiarán sólo para pasar la materia sin llegar a ese aprendizaje significativo que se persigue. La ventaja de esta metodología es que se terminará con todos los puntos de cada uno de los temas del programa de la materia

El objetivo de este artículo es presentar los resultados de la aplicación de una nueva metodología de enseñanza de la Programación Concurrente/Paralela, desarrollada por los autores para el curso de Programación Concurrente, que se imparte en la Universidad del Valle de México, campus Aguascalientes, a alumnos de las carreras de Ingeniería en Sistemas Computacionales y Mecatrónica. La metodología incluyó la preparación de un conjunto de materiales sobre fundamentos teóricos, seguida de cuatro ejemplos ilustrativos, adaptados o programados de manera original por los autores, lo cual resultó altamente motivante y además generó compromiso en el estudiante y en el profesor mismo.

La materia debe presentarse en 45 horas en aula del centro de cómputo.

Con este material se busca que el alumno tenga menos dificultades para enfrentarse a la complejidad del manejo del paralelismo.

Después de implementar el nuevo método se realizaron dos exámenes, uno teórico y otro de programación y fueron comparados con los resultados de un grupo que cursó la materia con el método tradicional. Los resultados arrojan que los alumnos que cursaron la materia con el nuevo método lograron dominar tanto el material teórico como el práctico, así como el dominio del manejo de software de desarrollo.

Fundamentos teóricos

A. Justificación del uso del lenguaje Java

Cualquier lenguaje, natural o informático, tiene un carácter dual: a la vez que es capaz de expresarse, también limita el entorno en el que se aplica dicha capacidad expresiva. Si un lenguaje no permite cierta noción o concepto particular, entonces aquéllos que utilizan el lenguaje no podrán emplear este concepto, e incluso, existe la posibilidad de que desconozcan por completo su existencia.

Pascal, C, FORTRAN y COBOL comparten la propiedad común de ser lenguajes de programación secuenciales. Los programas escritos en estos lenguajes tienen un único hilo de control. Comienzan la ejecución en cierto estado y avanzan ejecutando una instrucción cada vez, hasta que el programa finaliza. El comportamiento del programa puede diferir debido a variaciones en los datos de entrada, aunque para una ejecución concreta del programa existe un único comportamiento. Esto no es adecuado para la programación de sistemas de tiempo real. Según los trabajos precursores de Dijkstra [2], un programa concurrente puede verse como un conjunto de procesos secuenciales autónomos, que son ejecutados (lógicamente) en paralelo. Todos los

lenguajes de programación concurrente incorporan, explícita o implícitamente, la noción de proceso; cada proceso tiene un hilo de control.

Java es un lenguaje de propósito general orientado a objetos establecido con mecanismos para programación multi-hilo [4, 5]. Originalmente fue diseñado para soportar la programación de Internet, pero afortunadamente fue más allá de este dominio. Java es un lenguaje relativamente simple, comparado con C++, y es considerado mucho más seguro que otros lenguajes debido a su recolección automática de basura y carencia de apuntadores en general, entre otras cosas.

A pesar de la relativa simplicidad del manejo de hilos en Java, la escritura de programas concurrentes en Java puede ser aún confusa. En la mayoría de los libros se trata la programación concurrente en un capítulo; introducen al lector con las clases de hilos de Java pero no dan bases conceptuales para programar con hilos efectivamente. Algo que nunca falta en un plan de estudios es una materia de programación en C y C++, por lo tanto, algo que puede hacer más interesante a Java para los estudiantes es que se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Sin embargo, al ser multihilo *(multithread),* Java permite muchas actividades simultáneas en un programa. Los hilos, a veces llamados, procesos ligeros, o hilos de ejecución, son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar estos hilos construidos en el mismo lenguaje, son más fáciles de usar y más robustos que sus homólogos en C o C++.

El beneficio de ser multihilo consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente de la plataforma, aún supera a los entornos de flujo único de programa (single-threaded) tanto en facilidad de desarrollo como en rendimiento.

Cualquiera que haya utilizado la tecnología de navegación concurrente, sabe lo frustrante que puede ser esperar por una gran imagen que se está trayendo de un sitio interesante desde la red. En Java, las imágenes se pueden ir trayendo en un hilo de ejecución independiente, permitiendo que el usuario pueda acceder a la información de la página sin tener que esperar por el navegador.

Metodología

La metodología pedagógica empleada consistió en un compendio de experiencias de aprendizaje con el docente, vinculadas con factores que estimulan el aprendizaje significativo como la forma en que se da la comunicación entre el profesor y el estudiante cubriendo los ámbitos de: *recepción* donde el docente establece una comunicación expositiva, *interpersonal* en el que se propicia un espacio de comunicación bidireccional, de *selección* en el que el docente enfrenta al estudiante a un problema para que sea resuelto por el estudiante mismo y el profesor podrá dar asesoría, y un *proceso de grupo* donde el profesor promueve el trabajo en equipo.

Se diseñó un módulo de fundamentos teóricos, basado en la experiencia de los autores, el cual es impartido en las primeras 3 horas, además de dar una justificación del uso del lenguaje de Java se realizó una práctica donde los alumnos bajaron de la web una máquina virtual [6], Figura 1, e instalaron el sistema operativo Linux Mandriva, Figura 2, con el cual llevaron a cabo una actividad en la que conocieron el funcionamiento de los procesos, herencia, proceso padre y descendencia y cómo se comparten los recursos en el sistema operativo Linux.

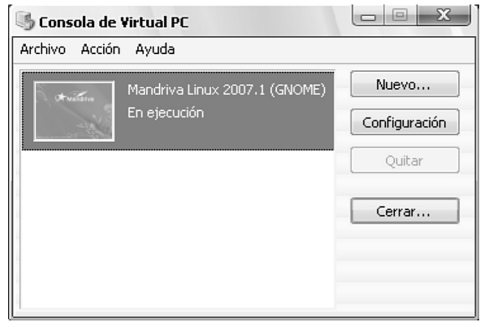


Figura 1. Consola de la Máquina Virtual PC

Figura 2. Sistema Operativo Mandriva Linux 2007.1

En la práctica introductoria del conocimiento de procesos e hilos se trabajó con el comando *top* con una terminal en Linux Mandriva. El programa *top* proporciona una vista dinámica en tiempo real de los procesos que se están ejecutando en el sistema, Figura 3.

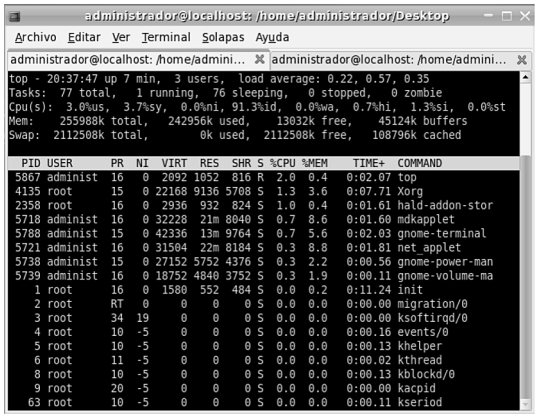


Figura 3. Programa *top* en Linux Mandriva

El programa proporciona una interface interactiva limitada para manipulación de los procesos, así como una interface más extensiva para una configuración personal.

A continuación, se presentaron opciones de software de desarrollo para programar en Java 1.5 horas (JCreator, NetBeans, comandos en línea) una práctica del uso del software de desarrollo, sus ventajas y desventajas 1.5 horas, y cuatro ejercicios de programación (figuras 4 y 5). El tiempo para estas actividades fue de 6 horas.

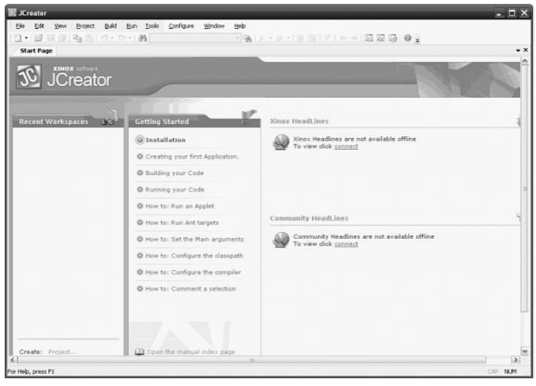


Figura 4. Programa JCreator.

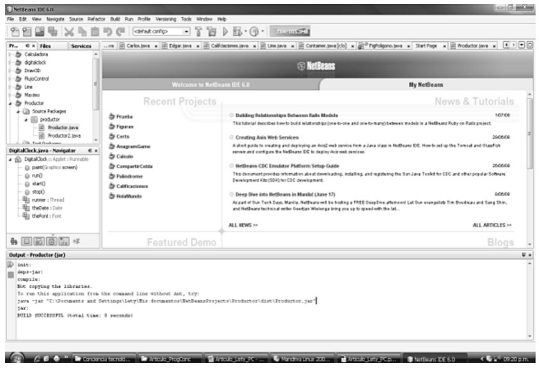


Figura 5. Programa NetBeans 6.0

Finalmente se realizó una evaluación del avance, la cual fue comparada con otra evaluación realizada sobre el mismo tema a otro grupo utilizando los materiales tradicionales [1, 2, 5].

A. Aplicaciones de Concurrencia

Después de la demostración de la programación dinámica de hilos en el sistema operativo Linux, se procede a exponer y dar ejemplos de aplicaciones reales de la concurrencia.

La concurrencia abre las posibilidades de diseño que son imprácticas en la programación secuencial. Los hilos liberan al programador de las limitaciones de código que invocan un método y entonces lo bloquea, no haciendo nada mientas espera por una respuesta. Usando los hilos, se pueden activar nuevas actividades independientes que corren concurrentemente, con o sin esperar su realización. Las razones para explotar el manejo de hilos incluyen [7]:

Programación reactiva: Algunos programas son requeridos para hacer más de una cosa al mismo tiempo, desempeñar cada actividad como una respuesta reactiva a alguna entrada. Por ejemplo en un navegador de www puede estar desempeñando simultáneamente un número de bytes recibidos o alguna imagen, y un diálogo de consulta con el usuario.

Disponibilidad: La concurrencia permite mantener alta disponibilidad de servicios, por ejemplo entre los modelos más comunes de concurrencia, como son servicios de Internet e incluso en muchos applets, permite tener un objeto y servir como una interface para un servicio, manejando cada petición para construir un nuevo hilo que realice asincrónicamente las acciones asociadas. El servicio es capaz de aceptar otra petición rápidamente.

Control: Las actividades con hilos pueden ser suspendidas, continuadas o paradas por otros objetos. Esto proporciona simplicidad y flexibilidad que no se encuentra en la programación secuencial, donde es difícil implementar una secuencia de cancelar una actividad y luego hacer otra.

Objetos Activos: El software orientado a objetos a menudo modela objetos reales. La mayoría de los objetos reales muestran una conducta independiente y autónoma. Por lo menos en algunos casos, la forma más fácil de programar esto es lanzar un nuevo hilo siempre que se genere un objeto.

Mensajes asíncronos: Cuando un objeto envía un mensaje a otro, el primer objeto no siempre cuida en qué momento la acción resultante se ejecuta. Los hilos permiten al primer objeto continuar con su propia actividad sin tener que esperar por acciones no relacionadas con su tarea para concluir.

Paralelismo: En una máquina con procesadores múltiples, la programación concurrente puede ser usada para explotar el poder computacional disponible y mejorar su desempeño.

Concurrencia requerida: Incluso si el programador no quiere explícitamente escribir programas concurrentes, muchas clases predefinidas de java soportan la concurrencia, y las características del tiempo de corrida operan de una forma concurrente. Éstas incluyen las clases java.applet y java.awt para desplegar clips de audio y desplegar imágenes y mecanismos que causan todos los Applets de Java para correr en su propio hilo.

B. Dificultades típicas de la programación concurrente

Algunos problemas que se llegan a dar durante la programación concurrente son:

- ◊ Necesidad de *exclusión mutua:* En este caso los hilos deberán acceder de forma exclusiva a ciertos recursos o zonas de memoria considerados como críticos. En el entendido de que la exclusión mutua no es precisamente un problema, el problema es lograr la exclusión mutua.
- ◊ *Interbloqueos:* tienen lugar cuando ninguno de los procesos en competencia puede continuar su ejecución normal por carecer de alguno de los recursos que necesita.
- ◊ *Sincronización Condicional o Inanición:* este problema tiene lugar cuando la ejecución de un proceso queda siempre pospuesta a favor de algún otro de los procesos en competencia.

Algunos ejemplos de competencia entre procesos:

Exclusión mutua (EM): suponer que dos o más procesos requieren acceso a un solo recurso. Durante la ejecución, cada proceso enviará comandos al dispositivo de E/S, recibirá información de estado, enviará y/o recibirá datos.

Al recurso se le llama *recurso crítico,* y a la parte del programa que lo usa se le conoce como *sección crítica* del programa. ¿Qué pasaría si varios procesos tuvieran control de la impresora mientras se envían varios archivos a impresión? Pues lo único que pasaría es que se imprimiría una mezcla de líneas de uno y otro archivo.

Interbloqueo: Considerar dos procesos, P1 y P2 y dos recursos críticos R1 y R2. Suponer que cada proceso necesita acceso a los dos recursos para ejecutar parte de su función. Entonces es posible tener la situación siguiente:

El sistema operativo asigna R1 a P2 y R2 a P1. Cada proceso espera uno de los dos recursos. Ninguno liberará el recurso que ya le pertenece sino hasta que haya adquirido el otro y ejecutado su sección crítica. Ambos procesos están interbloqueados.

Inanición: Suponer que de tres procesos, P1, P2 y P3 cada uno requiere acceso periódico a los recursos R. Considerar la situación en la cual P1 está en posesión del recurso y P2 y P3 están retrasados, esperando ese recurso. Cuando P1 sale de su sección crítica, a P2 o P3 se le permitirá tener acceso a R. Pensemos en que a P3 se le permite el acceso y que antes de que complete su sección crítica, P1 requiere acceso otra vez. Si a P1 se le permite el acceso después de que P3 ha terminado y si a P1 y P3 se le concede un acceso alternado, entonces a P2 se le puede negar el acceso al recurso en forma indefinida, aun cuando no exista situación de interbloqueo.

C. Requisitos para lograr la Exclusión Mutua (EM).

Cualquier sistema que va a proporcionar soporte para exclusión mutua debe cumplir los siguientes requisitos:

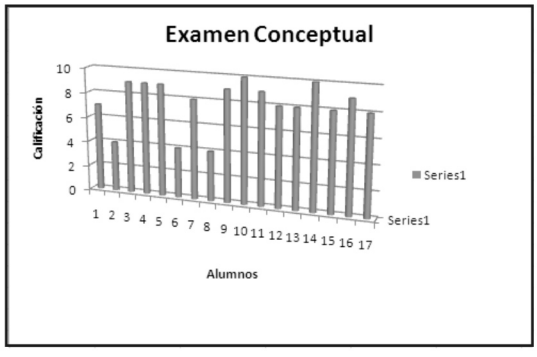
- ◊ La EM debe reforzarse: sólo se permite un proceso a la vez dentro de su sección crítica, entre todos los procesos que tienen secciones críticas para el mismo recurso u objeto compartido.
- ◊ Un proceso que se detiene en su sección no crítica debe hacerlo sin interferir con otros procesos.
- ◊ No debe ser posible que un proceso que requiere acceso a una sección crítica se retrase en forma indefinida; no pueden permitirse el interbloqueo o la inanición.
- ◊ Cuando ningún proceso está en una sección crítica, a cualquier proceso que solicite entrada a su sección crítica se le debe permitir entrar sin retraso.
- ◊ No se suponen velocidades de proceso relativas o número de procesos.
- ◊ Un proceso permanece dentro de su sección crítica sólo durante un tiempo finito.

Resultados y discusión

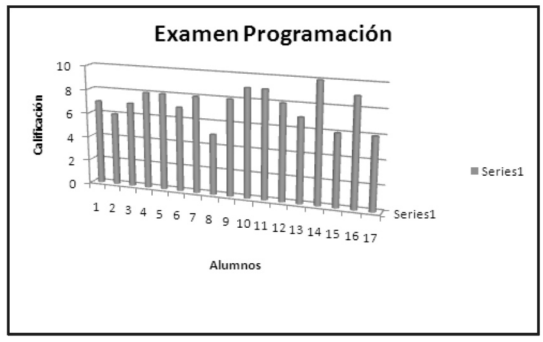
Al término de la exposición del material con el nuevo método, se realizó una evaluación consistente en un examen conceptual y un examen de programación concurrente. Los resultados se compararon con los obtenidos por otro grupo, que aprendió por el método de enseñanza tradicional y respondió el mismo examen (Tabla 1). En las gráficas 1 a la 4 se puede ver las calificaciones individuales de los alumnos de ambos grupos.

	Grupo Método Tradicional	Grupo Método nuevo
Número de alumnos	17	6
Promedio de Calificación en Examen Conceptual	7.82	9.66
Desviación Estándar del Examen conceptual	1.33	0.82
Promedio de Calificación en Examen programación	7.53	9.5
Desviación Estándar del Examen de Programación	1.95	1.22

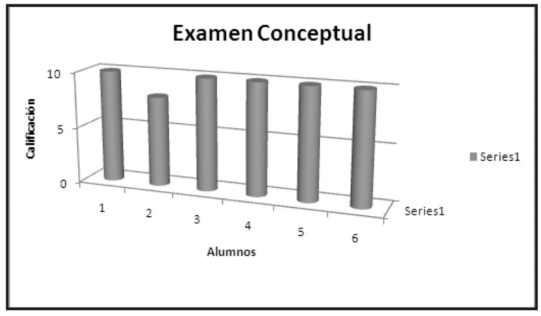
Tabla 1. Resultados de las evaluaciones.



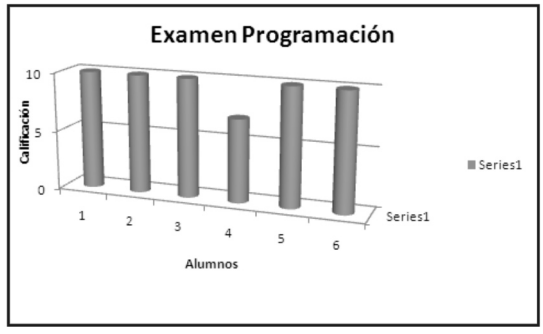
Gráfica 1. Examen conceptual método tradicional



Gráfica 2. Examen programación método tradicional



Gráfica 3. Examen conceptual método nuevo



Gráfica 4. Examen programación método nuevo

Resulta evidente que el grupo que utilizó el método nuevo obtuvo mejores notas, tanto en la parte teórica como en la de programación. Los estudiantes manifestaron haber tenido facilidad para entender los conceptos y para realizar las prácticas, en comparación con las opiniones de los alumnos del grupo bajo el método tradicional.

Conclusiones

El curso de programación concurrente que se imparte en la Universidad del Valle de México es un cimiento indispensable para los alumnos de las carreras de Ingeniería en Sistemas Computacionales y Mecatrónica. Uno de los problemas detectados es la complejidad de entender el concepto de concurrencia y las pocas bases que los alumnos tienen de programación para resolver problemas concurrentes;

esto se debe a que este curso lo estudian en el tercer semestre de la carrera. Para hacer menos complicado y más dinámico el curso, se instaló una máquina virtual y el sistema operativo Linux Mandriva, con el fin de exponer los conceptos de procesos, hilos, concurrencia y otros más. Además se hizo una práctica más en el laboratorio de cómputo para el uso del lenguaje de programación Java con los programas de desarrollo NetBeans 6.0 y JCreator, y se hicieron cuatro programas introductorios.

Después de realizar la exposición teórica y hacer las prácticas de laboratorio se aplicaron dos exámenes, un examen teórico y uno más de programación, estos resultados fueron comparados con los exámenes aplicados a un grupo que cursó la materia con el método tradicional, el cual consiste en seguir literalmente el programa de la materia sin preocuparse por impartir el curso construyendo sobre la base de lo que los estudiantes ya conocen y si carecen de algunas bases necesarias para cursar la materia, homogeneizar el grupo para lograr un aprendizaje significativo.

Los alumnos demostraron tener el dominio conceptual y también el de la programación concurrente, además en la evaluación del curso teórico se les pidió opinión sobre el software de desarrollo, en esta pregunta expresaron agrado por las herramientas. Esto aceleró el avance del plan de estudios y fue posible resolver más problemas concurrentes, incluso dar una introducción a aplicaciones en la Inteligencia Artificial como redes neuronales, algoritmos genéticos, computación gráfica y procesamiento de imágenes.

Como trabajo futuro se planea agregar algunas estrategias de enseñanza-aprendizaje ya que no basta que el alumno aprenda sobre el tema, lo interesante es que cada vez lo haga mejor y se transforme el conocimiento logrando realizar aplicaciones reales, participando en concursos como competencias tecnológicas entre estudiantes, por ejemplo la Imagine Cup, y seguir comparando resultados con cursos impartidos anteriormente.

Referencias

[1] D. Lea, (1999), *Concurrent Programming in Java. Design Principles and Patterns*. New York: Addison-Wesley, pp. 2-8.

[2] Deitel y Deitel, (1995), *Cómo programar en Java.* Naucalpan, Mex: Pearson Educación, pp. 670-700.

[3] J. Niemiec. CC++, pC++, Charm++ and Orca: Languages for Parallel Programming. CIS Department, Syracuse University Science and Technology. December 22, 1993. Documento de Internet http://citeseer.ist.psu.edu consultado el 28 de Abril de 2007.

[4] K. Havelund, T. Pressburger, (2005), "Model Checking Java Programs Using Java PathFinder". In *Lecture Notes in Computer Science,* Springer Berlin, Ed. Heidelberg, pp. 444-456.

[5] Tutorial de Java in Technology. Documento de Internet http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte2/cap2-5.html. Consultado el 28 de Abril de 2007.

[6] Equipos Virtuales para Microsoft Virtual PC (2007). Documento de Internet http://www. ofgsoftware.com/web/virtualizacion/virtualpc2007/virtualpc2007_equiposvirtuales.html. Consultado el 2 de Julio de 2008.

[7] A. Wellings, (2004). *Concurrent and Real-Time Programming in Java.* West Sussex, England: Wiley.

Artículo recibido: 8 de diciembre de 2008

Aceptado para publicación: 8 de mayo de 2009