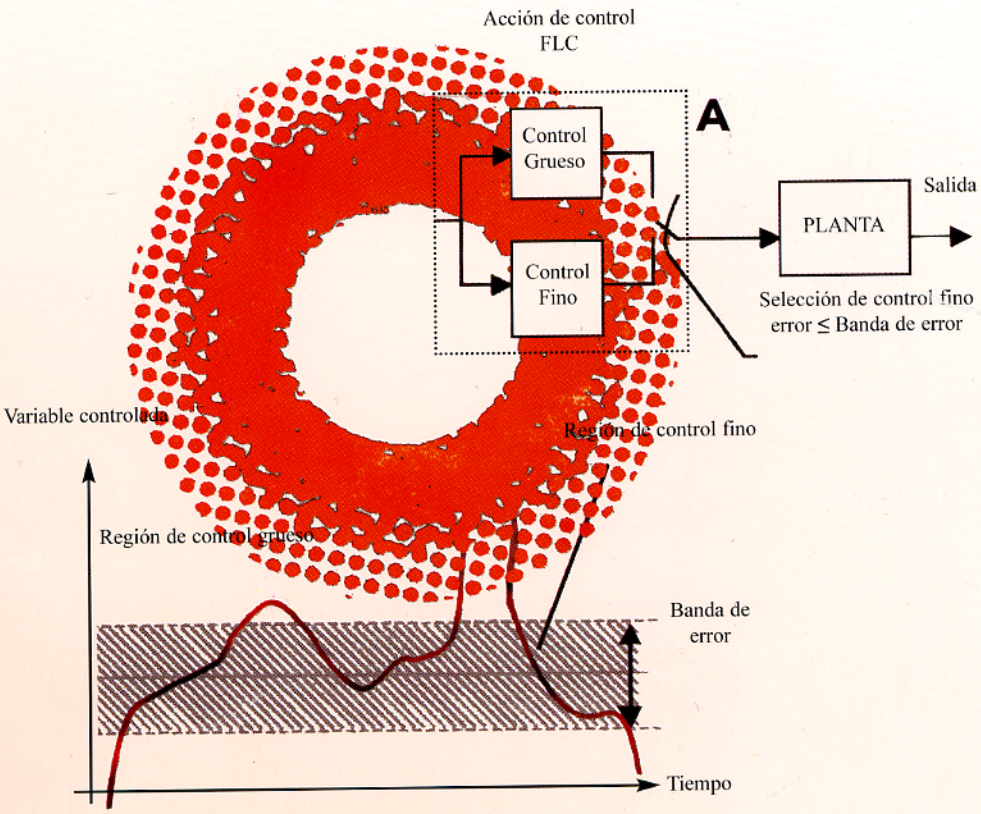
ISSN 1409-2441

Ingeniería

Revista de la Universidad de Costa Rica
Enero/Diciembre 1999 VOLUMEN 9 Nos. 1 y 2



INGENIERIA

Revista Semestral de la Universidad de Costa Rica
Volumen 9, Enero/Diciembre 1999 Números 1 y 2

CORRESPONDENCIA Y SUSCRIPCIONES

Editorial de la Universidad de Costa Rica
Apartado Postal 75
2060 Ciudad Universitaria Rodrigo Facio
San José, Costa Rica

CANJES

Universidad de Costa Rica
Sistema de Bibliotecas, Documentación e Información
Unidad de Selección y Aquisiciones-CANJE
Ciudad Universitaria Rodrigo Facio
San José, Costa Rica

Suscripción anual:
Costa Rica: ¢ 1 000,00
Otros países: US $ 30,00

Número suelto:
Costa Rica: ¢ 750,00
Otros países: $ 20,00

Ingeniería 9(1): 155-165, 1999 San José, Costa Rica

ENSEÑANZA DE CURSOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS PARA LOS PRINCIPIANTES

Elzbieta Malinowski G.[1]

Resumen

Aunque las universidades en todo el mundo incorporan el paradigma de orientación a objetos en su curriculum, no hay seguridad de que se deba enseñarlo como primer curso de programación. El presente artículo menciona en forma breve las metodologías usadas en la enseñanza de la tecnología de orientación a objetos. Se describen algunos factores que influyen en la introducción exitosa de este tipo de programación. Además, se presenta la experiencia y se enumeran las dificultades que encontró el autor de este artículo para impartir, durante varios semestres, a los novatos el curso de programación orientada a objetos.

Summary

Even though many universities in the world used the object-oriented paradigm in their curriculum, still there is not unified criteria about teaching object-oriented programming as a first programming language. The present article mentions briefly about methodologies used in teaching object-oriented techniques. It describes some features that influence success in the introduction of this type of programming. Moreover, this article describes the experience and enumerates some difficulties that author of this article found in teaching of object-oriented programming to beginners.

1. INTRODUCCIÓN

Actualmente, la programación orientada a objetos (POO) es indiscutiblemente un paradigma dominante de programación no sólo en las universidades, sino también en el sector industrial. El proceso evolutivo que dio como resultado este tipo de paradigma comenzó con los lenguajes poco estructurados, tipo BASIC, FORTRAN, siguiendo con aquellos que introducían en forma eficiente los principios de programación estructurada como PL/1, ALGOL, PASCAL. Otros tipos de lenguajes como los modulares, de programación funcional o lógica, que se usan actualmente para resolver problemas específicos, no encontraron una amplia aceptación, como se esperaba en el momento de su aparición.

Sin embargo, conforme crecen los requerimientos de los usuarios sobre el *software*, el tamaño de los programas crece también. El *software* escrito, usando el paradigma de programación estructurada, gasta 67% en el mantenimiento del total de su costo [SOMM92]. Además, el diseño se vuelve un trabajo difícil y la necesidad de trabajar en grupos también es engorrosa por la misma estructura de los programas. Según Booch, el *software* de "dimensión industrial" debe ser desarrollado usando la POO que por sus características de reutilizacón de código, polimorfismo, herencia, encapsulación, entre otras, se pueden manejar en forma más eficiente los aspectos de la complejidad del sistema tanto en el nivel de diseño como en el nivel de desarrollo [BOOC96].

Aunque está bien justificada la utilidad de análisis, diseño y programación orientadas a objetos [BOOC96, JOVA96, KELL96, GAH96] y muchas universidades la incorporaron en el curriculum de su carrera [GIBB96, LIND96, MANN96, MCKI96], no existe todavía la claridad sobre si es o no recomendable impartir el curso de POO al principio de la carrera a los estudiantes sin ninguna experiencia en la programación.

[1] Ing. M.Sc. Prof. Esc. Ciencias Comp.e Informática Univ. de Costa Rica

El presente artículo resume en forma breve las metodologías usadas en la enseñanza de los cursos de la tecnología de orientación a objetos (TOO). También, se presentarán los aspectos que pueden afectar el desarrollo exitoso del primer curso de programación donde se usa el paradigma de orientación a objetos. Al final se describe la experiencia que tuvo la autora de este artículo durante la enseñanza del primer curso de programación usando POO para los estudiantes del primer ingreso en la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica. Se finalizará el artículo con una descripción de las dificultades presentes durante el desarrollo de este tipo de curso.

2. BREVE DESCRIPCIÓN DE LAS METODOLOGÍAS USADAS EN LA ENSEÑANZA DE LA TECNOLOGÍA ORIENTADA A OBJETOS

En general, la enseñanza de TOO se basa en uno de los dos enfoques presentados por J.C.McKim: *breadth-first* y *depth-first* [MCKIM96]. El primero se caracteriza por ofrecer una amplia variedad de tópicos relacionados con la TOO, donde los estudiantes al finalizar el curso son capaces de leer y entender artículos relacionados con la misma, compartir la información con los profesionales avanzados del área y reconocer los conceptos avanzados sobre TOO [MANNS96]. Sin embargo, como la idea principal es desarrollar en el estudiante una visión amplia sobre TOO, este tipo de cursos se realiza sin depender de ningún lenguaje de programación. La carencia de éste hace imposible fortalecer el curso con la parte práctica.

Otro enfoque de *depth-first* en general se basa en algún lenguaje de programación orientado a objetos (POO) y desarrolla en los estudiantes conocimientos sobre TOO basándose en este lenguaje y profundizando su conocimiento. Así los estudiantes pueden aprender y también practicar muchos de los conceptos de TOO. Sin embargo, la desventaja de este enfoque es que los estudiantes pueden tener una visión sesgada sobre el concepto del mismo basado en un solo lenguaje POO [MANNS96].

Según opiniones, el enfoque de *breadth-first* debe ser usado en los cursos más avanzados de la carrera de computación y *depth-first* al inicio de la carrera [WU93, MANNS96]. También, de acuerdo a T.Wu, en este último se pueden considerar diferentes formas de aprender la POO: *top-down* y *bottom-up* [Wu93]. El primero comienza con la visión global de lo que son los lenguajes de programación, entre otros, conceptos de objetos, herencia, polimorfismo, funciones virtuales y después presenta los detalles de implementación. La segunda forma, de *bottom-up,* adopta la pedagogía de "aprende haciendo", donde primero se presenta el problema, después se proporciona la solución al mismo y se hace el adecuado análisis. Según T.Wu, la primera forma es más apta para los estudiantes avanzados en programación debido a que para los novatos sería demasiado abstracta y poco clara [WU93]. El autor anteriormente mencionado recomienda para estos estudiantes usar el enfoque de *bottom-up.*

De la misma forma el reconocido creador del lenguaje C++, B.Stroustrup, apoya este enfoque cuando recomienda el desarrollo del curso de lo concreto a lo abstracto. Sin embargo, el recomienda tener cuidado de que los detalles sintácticos y semánticos del lenguaje no se vuelven un aspecto importante de ningún curso y tanto los programadores novatos como los programadores experimentados deberían enfocar en los conceptos y las técnicas, que permiten hacer los programas más eficientes. Esto se puede lograr usando un ejemplo bien escogido donde se presenta nuevos conceptos dentro de un contexto [STRO99].

Aunque Stroustrup recomienda el uso de bibliotecas y clases desde principio, de acuerdo a la experiencia presentada por Gibbon et al., los estudiantes de nivel de pregrado encuentran las dificultades en el proceso de aprendizaje de los conceptos relacionados con la POO [GIBB96]. Para

facilitar este proceso se necesita el apoyo dirigido del profesor, quien usando las técnicas simples, ayuda a los estudiantes a encontrar los objetos "buenos" y desarrollar las heurísticas para mejorar el proceso de diseño.

Aunque el curso de POO forma parte del curriculum en la mayoría de las universidades, existen muchos factores que influyen en la decisión si es o no recomendable impartir el curso de POO a novatos.

3. FACTORES INFLYUENTES A LA ENSEÑANZA DE PROGRAMACIÓN ORIENTADA A OBJETOS A LOS PRINCIPIANTES

La enseñanza de POO no es solo un aspecto de gusto. Ahí se debe considerar que se está proponiendo el cambio de paradigma de analizar y de diseñar. Cualquier cambio de este tipo encuentra resistencias. Así, algunos profesores opinan que el estudiante novato no es capaz de asimilar este concepto por ser muy abstracto; otros dicen que como los estudiantes no saben nada sobre programación pueden asimilar todo lo que se les presenta, solo si esto se hace en una forma adecuada. Sin embargo, se presentan los aspectos más importantes necesarios a considerar para la introducción exitosa de POO:

- Maestros de cambio: el cambio ocurre porque crece la insatisfacción con la tecnología actual. Sin embargo, el cambio no siempre ocurre por las razones de los factores internos (recursos, comportamiento, actitud, sentimientos, miedo, falta de reconocimiento, etc) y factores externos (normas culturales, razones políticas). Se necesitan personas que tengan la motivación para tal propósito, algún grado de influencia y que sean capaces de cambiar aspectos dentro de la organización. También, deben tener una visión del futuro y ser capaces de enfrentar la resistencia siempre presente cuando se realiza cualquier transformación.

- Profesores que dominan conceptos de TOO y algún lenguaje que sirve para la explicación de los conceptos de POO y están dispuestos a servir de pioneros: no siempre se dispone del personal altamente calificado para la tarea propuesta. Además, se necesita que el profesor disponga del tiempo extra para incorporar estos nuevos conceptos. Muchas veces tendrá que desarrollar los programas necesarios, preparar toda la estructura del curso, tomar la decisión cuál enfoque usar, entre otros aspectos. Las tecnologías nuevas no tienen establecida la metodología de enseñanza, no existen libros de texto que pueden respaldar la materia vista en clases. La preparación del material es responsabilidad del profesor, lo cual implica más trabajo.

- La selección del lenguaje de programación: ésta puede tener consecuencias significativas respecto a si el estudiante percibe la POO y si los profesores pueden hacer sus cursos flexibles de acuerdo a su estilo de enseñanza. Aunque es claro que el curso de POO no debe ser la enseñanza de sintaxis del lenguaje, sin embargo, el lenguaje de programación debe incluir los conceptos y técnicas enseñadas en clases. No importa la metodología aplicada en el curso, el lenguaje de programación usado tendrá una influencia enorme en el desarrollo de las habilidades de los estudiantes para construir programas bien estructurados [LIND96]. En general, se debe considerar si en la enseñanza de cursos de POO usar un lenguaje puro de POO, como Smalltalk, o lenguaje híbrido como lo es C++. Esta discusión está abierta y en la práctica se usa toda la variedad de lenguajes, como Eifel, C++, Smalltalk [MANN96], BETA [LIND96], C++ [GIBB96, MALL96]. Cada uno tiene sus ventajas y desventajas. Muchos autores, especialmente de libros de texto donde se enseñan técnicas de POO, recomiendan primero aprender un

lenguaje de programación estructurada como por ejemplo C y solo después pasar a POO usando C++. La opinión contraria es presentada por B.Stroustrup el cual expresa que el lenguaje C no se debe considerar como primer subconjunto del C++. De acuerdo a su experiencia, el lenguaje C enfoca mucho en los aspectos de los detalles propios de la implementación y así oscurece los aspectos del estilo de programación. Un curso de programación que podría ser interesante y motivador para el estudiante se vuelve en curso tedioso, para enfrentar los detalles de sintaxis.

- Recursos de *hardware* necesarios: la decisión de cuál lenguaje usar como primero está estrechamente relacionado con la disponibilidad de *hardware*. Este problema se hace más grande, si se dispone de recursos económicos limitados. Usar la herramienta que tiene amplias bibliotecas estándar de clases y que es posible programar en el ambiente gráfico, es un aspecto motivador. Esto es posible usando los compiladores más poderosos que necesitan adecuado equipo computacional.

- Entendimiento del cambio que se da cuando existe la insatisfacción con la tecnología actual: aunque puedan existir los maestros de cambio, necesitan sus seguidores, personas que entienden la necesidad de cambio, no se resisten a él, justificando la lealtad a lo viejo con toda gama de argumentos.

- Apoyo bibliográfico en la enseñanza: se debe tomar en cuenta que el estudiante que aprenda a programar usando orientación a objetos debería poder contar con algún libro de texto que en forma simple enseña los conceptos de POO y así le permite reforzarlos fuera de las lecciones. Aunque existen muchos candidatos a ser libros de texto, no siempre el enfoque que dan es adecuado. En la mayoría de libros se necesita al principio aprender o ya conocer la programación estructurada y después POO. Las clases se presentan como conceptos avanzados que dan al estudiante la imagen para programar usando los objetos, la cual es difícil [DEIT95, DEIT99, HAID98, STAU98]. En consecuencia, el estudiante, especialmente de primer ingreso, al no tener acceso a ningún libro de texto, siente la inseguridad de que si no aprende durante las lecciones, después le es más difícil asimilar los conceptos por falta de bibliografía necesaria.

Tomando en cuenta las dificultades mencionadas anteriormente y conociendo las decisiones que tomaron algunas universidades sobre el año de la carrera en que debe enseñarse la POO, en forma implícita se promueve la opinión de que no es recomendable comenzar a enseñar la programación usando POO [MCKIM96, MANNS96]. La justificación de esta posición se puede basar en los aspectos mencionados y además, entre otros, falta de experiencia en metodología de enseñanza de POO, falta de unificación de criterios de usar el enfoque puro de POO o híbrido, falta literatura adecuada para principiantes, demasiado alto nivel de abstracción, etc.

Aunque no es fácil la promoción del cambio, desde 1998 en la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica se imparte POO en el primer curso de programación. Los profesores que lo imparten cuentan con diferentes experiencias y es importante que éstas se compartan, para el futuro mejoramiento de dicho curso.

4. PRESENTACIÓN DE LA EXPERIENCIA DE IMPARTIR EL CURSO DE PROGRAMACIÓN ORIENTADA A OBJETOS A LOS NOVATOS

El primer curso de programación es de suma importancia, porque es un curso formativo. Si

el estudiante desarrolla malas técnicas de programación será muy difícil corregirlas en los cursos posteriores. Esta responsabilidad está puesta en el profesor que imparte el curso. El autor de este artículo considera como su responsabilidad compartir con los compañeros de trabajo la experiencia de impartir el curso usando la POO a los novatos

El autor de este artículo impartió el curso de POO como primer curso de programación varias veces. La primera vez el aspecto importante para considerar fue, ¿cual lenguaje de programación usar?.

Como se pudo notar anteriormente, en general no existe una opinión unificada sobre este aspecto. De la misma forma como en otras universidades, las discusiones sobre la elección del lenguaje de programación en la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica se convirtieron en las discusiones más a fondo: se puede o no enseñar POO sin tener conocimientos previos de programación estructurada. Como no existe la respuesta única a esta pregunta se tomó la decisión de usar el lenguaje C++ que permite usar los dos paradigmas y así asegurará a los profesores la libertad de escoger la metodología de enseñanza.

Además, los profesores que imparten el segundo curso de programación usando el lenguaje C++ presentaron argumentos adicionales a su favor. De acuerdo a ellos, si en los dos cursos de programación se usan los lenguajes diferentes, se necesitan 4 o 5 semanas para explicar el nuevo sintaxis provocado por cambio de lenguaje. Usando el mismo lenguaje, este valioso tiempo se podría aprovechar ampliando el segundo curso de programación por nuevos conceptos de POO. Esta elección de lenguaje de programación era una tarea compartida por varios profesores. A partir de aquí cada uno de ellos tomó su decisión sobre la metodología de enseñanza usada.

Se conoce desde un punto de vista metodológico que la mejor forma de aprender los conceptos nuevos es unirlos en forma de espiral con las que ya tenemos asimilados. Se puede ver este proceso como el aprendizaje acumulativo, donde asimilación es la parte más importante que la propia acumulación. Asimilación se da cuando el concepto se hace parte de la persona y esto se da, si uno puede vivir las experiencias donde puede aplicar los nuevos conceptos. Viendo el aprendizaje de esta forma es importante analizar en qué orden se ve los conceptos en el primer curso de programación y cómo se hace su unión para asegurar la asimilación de los conceptos nuevos por medio de las prácticas adecuadas y así asegurar el crecimiento paulatino de nivel de abstracción.

La autora de este artículo, dando el curso por primera vez, introdujo desde el principio el concepto de clases y enseñó usar las estructuras de control (condicionales, repetitivas) como parte de los métodos de las clases. Las clases se presentaron como elementos ya definidos. Sin embargo, como se puede observar de la primera parte de este artículo, el propio concepto de clase es un elemento bastante abstracto para ser introducido desde el principio. Por esta razón, muchos de los estudiantes encontraron difícil la POO. Esto les dificultó la apertura hacia experimentos, pruebas de programas cortos, una especie de "juego" con el compilador para acostumbrarse al ambiente, a los simples errores de sintaxis, etc.

La segunda vez se cambió la metodología. Se les explicó a los estudiantes la idea general de los objetos basándose en la vida real, sin comentar sobre los detalles de su implementación en los lenguajes de programación. Sin embargo, el primer reto que se encontró en este curso fue cómo cambiar la forma de pensar de funciones a objetos. Desde las formaciones tempranas en el nivel educativo los alumnos separan las características de los objetos de su comportamiento, por ejemplo en el campo de las matemáticas, cuando se introducen los

conceptos de los números naturales, enteros, racionales, etc., se describen sus características, pero no se les abarca como objeto que además de tener sus propiedades, tiene asignado su conjunto de operaciones. En la vida real se manejan los objetos en forma implícita. Cuando la persona va al banco sabe que significa el retiro de dinero y no lo confunde con el retiro de la persona de su trabajo. Aquí implícitamente se ubica el objeto (cuenta o persona) y se entiende el significado diferente de la función de retiro. Cuando alguien nos dice: llámame, en nuestra mente no hay duda que la operación de llamar es aplicada al teléfono. Como se puede ver, se acostumbra mencionar explícitamente la función e implícitamente ubicar el objeto. Sin embargo, para programar se necesita hacer la conciencia de todos los elementos que abarca el objeto tanto sus características como sus funciones.

Después de que los estudiantes pudieron visualizar los objetos en su entorno, se les explicó la necesidad de presentar la funcionalidad de objetos en la computadora por medio de estructuras de control (condicionales y ciclos). Se introdujeron estos conceptos en programas simples y cortos que permitieron a los estudiantes obtener la confianza de dar órdenes a la computadora para que ésta las ejecute, la cual es necesaria para que el estudiante pueda con libertad experimentar con la computadora.

El concepto de clase se introdujo cuando los estudiantes ya manejaban las estructuras de control, pero todavía no desarrollaron las habilidades de programación estructurada. Los estudiantes para este momento ya tenían interés de conocer en qué consisten las clases, porque se les dio un "jueguito" simple que usaba dos definiciones de clases: bicho que era un carácter ASCII "extraño", con su posición en la pantalla y color, con los métodos, entre otros, de moverse, desaparecer, y otra clase de jugador con sus respectivas características y métodos bastante simples. El juego tenía dos instancias de la clase bicho y dos de la clase jugador. Este juego se les presentó para motivar a los estudiantes a estudiar la materia presentada durante lecciones. Sin embargo, para explicar los conceptos nuevos como secuencia necesaria para llegar a este producto final, se desarrolló toda serie de programas en cada uno de los cuales se aumentaba el nivel de dificultad y se incorporaba nuevas partes. En esta forma aunque el programa final era de varias páginas de código, el crecimiento de este código fue paulatino y fácil de abarcar por los estudiantes.

Además, se buscó como primera tarea programada la posibilidad de que el estudiante usara estas clases con algunas modificaciones . Como, por ejemplo, se implementó el juego de escribir a máquina, donde el bicho se convirtió en una letra que se movía en la pantalla, un juego de comelón que engorda conforme se come los bocadillos que aparecen en la pantalla; otros profesores lo usaron en un juego de ping-pong donde el bicho era la bola de ping-pong, un juego de gato y ratón, etc. Esta primera tarea programada era crítica porque les daba confianza a los estudiantes que pueden programar usando objetos.

En las siguientes tareas programadas el estudiante desarrollaba sus propias clases guiado por el profesor y con las discusiones dirigidas sobre el diseño realizadas en clase entre diferentes grupos del proyecto.

Los estudiantes al principio tenían dificultades para ver que tan complejos deben ser métodos. Este aspecto, aunque ahora parece obvio, no se enfatizaba desde el principio. Sin embargo, como el profesor revisaba las tareas programadas, de inmediato se notó que este aspecto es de suma importancia. Se decidió trabajar sobre el diseño de las clases en nivel personal a cada uno de los estudiantes y al final del curso ya tenían la visión clara de la limitación del código que deben tener los métodos. La decisión de trabajo en forma individual se tomó cuando se notó las diferencias significativas entre estudiantes con previo conocimiento de programación estructurada y sin este conocimiento.

Los estudiantes con los conocimientos previos de la programación estructurada tenían desarrollada la modalidad de pensamiento funcional y parámetros, y aunque diseñaban clases con sus adecuados atributos, nunca usaban los atributos y todos los mensajes a objetos se manejaban por medio de los parámetros que en su totalidad reemplazaban los atributos de clase. Estos estudiantes de la misma forma tenían más dificultades de limitar el código propio de los métodos. En general, sus métodos eran pedazos de programas que hacían de todo.

Al contrario, los estudiantes sin ningún conocimiento previo de programación aprendieron más rápido a diseñar y usar las clases. Además, como se usaba el lenguaje C++ se consideró incorporar los aspectos de programación estructurada, siendo éstos todavía en uso general. Estos conceptos fueron introducidos en forma natural conforme crecía el código de programa principal. Los estudiantes solos preguntaban sobre la posibilidad de crear funciones para no repetir la escritura del código. Estos conceptos se introdujeron después de que el estudiante ya pudo manejar los objetos (tanto su diseño como la implementación). Así, en el mismo curso el estudiante pudo asimilar los dos conceptos paralelamente: POO y programación estructurada. Sin embargo, el proceso de asimilación fue diferente para los estudiantes con conocimientos previos en programación estructurada y sin este conocimiento. El autor de este artículo, basado en la experiencia de impartir el curso de POO durante varios semestres, considera que tienen menos dificultades para aprender POO los estudiantes que primero aprenden la POO y después la programación estructurada que en el caso inverso.

Aunque es evidente la necesidad de introducir el paradigma de orientación de objetos a los cursos de programación, el profesor al impartir este curso a los novatos puede encontrar varias dificultades. Una de las formas de superarlas es compartir las experiencias de impartir este curso por diferentes profesores. En general, es más fácil estar de acuerdo o en desacuerdo con algunas ideas y mejorar algunas o buscar otras cuando se tiene alguna base de pensamiento que comenzar todo en proceso de nuevo.

5. DIFICULTADES PRESENTES EN EL DESARROLLO DEL CURSO INTRODUCTORIO USANDO EL PARADIGMA DE ORIENTACIÓN A OBJETOS

La experiencia de impartir el curso de POO a novatos lleva al autor de este artículo a mencionar algunos retos que cada profesor debe considerar antes de tomar la decisión de impartir este curso. Se presentará la opinión personal de cómo algunos de estos retos se solucionaron y dieron resultado positivo y como para algunos de ellos se necesita hacer más investigación:

- Existencia de diferentes opiniones de cuál de los paradigmas se debe usar para iniciar: POO o programación estructurada. El mismo curso fue dado por diferentes profesores algunos de los cuales primero usaban el enfoque de programación estructurada y posteriormente de POO. La opinión del autor de este artículo, basada en la experiencia, y análisis de aprendizaje individual de cada uno de los estudiantes es que los estudiantes tienen menos dificultades de pasar de POO a programación estructurada que hacerlo al revés. Sin embargo, la dificultad de este enfoque es que las clases tienen que ser presentadas en forma simple, obligando al profesor a preparar sus propias bibliotecas adecuadas al nivel del conocimiento de los estudiantes. Esto desgraciadamente impide rápida introducción de bibliotecas estándar de clases, especialmente del compilador de Borland, por ser bastante complejos.

- Falta la metodología establecida que hace que el profesor tiene que guiarse por su intuición y aplicar el método de "prueba-error". Como se mencionó anteriormente,

cuando desde primeras lecciones se usó clases y presentó las estructuras de control como la parte de los métodos, los estudiantes desde principio enfrentaron los problemas durante la compilación que no fueron fáciles para su manejo. Las siguientes veces, aunque siempre se discutía sobre clases, los programas eran de programación estructurada, dando así al estudiante la posibilidad de disfrutar su acercamiento a la programación.

- Existencia de una variedad de ambientes de POO que de un lado puede ser ayuda, de otro dar confusión. Se necesita tomar la decisión de estar en ambiente de programación por eventos de Windows o volver al ambiente de DOS. Apoyando la idea que los estudiantes tienen que ser más que usuarios de los sistemas y deben aprender a ser creadores de *software*, se optó hacer las primeras tareas programadas usando el ambiente DOS, donde con más claridad se pudo manejar el ambiente de gráficos, las teclas direccionales con flechas, ratón, etc. Después los estudiantes más avanzados por su propia cuenta buscaban las facilidades que ofrece el ambiente Windows para creación de ventanas, uso de los botones, etc. El autor de este artículo considera que es muy importante enseñar a los estudiantes novatos las bases, donde pueden entender la complejidad de programación haciendo funcionar los elementos sin usar palabras "mágicas" que hacen todo el trabajo por ellos. Sin embargo, se considera importante que los conceptos presentes en la programación en ambiente Windows deben ser vistos en algunos cursos más avanzados.
- Uso de ayuda en línea. La ayuda en línea de Borland C++ de las versiones 4.5 y 5.0, que se usaba en el curso de POO, es poco clara y difícil de usar para los estudiantes novatos. También, los ejemplos se presentan usando el lenguaje C, provocando confusión al principio. Desde el principio se inculcó al estudiante el uso de la ayuda para encontrar diferentes funciones, contestarse a las preguntas, qué pasa si cambio esta operación por otra, buscar ejemplos hechos para ayudar al estudiante a independizarse del profesor y aprender a crecer como programador de acuerdo a su ritmo individual. Sin embargo, este propósito no siempre se pudo cumplir debido a las dificultades mencionadas anteriormente presentadas en Borland C++. Este compilador fue el único que se pudo usar debido a las limitaciones de *hardware*.

- Falta de libros de texto, donde el estudiante desde principio puede ver los ejemplos y uso de clases simples. Las primeras lecciones donde se introducía los conceptos de clases eran críticas. Sin embargo, si un estudiante no pudo asimilar los conceptos, los libros no le podrían ayudar porque en general se presentan clases como elementos avanzados, donde el estudiante ya domina los conceptos de estructuras, apuntadores, arreglos, etc. Para los estudiantes que dominaban apenas las estructuras de control (condicionales y de ciclo) y tenían alguna moción de funciones y parámetros, era muy difícil y confuso respaldar las explicaciones hechas durante lecciones usando libros existentes.

- Necesidad de contacto más directo con los estudiantes. Para que el estudiante pueda aprender la POO, se necesita el trabajo más cercano y directo con él. El profesor debe contar con continua retroalimentación con respecto al desarrollo de los estudiantes en el curso. Esta retroalimentación solamente se puede obtener revisando los trabajos de los estudiantes y adaptando el curso a sus necesidades. Esta situación es diferente de la situación existente en la época de solo lenguajes estructurados debido a que ahora se debe enfocar los diferentes aspectos para los estudiantes con previa

preparación en programación estructurada o sin ella.

- Limitado apoyo de los asistentes y más carga real de trabajo para los profesores. Hasta este momento se contaba con el asistente que revisaba, por ejemplo, las tareas programadas. Estas tareas son lo más esencial del curso. Sin embargo, dejar al asistente la carga de formar la mentalidad de orientación a objetos no es la opción ideal, si todavía no es claro para el profesor cual sería la mejor manera de hacerlo. El estudiante aprende programando, así el profesor para poder hacer su trabajo de la mejor forma debería revisar tareas programadas, reunirse con los grupos, revisar exámenes, etc. Por estas razones, los asistentes pueden hacer un trabajo de apoyo al profesor, pero no necesariamente el trabajo que hacían antes de revisión de tareas programadas, exámenes cortos, tareas cortas. Esta demanda de esfuerzo adicional no siempre es reconocida. Esto puede crear ambiente de no tener muchas posibilidades de encontrar los profesores que cumplan con las características necesarias y que estén de acuerdo a realizar el trabajo extra.

- Falta de algún grupo de discusión y enriquecimiento por medio de intercambio de opiniones. En general, las personas no quieren compartir sus debilidades y cuando uno imparte el curso de POO a objetos es difícil sentirse satisfecho. Los estudiantes solamente después de pasar otro curso más de programación pueden ver si obtuvieron las bases adecuadas. Así la evaluación de trabajo es una evaluación subjetiva del profesor. La posibilidad de compartir sus experiencias, ver el proceso que pasan los profesores para adecuar las metodologías de enseñanza, hacer intercambio de las ideas implementadas en el curso que dieron buen o mal resultado daría más seguridad en mejorar la forma de enseñar el curso de POO. Desgraciadamente, en general no se establece este tipo de apoyo e intercambio de opiniones es mínimo.

- Falta de retroalimentación efectiva de los profesores de cursos posteriores para ver los aspectos que se pueden mejorar. En general no se da el seguimiento a los estudiantes para ver si las diferentes metodologías aplicadas en el primer curso de programación influye al desarrollo de las habilidades de programación. Esto impide tener los datos suficientes para tomar la decisión en qué dirección realizar los cambios metodológicos. De esta forma los profesores cuentan únicamente con su intuición y aplican el método de "prueba-error" al costo de estudiantes.

- Uso de biblioteca de clases simples hechas para los propósitos de explicación de los conceptos. La biblioteca estándar de clases no es fácil de usar desde primeras lecciones por su compleja estructura de herencia, parámetros tipo apuntador, etc. El estudiante tiene que tener conocimientos no necesariamente básicos para sentirse cómodo de usarlos (es una opinión particular del autor de este artículo con respecto a bibliotecas de clases presentes en Borland C++). Para introducir los conceptos de contenedores, el autor de este artículo desarrolló plantillas de clases en forma simple. Esto demandó otra porción de trabajo extra.

6. CONCLUSIONES

La programación orientada a objetos nació como el proceso evolutivo de lenguajes de programación. Comenzando por lenguajes estructurados, seguido por los lenguajes que usan módulos, se logró incorporar la programación orientada a objetos como la necesidad de respuesta a las crecientes exigencias de sector industrial, donde se notaba el aumento considerable de tamaño de los programas difíciles de escribir y mantener sin el uso de objetos.

Como respuesta a este cambio de paradigma de programación, se necesita preparar los profesionales en computación con las habilidades de desarrollar el *software* usando las tecnologías de orientación a objetos. Esto despertó la polémica sobre si es o no recomendable dar primeros cursos de programación usando el enfoque de orientación a objetos.

Aunque se desarrollaron diferentes metodologías para impartir este curso, en la sociedad científica no existe la opinión unificada sobre el aspecto de introducir la POO a los novatos. Además, la inclusión de un curso de este tipo exige contar con los "pioneros" dispuestos a pagar el precio de introducir un cambio sustancial en lo que se refiere la enseñanza de la programación.

La experiencia personal del autor de este artículo fue diferente en cada uno de los semestres en los cuales se impartió este curso. La primera vez se introdujeron muy temprano los conceptos relacionados con las clases y éstos fueron recibidos por los estudiantes como conceptos demasiado abstractos. El cambio de metodología de enseñanza reflejó la posibilidad de aprender la POO como primer lenguaje de programación sin ninguna dificultad diferente de las presentes en la enseñanza de programación estructurada. Más aún, los estudiantes sin ningún conocimiento previo en programación tenían menos dificultades en aprender POO y también fueron capaces de aprender el paradigma de programación estructurada.

Sin embargo, debido que el paradigma de POO es nuevo, el profesor que imparte este curso para novatos encuentra muchas dificultades que no siempre son fáciles de sobrepasar.

Es importante que los profesores en Costa Rica que imparten este tipo de curso comparten sus experiencias. Esto nos permite encontrar y fortalecer metodologías más aptas a este tipo de enseñanza.

Aunque la enseñanza de POO a novatos no es un proceso fácil, la situación actual con respecto al desarrollo de *software* tanto en el nivel académico como en el nivel industrial señala que es un aspecto muy importante y no puede ser ignorado. Gracias a los profesores aptos y dispuestos de seguir en este camino de cambio y buscar el mejoramiento en la enseñanza de POO, los estudiantes salen beneficiados.

7. BIBLIOGRAFIA

[BOOC96] Booch G. *Análisis y Diseño Orientado a Objetos con Aplicacciones.* Addison-Wesley, 1996.

[DEIT94] Deitel H.M y Deitel P.J. *Como Programar en C/C++.* Prentice-Hall, 1994.

[DEIT99] Deitel H.M y Deitel P.J. *Como Programar en C++.* Prentice-Hall, 1999.

[GIBB96] Gibbon C, Lovegrove G. y Higgins C. *Tools, Heuristic and Techniques to Assist OO Education.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1996.

[HAID98] Haiduk P. *Turbo Pascal Orientado a Objetos,* McGraw-Hill, 1998.

[HELI97] Heliotis J. *Experiences Teaching Objects. A New Curriculum for First Year Computer Science Students.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1997.

[JOYA96] Joyanes L. *Programación Orientada a Objetos: Conceptos, Modelado, Diseño y Codificación en C++*. McGraw-Hill, 1996.

[KELL96] Kelly P. Smith D. *The Impact of Object-Oriented Technology on the Computer Science Curriculum at the University of Ballarat, Australia.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1996.

[LIND96] Lindskov J. y Lehrmann O. *Using Object-Orientation as a Common Basis for System Development Education.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1996.

[MANN96] Manns M.L y McKim J. *Teaching OT: A Bread-first and Depth-first Approach.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1996.

[MCKI93] McKim J.C. *Teaching Object-Oriented Programming and Design.* JOOP, marzo-abril, 1993.

[MALL96] Malloy B., McGregor J. y Gupta D. *An Apporach to Blending Nalysis, Design and Implementation.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1996.

[PATT97] Pattis R. *OOP and CS1: Oil and Water or Oil and Winager.* Proceedings of Conference on Object-Oriented Programming Systems, Languaes & Applications, octubre 1997.

[REIS99] Reisman A. *Aprendiendo C++ en 21 Dia.* Prentice-Hall, 1999.

[RUMB96] Rumbaugh James, Blaha M., Premerlani W, Eddy F. y Lorensen W. *Modelado y Diseño Orientados a Objetos. Metodología de OMT.* Prentice Hall, 1996.

[SOMM92] Sommerville I. *Software Engineering.* Addison-Wesley, 1992.

[STAU98] Staugaard A. *Técnicas Estructuradas y Orientadas a Objetos.* Prentice Hall, 1998.

[STRO99] Stroustrup B. *Learning Standard C++ as a New Language.* Mayo, 1999.

[WU93] Wu T. *Teaching OOP to Beginners.* JOOP, marzo-abril 1993.

Elzbieta Malinowski G.,
emalinow@cariari.ucr.ac.cr