Enseñanza Inicial de la Programación: Comparación entre Programación Estructurada y Programación Orientada a Objetos

Ing. Inés Kereki[1] **Mc. Fernando Salvetto**[2]
e-mail: kereki_i@athenea.ort.edu.uy e-mail: salvetto@athenea.ort.edu.uy

Resumen

La enseñanza inicial de la programación, tradicionalmente bajo el paradigma estructurado, parece estar encaminándose hacia la orientación a objetos.

En la Universidad ORT Uruguay, en 1996, comenzamos la enseñanza de la programación bajo el paradigma de objetos en la carrera de Ingeniería en Sistemas.

En este trabajo se presenta una comparación entre dos opciones: la enseñanza inicial de la programación de acuerdo al paradigma estructurado y de acuerdo al de orientación a objetos. Se explican las características de dichos cursos en el contexto de su respectiva carrera y se comparan ambas situaciones didácticas, a partir del análisis de la producción de los alumnos y entrevistas a alumnos y docentes.
Finalmente, se ofrecen conclusiones.

Palabras clave:
Programación, enseñanza inicial de la programación, orientación a objetos, programación estructurada, Smalltalk, Pascal, estudiantes universitarios.

1. Introducción

En muchas universidades, según podemos constatar leyendo los programas académicos de carreras de Analista de Sistemas, Ingeniería de Sistemas y similares[3], se comienza enseñando la programación con un enfoque estructurado (PE) y en cursos posteriores se les presenta a los estudiantes la orientación a objetos (POO).

Hasta 1995, en la Universidad ORT Uruguay (en ese momento, Instituto Tecnológico ORT), se enseñaba a los alumnos de la carrera Analista de Sistemas en los cuatro primeros semestres programación estructurada y en el quinto semestre se comenzaba con programación orientada a objetos.
Allí los docentes observaban ciertas características, como por ejemplo: alguna dificultad de abstracción o actitud receptiva y pasiva de los alumnos, que fueron objeto de prolongado análisis y discusión entre los docentes de ORT y de otras universidades[4].

En 1996 se creó la carrera de Ingeniería en Sistemas y se planteó, desde el comienzo, una perspectiva diferente. Se cambió el paradigma de programación[5] y los métodos tradicionales de

---

[1]Docente y Coordinadora de Programación I y II, Ingeniería en Sistemas, Universidad ORT Uruguay.
[2]Docente de Programación I y II, Análisis de Algoritmos I y II, Ingeniería en Sistemas, Universidad ORT Uruguay.
Universidad ORT Uruguay. Cuareim 1451. Tel. 9021505 Fax: 9002952. 11100 Montevideo, Uruguay
Internet: http://www.ort.edu.uy
[3]Los programas se obtuvieron a través de Internet, por e-mail y por consulta directa a docentes de varias universidades.
[4]En particular, con docentes de la Universidad Nacional de La Plata (Argentina) y Universidad del Centro de la Provincia de Buenos Aires (Argentina).

enseñanza. Se comenzó enseñando programación orientada a objetos (utilizando Smalltalk V) durante un año y luego programación estructurada.

2. Descripción de las carreras de Analista de Sistemas (plan 1989 y 1992) e Ingeniería en Sistemas (plan 1996) y cursos de Programación I y II.

2.1. Carrera de Analista de Sistemas (planes 1989 y 1992)

2.1.1. Características generales

La carrera tenía como *objetivo* formar profesionales capaces de analizar los requerimientos de información de empresas, diseñar sistemas apropiados para dichas organizaciones y poder implementarlos con éxito con la participación de personal (programadores, operadores, etc.) a su cargo. Tenía particular importancia formar al profesional de modo que pudiera trabajar en equipos multidisciplinarios con otros profesionales.

El *requisito de ingreso* era tener 6to. año del Bachillerato Diversificado (de cualquier orientación) completo y aprobado.
La *duración* de la carrera era 3 años.

En el currículo se incluían materias tales como Programación, Bases de datos, Análisis de algoritmos, Diseño y análisis de sistemas, Redes, Sistemas operativos.

2.1.2. Programación I y II en Analista de Sistemas (planes 1989 y 1992)

En particular, la materia Programación I tenía como *objetivo*: iniciar la enseñanza de la programación, empleando fundamentalmente técnicas de programación estructurada. Se daba énfasis a la enseñanza de una metodología de resolución de problemas, a través del uso de diagramas de flujo y seudocódigo. Se daba también particular importancia al diseño estructural del programa, que se implementaba en lenguaje Pascal.

Programación II tenía como *objetivo* continuar la línea de la materia previa Programación I presentando elementos más avanzados en cuanto a construcción de programas, modularidad, diseño, etc. incluyendo el manejo de organizaciones sencillas de archivos y pautas sobre métodos para la creación de archivos.

Los *contenidos* eran:

> **Programación I**: *Conceptos básicos*: máquina; algoritmo; programa; ciclo de programación; noción de gramática y sintaxis; Pascal.
> *Algoritmos*: representación de algoritmos; estructura de programas; flujo de control; modularidad; programación Top-down; prueba de programas.
> *Elementos*: forma de un programa; variables; tipos de datos básicos; operadores; estructuras de control; entrada y salida; manejo de pantalla; procedimientos y funciones; strings; vectores; matrices; registros; tablas; subrangos; conjuntos; enumerados.

---

[5] En Kereki (1997) se presentan justificaciones teóricas, prácticas e institucionales del cambio de paradigma, así como también las características y resultados de los cursos.

Programación II: *Tipo de datos*: conceptos; Pascal: conjuntos, escalares, enumeración; tipo abstracto de datos.
Recursividad: definición; ejercicios básicos; técnicas de diseño de algoritmos: dividir para conquistar, backtracking; aplicaciones.
Módulos: programación del tipo de datos; encapsulamiento; unidades; pautas sobre modularidad y diseño.
Archivos: de texto, secuenciales y relativos: conceptos básicos; procesos comunes.

El lenguaje utilizado era Turbo Pascal 5.5.

La *evaluación* se realizaba originalmente con 2 pruebas parciales realizadas en clase y 2 trabajos obligatorios en grupos de 2 alumnos, de un mes de duración cada uno.
Para que los alumnos sintieran, entre otras, la necesidad de tener una buena documentación y la importancia de un testing correcto y completo se agregó la 'corrida cruzada', en la cual cada grupo de dos estudiantes recibe el trabajo de otro grupo y debe realizar un informe detallado de ese obligatorio, incluyendo una crítica general y particular del testing realizado, de la documentación, de la calidad del código, etc.
Al final del curso, deben rendir un examen individual.

2.2. Carrera de Ingeniería en Sistemas (plan 1996)

2.2.1. Características generales

La carrera tiene como *objetivos*, entre otros, formar profesionales capaces de diseñar y desarrollar sistemas de alta escala y complejidad, desempeñarse con éxito como desarrolladores de software, gerentes y consultores, adaptarse al constante cambio e integrarse a equipos multidisciplinarios.

El *requisito de ingreso* es tener 6to. año del Bachillerato Diversificado (opción Ingeniería) completo y aprobado.
La *duración* de la carrera es 5 años.

Algunas materias del currículo son: Programación, Estructuras de datos y algoritmos, Ingeniería de Software, Organización y Arquitectura de sistemas, Sistemas Operativos y Redes, Bases de datos, Matemáticas y Física.

2.2.2. Programación I y II en Ingeniería en Sistemas

El *objetivo* de Programación I es iniciar la enseñanza de la programación empleando técnicas de programación orientada a objetos. Se da énfasis a la enseñanza de una metodología de resolución de problemas. Capacita al estudiante para desarrollar aplicaciones con lenguajes puros orientados a objetos (Smalltalk V).

A su vez, el objetivo de Programación II es continuar la capacitación del estudiante en el paradigma de objetos presentando elementos más avanzados en cuanto a construcción de programas, re-uso, polimorfismo, modularidad y diseño

Los *contenidos* de las materias de programación son:

Programación I: *Conceptos básicos de programación orientada a objetos*: objetos, mensajes, clases, métodos, composición, herencia y polimorfismo.
Introducción al lenguaje Smalltalk: ambiente, sintaxis, objetos simples, tipos de mensajes, tipos de variables, definición de clases y métodos, herencia, polimorfismo, jerarquía de clases. Resolución de problemas simples.

Programación II: *Resolución de problemas con orientación a objetos*: características y ventajas de la tecnología orientada a objetos. Abstracción y encapsulación. Concepto de dominio del problema. Conceptos de interfaz de usuario. Conceptos de diseño orientado a objetos. Composición vs. herencia. Delegación.

El lenguaje utilizado es Smalltalk V (Digitalk).

La *evaluación* de los alumnos se realiza en forma similar a la explicada en Programación I y II de Analista de Sistemas.

3. Comparación entre la enseñanza inicial de programación estructurada y de programación orientada a objetos.

Tomando como referencia el análisis de los trabajos de los alumnos (obligatorios, parciales y exámenes) y las 50 entrevistas que se realizaron a estudiantes y docentes, así como las encuestas que se realizaron a todos los alumnos, es posible comparar las materias Programación I y II de las dos opciones: (paradigma estructurado: PE; paradigma orientado a objetos: POO) según:

- **metodología y dinámica de la clase**
- **bibliografía**
- **actitud de los alumnos**
- **capacidad de abstracción**
- **comprensión del dominio del problema**
- **técnicas de diseño y programación**
- **algoritmos y manejo de detalles de implementación**
- **resultados de cursos y exámenes**

- **Metodología y dinámica de la clase**

PE	Se enseñaba, según indicaron los docentes, a través de ejemplos planteados en el pizarrón para mostrar la necesidad de determinado elemento y luego presentarlo. Así, por ejemplo, para explicar vectores, se planteaba que se quería ingresar los datos de resultados de un curso y que se debía mostrar el promedio y, para cada nota, la diferencia que tuvo respecto a ese promedio. Se dictaban algunas clases en laboratorio, básicamente para explicar el manejo del ambiente de trabajo y el uso del debugger. La mayoría de las clases (especialmente al comienzo del curso) se dictaban en forma magistral.
POO	Además de los ejemplos tradicionales, en los que se crea la necesidad de un elemento, se agregó en la clase el uso del TV Elite: un PC con un monitor adicional grande y software que permite resaltar y destacar áreas de la pantalla. Los alumnos ven en el momento cómo se implementa una solución a un ejercicio y además ven la programación 'real', con aciertos y errores. Se programa y se investigan las clases del Smalltalk con ellos. Los alumnos participan activamente en la identificación y discusión de las causas de un determinado comportamiento o problema y aportan soluciones. La dinámica de la clase cambió. El ambiente integrado e interactivo de desarrollo del Smalltalk, presente en la clase con el TV Elite, incentiva la participación del alumno. Los alumnos manifestaron en las entrevistas que su uso les brindó seguridad ('no lo dice el docente solamente, yo lo vi') y agilidad a la clase ('no tengo que copiar todo, me llevo en un diskette el trabajo'). Otro elemento que apoyó este cambio fue la creación del WEB académico donde los estudiantes cuentan con ejercicios, soluciones, letras de obligatorios, parciales y exámenes, listas de discusión, lista de direcciones en Internet interesantes para el curso, etc.

- **Bibliografía**

PE	En Analista de Sistemas se recomendaban varios autores[6] y según los temas, se señalaba dónde sería más claro o completo leerlo. En las entrevistas realizadas a los estudiantes, constatamos que muy pocos recurrían a los manuales o libros en general. Estudiaban básicamente de los apuntes de clase.
POO	El curso se desarrolló siguiendo el libro Discovering Smalltalk, LaLonde (1994). Es escasa la bibliografía de programación orientada a objetos adaptada a estudiantes que recién se inician. La mayoría del material para aprender programación orientada a objetos está dirigido a quienes conocen o dominan el paradigma estructurado. En las entrevistas constatamos que los alumnos leían, investigaban y buscaban más información por todos los medios a su alcance (libros, revistas, Internet, WEB académico, seminarios).

- **Actitud de los alumnos**

PE	Según las entrevistas con los docentes, se notaba una actitud receptiva y pasiva en los alumnos. El docente funcionaba como transmisor de conocimiento, liderando el proceso de enseñanza-aprendizaje. Los alumnos no generaban nuevas situaciones de aprendizaje. Si tenían un problema, por ejemplo con el trabajo obligatorio, lo planteaban individualmente al docente pero no lo discutían o compartían con los demás compañeros. Parecían tener dificultades para integrarse y trabajar en equipo.
POO	Los docentes indicaron que sienten que los alumnos colaboran entre ellos y con el docente y comprenden que el aprendizaje depende de sí mismos (Kereki (1997b)). El docente funciona como facilitador del aprendizaje. El disponer del equipo en clase permite que, ante un problema específico, los alumnos traigan el diskette con el mismo, se instale y se discuta entre todos. Esto fomenta la colaboración entre los estudiantes, pues comparten las dificultades y soluciones. También facilita la integración con el docente, pues participa como uno más en la discusión.

- **Capacidad de abstracción**

PE	Durante varios años se dictó el curso de Programación I y II en Analista de Sistemas. Uno de los mayores problemas que percibieron los docentes (y que constatamos revisando trabajos de los alumnos) fue la dificultad para abstraer: los alumnos leían la letra de los ejercicios y no abstraían los conceptos o elementos fundamentales. Se preocupaban del 'cómo' antes del 'qué'. En semestres avanzados de la carrera se veía este mismo problema. Por ejemplo, ante un ejercicio de un parcial, los estudiantes (en su mayoría) comenzaban a programar, sin tener claro aún el problema en general. La atención del estudiante estaba absorbida por los detalles. Veían con nitidez el árbol y parecían no interesarse en ver o saber que existía el bosque.
POO	Dejó de ser un problema el análisis, diseño e implementación de problemas de complejidad media y baja, según se puede percibir de las entrevistas, ya que una de las preguntas era un ejercicio que se les pedía que resolvieran. Los alumnos piensan, discuten y analizan varias opciones antes de ponerse a programar efectivamente. En una investigación sobre el aprendizaje de la programación orientada a objetos realizada entre setiembre y noviembre de 1996 con alumnos de 2do. semestre de Ingeniería, se constató que solamente el 20% de los alumnos comenzaba a programar antes de tener todo analizado o, al menos, pensado (Kereki, 1997b op. cit.). La preocupación de "¿cómo hago esto?" no es su problema inicial, tratan de entender y ver todos los aspectos antes de empezar efectivamente a programar.

[6]Ver Apéndice A: Bibliografía recomendada para el estudio inicial de programación estructurada y Apéndice B: Bibliografía recomendada para el estudio inicial de programación orientada a objetos.

- **Comprensión del dominio del problema**

PE	Parecían estar centrados en el detalle. Daban más importancia a la programación que al análisis, según constatamos en las entrevistas. No referían el problema a un contexto, mostrándose sorprendidos frente a cambios menores (y anticipables) de los requerimientos.
POO	Demostraron en los trabajos mayor capacidad de investigación y ajuste a las necesidades del usuario, pudiendo resolver en forma eficiente modificaciones al dominio del problema. Por ejemplo, en el primer trabajo obligatorio del segundo semestre, se planteó una situación en la cual los docentes representaron el papel de usuario y con los alumnos se fueron definiendo los requerimientos. Ellos debían entregar, junto con el resto de la documentación y el sistema, sugerencias de modificación a los requerimientos. El segundo trabajo obligatorio de ese semestre fue la implementación de algunas de estas sugerencias. Los trabajos de la mayoría de los alumnos pudieron soportar estas modificaciones a costos razonables.

- **Técnicas de diseño y programación**

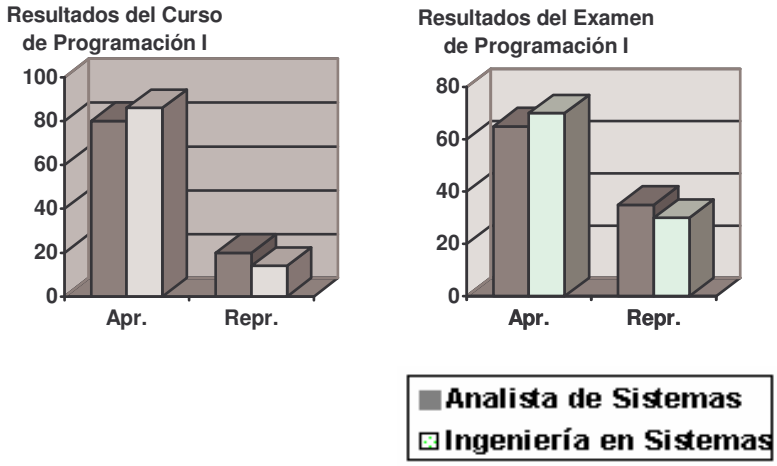
PE	Como indicaron los docentes, era difícil que apreciaran técnicas de diseño y programación cuando se les planteaba exclusivamente sistemas pequeños ya que el paradigma y lenguaje utilizados no favorecen ni simplifican el re-uso. Era muy difícil transmitir y, sobre todo, lograr que aplicaran, conceptos tales como: encapsulamiento, modularidad, cohesión, acoplamiento, importancia del testing, programación defensiva, necesidad de una buena documentación, etc., ya que representaban soluciones a problemas que no tenían los alumnos. A modo de ejemplo, el hecho de que un programa no mantenible no tiene mucho valor no era tenido en cuenta. No parecía haber una conciencia de la existencia e importancia del usuario.
POO	Los docentes indicaron que sienten que los alumnos están trabajando con método y tienen clara la importancia del usuario. No programan sólo para ellos, se sienten parte de un equipo. También esto se detecta en las entrevistas realizadas a los alumnos. Se interesan y muestran interés en aprender y usar técnicas de análisis y diseño. Los trabajos obligatorios alcanzaron mayor nivel de calidad.

- **Algoritmos y manejo de detalles de implementación**

PE	En general, al analizar los trabajos, se constata que hacían buenos algoritmos para situaciones específicas. El manejo de parámetros era un tema de gran dificultad. También el uso de programas recursivos planteaba problemas. (Esta situación se puede verificar revisando exámenes de Programación II. En dichos exámenes, como uno de los ejercicios, se pedía implementar en forma estrictamente recursiva un proceso simple, por ejemplo: *'Cargar un vector con la cantidad de veces que se utilizó cada letra en un string dado'*. Gran cantidad de los alumnos resolvía en forma incorrecta el final de las llamadas recursivas o no utilizaba correctamente funciones o procedimientos auxiliares.)
POO	Se nota al analizar los parciales y obligatorios realizados que tienen facilidad para plantear algoritmos pero dificultad para implementarlos al nivel de detalle. Por ejemplo, en el segundo trabajo obligatorio de Programación II, debieron implementar un algoritmo que seleccionara una combinación óptima de latas de pintura de distintos tamaños para satisfacer un pedido. Muy pocos alumnos realizaron algoritmos eficaces, eficientes y modificables.

- **Resultados de cursos y exámenes**

PE	El porcentaje de aprobación del curso de Programación I era de 80% y el del examen (en el primer período luego de finalizar el curso) era de 65%. En Programación II se obtenían similares valores. La deserción entre primer y segundo semestre era de 25% y entre primer y tercer semestre de 48%. (Es de destacar que para poder inscribirse en el tercer semestre deben tener aprobado por completo 6to. año del Bachillerato).
POO	En el curso de 1996, 85% de los alumnos aprobaron el curso de Programación I y de ellos, 70% aprobó el examen en el primer período. Estos valores son algo más altos que los obtenidos en Analista de Sistemas. En Programación II se obtuvieron valores similares. La deserción bajó considerablemente: 5% de deserción entre 1er. y 2do. semestre y 30% entre primer y tercer semestre.



4. Conclusiones

De acuerdo al análisis y evaluación de la producción de los alumnos (obligatorios, parciales, exámenes) y de las entrevistas y encuestas realizadas a alumnos y docentes, se puede constatar que los estudiantes que iniciaron el aprendizaje de la programación bajo el paradigma de orientación a objetos (con las características metodológicas indicadas) alcanzaron mayor nivel de abstracción, capacidad de análisis y resolución de problemas que los estudiantes que lo iniciaron bajo el paradigma estructurado.

Los estudiantes que comenzaron con el paradigma de objetos muestran dificultades para el manejo de los detalles de implementación. Habrá que esperar el desarrollo de los cursos de Análisis de Algoritmos I y II (en tercer y cuarto semestre) para enfrentar y solucionar esta carencia.

Es importante destacar que estos estudiantes lograron tempranamente una comprensión del dominio del problema (análisis), conciencia de la existencia del usuario y de la importancia de construir el sistema que éste necesita (ingeniería de requerimientos), características ausentes en los primeros semestres de la orientación estructurada.

Ya que únicamente se dictó el primer año de Ingeniería en Sistemas (y está finalizando el 2do año de dicha carrera), no se puede generalizar las conclusiones; nuestra intención es continuar con el estudio y seguimiento de estos estudiantes.

Referencias

Kereki, I. (1997). *Enseñando y aprendiendo programación orientada a objetos en los primeros cursos de programación: la experiencia en la Universidad ORT Uruguay.* III Jornadas Iberoamericanas de Informática, 7-11/4/1997, La Antigua, Guatemala.

Kereki, I. (1997b). *¿Qué es programar con orientación a objetos? Un enfoque fenomenográfico.* Monografía para la obtención del Diploma en Educación. Uruguay: Universidad ORT Uruguay.

- Universidad Autónoma de Madrid, España. http://www.ii.uam.es/esp/curso1.html.
- Universidad de Carleton, Canadá. http://www.scs.carleton.ca/scs/grad2/html
- Universidad de Chile, Chile. http://www.dcc.uchile.d/dcc/icc/icc.
- Universidad Nacional de Buenos Aires, Argentina. http://www.dc.uba.ar/materias.html
- Universidad ORT Uruguay, Uruguay. http://www.ort.edu.uy
- Universidad Tecnológica de Panamá, Panamá. http://www.utp.ac.pa/facultad/fisc/mat-ing.hmtl

Apéndice A

Bibliografía recomendada para el estudio inicial de la programación estructurada

Borland Internacional Inc. (1992) *Turbo Pascal for Windows.* USA: Borland International Inc.

Grogono, P. (1986) *Programación en Pascal.* USA: Addison Wesley Publishing Company.

Joyanes, L. (1990) *Programación en Turbo Pascal. Versiones 4.0, 5.0 y 5.5.* México: Mc. Graw Hill.

Kereki, F. (1987) *Recursividad.* Uruguay: Instituto Tecnológico ORT Uruguay.

Wirth, N. (1987) *Algoritmos + Estructuras de datos = Programas.* USA: Prentice Hall.

Wirth, N. (1987) *Algoritmos y Estructuras de datos.* USA: Prentice Hall.

Zaks, R. (1986) *Programación en Turbo Pascal.* México: Editorial Anaya Multimedia.

Apéndice B

Bibliografía recomendada para el estudio inicial de programación orientada a objetos

Booch, Grady (1991) *Object Oriented Design with Applications,* Redwood City, California. USA: The Benjamin/Cummings Publishing Company.

Jacobson, Ivar (1992), *Object-Oriented Software Engineering.* Reading Mass., USA: Addison-Wesley

LaLonde, W. (1994). *Discovering Smalltalk*. USA: The Benjamin/Cummings Publishing Company.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. (1991) *Object-Oriented Modeling and Design*. USA: Prentice Hall.

Winblad, A., Edwards, S. & King, D. (1993). *Software orientado a objetos*. USA: Addison-Wesley Iberoamericana S.A.