Una experiencia de enseñanza de Programación Funcional en carreras Informáticas e Ingenierías no Informáticas

Bernal, Rubén Alfredo[1] – Cuenca Pletsch, Liliana Raquel[1] – Fornari, Javier[2]

[1] Universidad Tecnológica Nacional
Facultad Regional Resistencia
Argentina
{rbernal, cplr}@frre.utn.edu.ar

[2] Universidad Tecnológica Nacional
Facultad Regional Rafaela
Argentina
javier.fornari@frra.utn.edu.ar

Resumen

En este trabajo se analiza el problema de los bajos rendimientos de los alumnos en los primeros años de carreras de Ingeniería y las dificultades adicionales que implica la enseñanza de la programación en estas carreras. Se analizan en detalle dos propuestas para mejorar el aprendizaje de conceptos básicos de programación y la capacidad para resolver problemas mediante la utilización de lenguajes funcionales. Se presentan, también, dos experiencias de utilización de un lenguaje funcional, Haskell, en la enseñanza de programación en una carrera Informática y en dos no Informáticas.

Palabras clave: desgranamiento, enseñanza de programación, resolución de problemas, paradigma funcional.

1. Introducción

La problemática del desgranamiento temprano en los primeros años de la carrera universitaria es, en nuestro país, objeto de preocupación para todos los actores involucrados. Los trabajos consultados al respecto abordan la problemática del estudiante universitario desde perspectivas diferentes, pero permiten advertir que la permanencia y el abandono, el éxito y el fracaso académico, se definen en la confluencia de múltiples factores. Hay coincidencia entre los autores en cuanto a que estos hechos, especialmente el éxito y el fracaso académico, no pueden explicarse basándose exclusivamente en déficits intelectuales o cognitivos, sino que deben considerarse otros condicionantes de índole motivacional y actitudinal.

Existen trabajos importantes a nivel internacional cuyo área de interés es la predicción del éxito de los alumnos en cursos iniciales de programación [6] [13] [14]. Los artículos referenciados en este sentido proponen la aplicación de pruebas de aptitud desarrolladas específicamente para este fin, las cuales se aplican, inclusive, a estudiantes que no acreditan conocimientos previos en programación. El estudio se realizó en instituciones de Gran Bretaña, Australia y Canadá. Los resultados expuestos corresponden al primer país. Los autores afirman que, en general, entre el 30 y el 60% de los alumnos de carreras en Ciencias de la Computación fallan en el primer curso de programación, lo cual no difiere de la realidad argentina. Este trabajo plantea que las personas tienen o no la capacidad para aprender a programar y que todo intento relacionado con mejoras en los procesos de enseñanza-aprendizaje o en los materiales de estudio son vacuos y están asociados al efecto Hawthorn[1].

Mc. Cracken y otros [7] analizaron las capacidades de los estudiantes de primer año de carreras de informática en el marco de un proyecto que involucró a investigadores de seis países. La principal conclusión a la que

[1] Las novedades introducidas por un nuevo sistema saca a los trabajadores de su letargo metodológico y permite obtener mejores rendimientos. Una vez que la novedad deja de ser tal, la productividad disminuye nuevamente.

arribaron fue que, al finalizar los cursos introductorios, un porcentaje importante de alumnos aún no sabe programar. Basados en el anterior, Lister y otros [12] llevaron adelante cabo un estudio en siete países cuyo objetivo fue determinar si las causas del fracaso se deben a la incapacidad para resolver problemas o, peor aún, a que los estudiantes no poseen los conocimientos y habilidades que son prerrequisitos para la resolución de problemas. Para indagar en esta cuestión evaluaron a los alumnos en dos aspectos: la capacidad para escribir una porción de código que permita resolver un determinado problema y la capacidad para leer un programa e interpretar su función. Los alumnos que fallan en el primer aspecto son quienes carecen de competencias para la resolución de problemas, en tanto que quienes fallan en el segundo no disponen de los conocimientos o habilidades que facilitan la adquisición de la mencionada competencia. Los autores coinciden con Wiedenbeck [1985] en que la estrategia para lograr mejores rendimientos en los alumnos del segundo grupo es asignarles práctica continua con material básico hasta que automaticen la práctica. La recomendación es que, en la formación inicial de programadores, el énfasis debe ponerse en la comprensión y seguimiento de programas.

En Argentina se están llevando adelante proyectos de investigación que se enfocan hacia esos aspectos. En lo que respecta a la enseñanza en carreras de Informática existen numerosos proyectos cuyo objetivo es lograr mejores rendimientos académicos en los alumnos de los primeros años, especialmente en asignaturas del área programación que es donde se verifican los peores rendimientos. Estos proyectos se enfocan hacia mejoras en el proceso de enseñanza-aprendizaje [1] [4] [18] [3] [10] [11], desarrollo de material de estudio de apoyo a la enseñanza presencial [2] [16] [3] y análisis de lenguajes más apropiados para el aprendizaje de la disciplina [4] [17], entre los más relevantes. También existen proyectos más recientes referidos a la articulación con el nivel medio, ya que se reconoce como uno de los obstáculos más importantes la formación previa de los alumnos que ingresan a la Universidad [5] [8]. Los resultados se consideran, en general, satisfactorios ya que permiten mejorar no sólo el rendimiento de los alumnos en términos de cantidad sino también en la calidad de la formación impartida.

La problemática de la enseñanza de la programación ha dado lugar a la realización de ateneos de profesores de Universidades argentinas y latinoamericanas que ofrecen carreras de Informática. También existen numerosos artículos de investigadores de Universidades de todo el mundo, preocupados por mejorar la formación de los alumnos de estas carreras.

En la Facultad Regional Resistencia de la U.T.N. se está llevando adelante una investigación para determinar las causas de los bajos rendimientos de los alumnos en los dos primeros años de carrera. Se propusieron como objetivos del mismo: describir el modo como operan en los estudiantes factores relevantes de índole motivacional y cognitivo en el contexto de un estilo institucional; reconstruir la interrelación entre estos factores y el contexto para evaluar su influencia en el desgranamiento temprano y elaborar una propuesta de intervención para contribuir al mejoramiento de los índices de retención de los primeros años. Los resultados de este proyecto permitirán caracterizar a los alumnos que ingresan a las carreras ofrecidas en la FRRe, detectar los déficits que deberían trabajarse mediante una adecuada articulación con el nivel medio y aquellos que deberían abordarse desde la Universidad.

Los alumnos que ingresaron a la carrera de Ingeniería en Sistemas de Información en el año 2006 se encuentran representados en esta muestra y a ellos también se aplicarán los resultados obtenidos. De todos modos, estos resultados no reflejarán todos los aspectos a tener en cuenta para analizar el éxito y el fracaso estudiantil. En el proyecto descrito se ha centrado el estudio en factores internos del alumno y no en otros que permitan dilucidar si la formación previa es adecuada, si las me-

todologías de enseñanza o los diseños curriculares influyen en el rendimiento y en qué medida lo hacen, lo cual constituye una línea de investigación futura.

El trabajo que presentamos en este artículo parte de la hipótesis que la enseñanza de la programación funcional en los primeros años de carreras de Informática y en Ingenierías No Informáticas podría facilitar el aprendizaje de las técnicas básicas de programación. La propuesta tiene origen en el trabajo encomendado por el profesor del curso sobre *Programación Funcional* dictado en el marco del Doctorado en Ingeniería de Sistemas y Computación de la Universidad de Málaga.

Respecto de los antecedentes analizados se destacan dos que consideramos relevantes para la experiencia. Uno de ellos [15] plantea la utilización de un lenguaje funcional (Miranda) para la enseñanza de diferentes tópicos del currículo de carreras de Informática. El segundo [9] plantea la enseñanza de las técnicas elementales de programación mediante el uso de un lenguaje funcional.

La propuesta educativa que se informa en [15] propone la utilización del lenguaje funcional Miranda como medio para enseñar los siguientes tópicos del plan de estudios de Ciencias de la Computación: Programación inicial, Lenguajes Formales, Arquitectura de computadoras, Semántica Formal y Computación Gráfica. La metodología de enseñanza es centrada en el alumno y la evaluación se realiza mediante una mezcla de evaluación continua y exámenes escritos. Entre sus conclusiones se destaca que al utilizar un mismo lenguaje para enseñar distintas asignaturas, los estudiantes pueden concentrarse en las nuevas ideas que se le presentan en lugar de considerar la forma particular en que están escritas; se trata de un lenguaje con control de tipos estático lo cual facilita la corrección; la elegancia matemática de los lenguajes funcionales constituyen una oportunidad para desarrollar pruebas formales en una variedad de áreas como, por ejemplo, máquinas de simulación y compilación; refuerza la exposición inicial a un lenguaje funcional en una institución que privilegia la utilización de este paradigma.

En [9] se plantea que enseñar programación funcional pura en cursos iniciales es perjudicial para el currículo y para la promoción del paradigma. Esta afirmación no implica que sus autores consideren adecuado enseñar programación procedural, orientada a objetos o lógica puras. La propuesta consiste en enseñar, mediante el lenguaje funcional Haskell, las técnicas elementales de programación y conceptos esenciales de la computación para fomentar el pensamiento analítico y la capacidad de resolución de problemas. La hipótesis es que los lenguajes funcionales puros son ideales para cursos iniciales pero sólo si la enseñanza se focaliza en conceptos generales en lugar de centrarse en aspectos específicos de la programación funcional. Los fundamentos de esta afirmación son: la semántica clara de los lenguajes funcionales facilita la comprensión de las técnicas elementales de programación y el razonamiento formal; la estrecha relación entre teoría y práctica en los lenguajes funcionales facilitan su enseñanza en el aula; el alto nivel de expresividad de estos lenguajes permite abordar tempranamente problemas complejos de programación, trasladando el eje de la clase, desde el mecanismo de programación hacia la solución del problema; son lenguajes de tipos, lo cual favorece la estructuración del proceso de resolución de problemas; la sintaxis limpia y ortogonal de Haskell ayuda a relegar la sintaxis a un segundo plano y concentrarse en conceptos generales de programación. En síntesis, los lenguajes funcionales permiten reemplazar el estilo de enseñanza centrado en el lenguaje por un estilo centrado en el concepto donde el lenguaje de programación es nada más que un medio para que el profesor ilustre los principios fundamentales de la informática.

En los dos artículos tomados como referencia se afirma que los resultados obtenidos son altamente satisfactorios ya que ambos concluyen que los lenguajes funcionales facilitan la enseñanza de la programación. El artí-

culo [15] también corrobora que el aprendizaje de otros tópicos del currículo de carreras de ciencias de la computación es facilitado por estos lenguajes. Los autores de [9] postulan, basados en su experiencia docente y en encuestas tomadas a los alumnos, que la utilización del lenguaje funcional Haskell permite, además, la introducción de conceptos esenciales de la informática y el desarrollo de la capacidad para resolver problemas.

En los siguientes apartados se explicarán la metodología, resultados y conclusiones de la experiencia que se presenta.

2. Metodología

2.1. Diseño de la experiencia

La propuesta se llevó a cabo en las Facultades Regional Resistencia (FRRe) y Regional Rafaela (FRRa), ambas dependientes de la Universidad Tecnológica Nacional (U.T.N.), en las asignaturas:

- *Paradigmas de Programación* que se dicta en el 2do año de la carrera de Ingeniería en Sistemas de Información (I.S.I.) de la FRRe.

- *Introducción a la Informática* que se dicta en primer año de la carrera de Ingeniería Civil y *Programación en Computación* que se dicta en 2do año de Ingeniería Electromecánica, ambas de la FRRa.

Los objetivos planteados fueron:

1. Evaluar si la enseñanza de un lenguaje funcional en primer año de carreras de Ingeniería no informática facilitan el aprendizaje de la programación y si generan mayor interés en la solución de problemas de otras áreas.

2. Evaluar si la resolución de problemas mediante programación funcional mejora el proceso de resolución de problemas en alumnos de 2do año de ISI.

3. Evaluar si la curva de aprendizaje difiere entre ambos perfiles de alumnos (Ingenieros tradicionales e Ingenieros en Sistemas). En caso de que la respuesta fuese positiva debería analizarse cuales son las características que facilitan o dificultan el aprendizaje en el alumno promedio de cada carrera.

Para llevar adelante la misma se propuso impartir, en ambas carreras, los siguientes conceptos del lenguaje Haskell: Tipos, Constantes, Operadores Matemáticos y Lógicos, Funciones, Listas, Cadenas de Caracteres, Tuplas, Patrones para números, Patrones para listas, Patrones para tuplas.

En función del momento en que se dictan las asignaturas involucradas en la experiencia, la propuesta se llevó a cabo durante el segundo cuatrimestre del ciclo lectivo 2007.

En *Paradigmas de la Programación* se dictan contenidos teóricos y prácticos relacionados con los paradigmas actuales, correspondiéndole al paradigma funcional la última etapa de dictado que suele ser de un mes o menos.

En el caso de *Introducción a la Informática* y *Programación en Computación*, además de conceptos generales de programación e implementación mediante un lenguaje, se imparten conceptos básicos sobre Hardware, por lo que también la enseñanza de Haskell se realizó durante el último mes de clases.

También se diseñó y aplicó, al finalizar el cursado, una encuesta dirigida a los alumnos con el objetivo de obtener una retroalimentación de la propuesta.

2.2. Implementación de la experiencia

2.2.1. Caracterización de muestras

Facultad Regional Resistencia

En la asignatura Paradigmas de Programación, el universo sobre el cual se realizó la experiencia está conformado por sesenta alumnos, todos los cuales ya habían cursado una asignatura del área Programación y al menos un taller de lenguajes.

Para llevar adelante la propuesta se dividió al curso en dos grupos, uno de los cuales debería resolver los prácticos planteados mediante lenguaje *C o Pascal* y el otro utilizando *Haskell*. Ambos grupos se conformaron con alumnos de características similares: todos habían cursado una materia introductoria de programación (Algoritmos y Estructuras de datos); se distribuyeron equitativamente entre ambos grupos según su año de ingreso; tenían fijados conceptos tales como: datos, tipos, abstracciones, paso y evaluación de parámetros y nociones de programación imperativa, orientada a objetos y lógica; conocían al menos un lenguaje de programación, dado que para aprobar las materias del área Programación de primero y segundo año es requisito aprobar los talleres de Pascal y C++ o Java respectivamente. En general, los alumnos manifiestan haber programado en más de un lenguaje, aunque ninguno de ellos pertenece al paradigma funcional.

Respecto del 40% de alumnos que respondió la encuesta, el 50% reconoce haber tenido problemas para aprobar la asignatura previa (Algoritmos y estructuras de datos), en tanto que alrededor del 25% manifiesta haberla recursado (Fig.1).

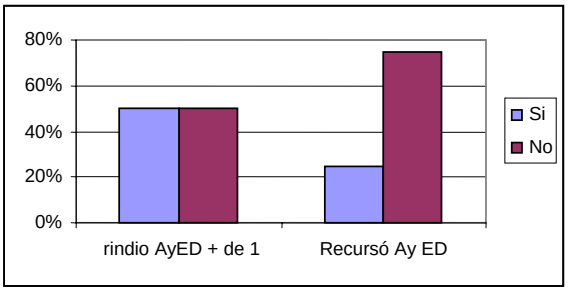

Fig.1. Rendimiento en Algoritmos y Estructuras de Datos

También se verifica que más del 70% de los alumnos que cursaron esta asignatura se retrasaron en el cursado de la carrera, correspondiendo a un 29,4% un retraso superior a un año (Fig.2).

Facultad Regional Rafaela

Los treinta y cinco alumnos que participaron de la experiencia pertenecen a carreras de Ingeniería no Informáticas: primer año de Ingeniería Civil y 2do año de Ingeniería Electromecánica. Se trata de alumnos sin conocimientos previos en programación, con una especial predilección hacia materias de las áreas Matemática y Física.

El desempeño de estos alumnos se contrasta respecto del rendimiento de los cincuenta y dos alumnos que cursaron las asignaturas entre los años 2004 y 2006.

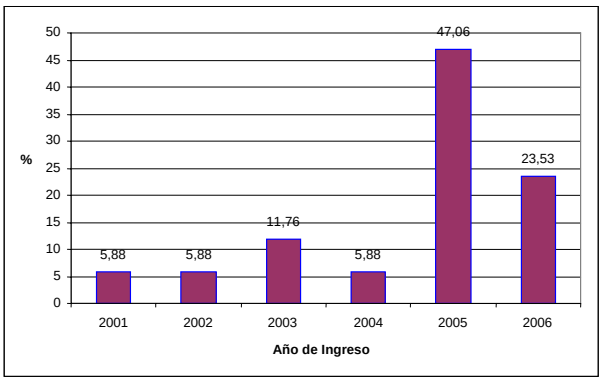

Fig.2. Distribución de alumnos según año de ingreso

2.2.2. Aplicación de la experiencia en el segundo año de una carrera informática

La experiencia se llevó a cabo luego de que los alumnos cursaron prácticamente el 85% de la asignatura. Durante las clases del paradigma funcional se realizó una presentación sobre sus orígenes y fundamentos, se plantearon las características principales del paradigma; se profundizó en conceptos tales como recursividad, polimorfismo, emparejamiento de patrones y evaluación perezosa. Además se llevó a cabo un repaso de estructuras de datos y tipos, tales como tuplas y listas.

Luego de la introducción teórica se plantearon ejercicios para resolución en clases. La metodología fue la siguiente: se dividió el total de alumnos en dos grupos; el primero resolvió los ejercicios propuestos mediante programación funcional usando el lenguaje

Haskell; el segundo lo hizo mediante el lenguaje C. El objetivo fue evaluar criterios tales como tiempo y facilidad de resolución, así como la simplicidad en los resultados obtenidos en ambos paradigmas.

El estilo de ejercicios sugeridos para su resolución fue orientado al trabajo con listas como, por ejemplo, encontrar la cabeza y cola de una lista, encontrar los n primeros elementos de una lista, eliminar n elementos de una lista, definir su tamaño o remover todas las ocurrencias de un elemento x de la lista.

Finalmente se impartieron encuestas a los alumnos con la finalidad de conocer su evolución en la carrera (como ser el año de ingreso a la Facultad y el rendimiento en materias correlativas a Paradigmas de Programación) como así también la opinión que les merece el lenguaje Haskell respecto de su simplicidad, sintaxis y utilidad.

También se requirió la presentación de los ejercicios resueltos en clase para evaluar la simplicidad y la corrección de los mismos.

2.2.3. Aplicación de la experiencia en el primer y segundo año de una carrera no informática

Las asignaturas que se dictan en la Facultad Regional Rafaela de la UTN son Programación en Computación (desde el año 2001) en la carrera de IEM y Fundamentos de Informática (desde el año 2004) en la carrera de IC. En ambas asignaturas el programa se divide en dos partes, por un lado se realiza una introducción al hardware (7 clases) y en una segunda etapa se dicta una introducción al diseño de algoritmos y resolución de problemas (9 clases). Hasta el año 2006, inclusive, se enseñó el paradigma procedural en ambas carreras. Los lenguajes utilizados fueron Pascal (en su versión Borland Turbo Pascal) en la carrera de IEM hasta el año 2004. A partir de ese año se comenzó a utilizar el lenguaje Basic (en su versión Microsoft Visual Basic) en ambas carreras, incorporando la utilización de una interfaz visual con el uso de eventos.

Para presentarse a rendir el examen final de la asignatura, los alumnos deben desarrollar una aplicación orientada a resolver un problema planteado en alguna de las asignaturas que cursaron hasta el momento. En general eligen problemas de las áreas matemática o física. Uno de los principales inconvenientes es el tiempo que les demanda aprobar la asignatura. Según pudieron corroborar los docentes, el principal inconveniente de los alumnos radica, no en la comprensión del problema a resolver, sino en el diseño del algoritmo y la codificación del mismo, lo cual les lleva aproximadamente un año.

En el año 2007 se impartió por primera vez el paradigma funcional, en reemplazo del procedural utilizado hasta ese momento, mediante la utilización del lenguaje Haskell. Los ejercicios que se desarrollaron en clases fueron los mismos que se plantearon en años anteriores, orientados a resolver problemas de índole matemática o física tales como: dada una lista de valores hallar máximos o mínimos, ocurrencias de un valor determinado, entre otros.

La evaluación de resultados se realizó mediante una medición del rendimiento de los alumnos en los exámenes finales y mediante consultas con los estudiantes que participaron de la experiencia.

3. Resultados

La enseñanza de la programación funcional en el segundo año de una carrera Informática permitió obtener los siguientes resultados: en una primera apreciación, durante el dictado de las clases, se percibió que los dos grupos, quienes resolvieron los problemas con Haskell y quienes lo hicieron con C, resolvieron con igual facilidad los ejercicios propuestos y lo hicieron en tiempos similares. En muchos casos los han resuelto más rápidamente con Haskell. Es importante recordar que los alumnos ya tienen conocimientos previos del lenguaje C, en tanto que esta experiencia constituyó su primera incursión en un lenguaje funcional. Esto podría atribuirse a

que la sintaxis de Haskell es bastante intuitiva y que no demanda demasiado tiempo comprender y asimilar los elementos más simples del lenguaje. Inclusive demanda menos esfuerzo al evaluar la corrección de los mismos por parte de alumnos y docentes.

Del análisis de las encuestas, respondidas por un 40% de los estudiantes que participaron de la experiencia, surge que la mayoría incursionó en varios lenguajes de programación pertenecientes a los paradigmas imperativo, orientado a objetos y lógico, pero ninguno había utilizado anteriormente un lenguaje funcional. Este resultado refuerza la apreciación anterior.

Consultados sobre las dificultades en el aprendizaje de la programación, si bien el 71% de los alumnos reconoce que les resultó relativamente simple asimilar la lógica de la programación, la gran mayoría afirma que los conceptos que les demandaron mayor esfuerzo fueron las abstracciones y las estructuras de datos estáticas y dinámicas.

Al referirse a las características del lenguaje Haskell, el 93% de los encuestados considera que facilita el aprendizaje de las técnicas básicas de programación y el 81% considera que permite razonar la solución de problemas sin preocuparse por una sintaxis compleja. Sin embargo, sólo el 25% opina que sería conveniente su enseñanza en el primer año de la carrera (Fig.3 y 4). La mayoría sostiene que este tipo de lenguajes serían de mucha utilidad luego de haber cursado los primeros años de la carrera y después de haber asimilado los lenguajes de programación imperativos, debido a que consideran a estos últimos más naturales en su esencia.

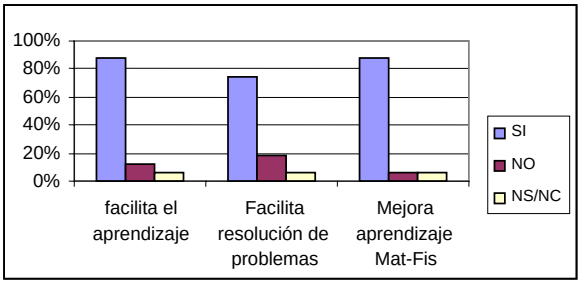

Fig.3. Ventajas del lenguaje Haskell según los alumnos

Al analizar los programas desarrollados por los estudiantes quedó evidenciado que, debido a la naturaleza declarativa del lenguaje funcional, la solución mediante el lenguaje Haskell siempre fue más simple y concisa, a pesar que la mayoría de los alumnos utilizó un proceso recursivo y no iterativo para la solución.

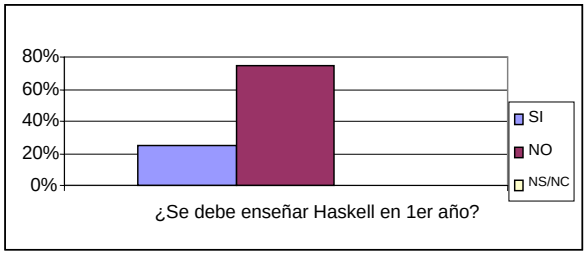

Fig.4. Conveniencia de enseñar Haskell en primer año según los alumnos

Con respecto a los resultados obtenidos en las asignaturas de programación de carreras no Informática, fueron superadas las expectativas de los docentes ya que a los alumnos les resultó sumamente fácil y natural diseñar algoritmos y programarlos en Haskell. No sólo resolvieron la guía de prácticos completa, algo que no había ocurrido nunca hasta el 2007, sino que solicitaron a la cátedra la edición de una guía complementaria, que fue resuelta completamente por el 70% de los alumnos.

Con respecto al rendimiento en los exámenes finales, también los resultados excedieron las expectativas ya que, apenas finalizado el dictado de la materia, más del 80% de los alumnos rindieron y aprobaron con nota superior a 7 (88% de alumnos de IC y 80% de IEM). La comparación del rendimiento de la cohorte 2007 con las cohortes anteriores (2004 a 2006) se muestra en las Fig.5 y 6.

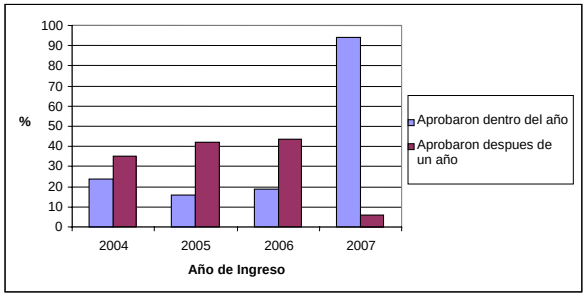

Fig.5. Rendimiento por cohorte

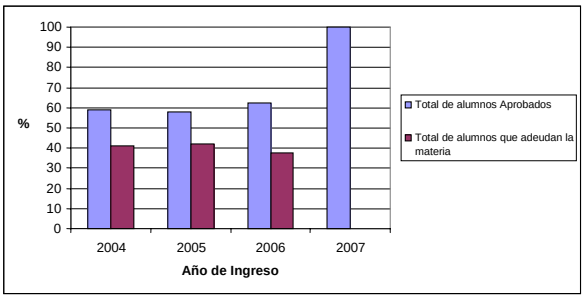

Fig.6. Aprobados por cohorte

Un resultado inesperado fue el examen de dos alumnos de la carrera de IEM que habían cursado el paradigma procedural y ya habían rendido la asignatura en dos oportunidades con resultado negativo. Ambos se presentaron a rendir por tercera vez en Diciembre de 2007 resolviendo el trabajo final y el examen mediante programación funcional, ya que habían preparado el final con un estudiante que cursó en el 2007. Ambos alumnos manifestaron que no habían tenido inconvenientes en aprender el lenguaje, pese a no haberlo cursado, y recomendaron mantenerlo ya que les resultó sencillo y sumamente práctico para resolver problemas.

4. Conclusiones

En base a los resultados presentados es posible concluir que la programación funcional es muy simple de entender y aplicar, tanto para alumnos novatos como para aquellos que ya han cursado asignaturas iniciales de programación, sean éstos de carreras Informáticas como de carreras de Ingeniería no Informáticas.

A pesar de las opiniones vertidas por los alumnos de ISI respecto de la enseñanza de la programación funcional en el primer año, las cuales no coinciden con sus apreciaciones respecto de la facilidad del lenguaje y las ventajas para enfocarse en la resolución de problemas, se considera como línea futura la posibilidad de llevar a cabo una experiencia en este sentido. Refuerza esta afirmación el hecho de que entre los conceptos que más dificultades presentan a los alumnos se encuentran el aprendizaje de las estructuras de datos, las abstracciones y la recursividad, todos conceptos muy simples de comprender mediante la utilización de lenguajes funcionales.

Ambas experiencias permiten acordar, al menos en una primera instancia, con los autores de los artículos analizados, [9] [15], respecto de las ventajas de enseñar un lenguaje funcional en los primeros años de carreras Informáticas, y se hace extensivo a carreras de Ingeniería no Informáticas. Se considera necesario mantener la propuesta en el tiempo, en el caso de las carreras IC e IEM, para verificar si los resultados se mantienen en sucesivos ciclos lectivos. En el caso de la experiencia realizada en la carrera de ISI debería analizarse la posibilidad de llevar a cabo la propuesta de enseñar el lenguaje Haskell a un grupo de alumnos de primer año, como exigencia para aprobar la asignatura del área Programación de ese nivel, y realizar un seguimiento de su desempeño en los dos primeros años comparados con aquellos que aprenden inicialmente lenguajes del paradigma imperativo.

No ha sido posible evaluar la curva de aprendizaje entre ambos tipos de alumnos debido a que el grupo de la carrera Informática, a diferencia de los de carreras no Informáticas, tenía conocimientos previos de programación. Esta evaluación será posible cuando se aplique la experiencia en grupos de ingresantes a la carrera de ISI.

Por último, cabe destacar la opinión de uno de los autores de este trabajo quien manifiesta que es posible que el cambio de paradigma le resulte más difícil al docente que al alumno. Esto se debe a que, en general, quienes dictan estas asignaturas han desarrollado sus carreras profesionales y docentes en el paradigma imperativo por lo cual el proceso de cambio les demanda más tiempo y esfuerzo que a los alumnos.

5. Referencias Bibliográficas

[1] A. Ferreira Szpiniak, G.A. Rojo. (2006) Enseñanza de La Programación. TE&ET – Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología. Nro.1

[2] A. González, G. Gorga, M.C.Madoz, C. Sanz. (2007) La importancia de la modalidad blended learning. Análisis de la experiencia educativa. II Congreso de Tecnología en Educación y Educación en Tecnología. Argentina.

[3] B. Fainholc. (2006) "Optimizando el uso de las TICs en Educación". Edutec – Revista electrónica de Tecnología Educativa. Nro 22.

[4] C. Luna, M. Pedemonte, M.Viera, E.Fraschini. (2007) Organización para un curso de programación en un contexto de pasividad. Resultados tras experiencia de cuatro años. TE&ET – Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología. Nro.2

[5] C. Madoz, G.Gorga. Análisis del proceso de articulación para alumnos de Informática, utilizando herramientas de Educación a Distancia. TE&ET – Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología. Nro.1.

[6] M.E. Caspersen, K.D. Larsen, J. Bennedsen. (2007). Mental models and programming aptitude**.** En Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education. Dundee, Scotland.

[7] M. Mc. Cracken, V.Almstrum, D.Diaz, M.Guzdial, D.Hagan, Y.B.D.Kolikant, C.Laxer, L.Thomas, I.Utting y T.Wilusz. (2001). A multinational, multi-institutional study of assessment of programming skills of first-year CS students. En Working group reports from ITiCSE on Innovation and technology in computer science education. Cantebury, UK. ACM Press

[8] M. Mc Gaul, M.F. Lopez, E.F.Fernandez, P.del Olmo. (2006) Los productos interactivos: haciendo clic en la Articulación. TE&ET – Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología. Nro.1

[9] M. M. T. Chakravarty, G. Keller. (2002) The Risks and Benefits of Teaching Purley Functional Programming in First Year. Proceedings of the International Workshop on Functional and Declarative Programming in Education (FDPE 2002).

[10] N. Figueroa, Z.Cataldi, P.Méndez, F. Lage, M.E. Vigliecca, G.Krauss (2006). Herramienta automatizada para la determinación de los estilos de aprendizaje en ingresantes a curso de Programación Básica. Anales Congreso Argentino de Ciencias de la Información y las Comunicaciones – CACIC 2006.

[11] P. Calvo, F.Salgueiro, Z. Cataldi, F. Lage. (2006) Sistemas Tutores Inteligentes Multiagentes: Los Agentes Docentes en el Módulo Tutor. Anales Congreso Argentino de Ciencias de la Información y las Comunicaciones – CACIC 2006.

[12] R. Lister, E.S. Adams, S. Fitzerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J.E. Mostr, K. Sander, O. Seppälä, B. Simon y L. Thomas (2004). A multinational study of reading and tracing skills in novice programmers. En 2004 ITiCSE working group report, 119-150.

[13] S. Dehnadi. (2006) Testing programming Aptitude. En Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group, 22-37, Brighton, UK.

[14] S. Dehnadi and R. Bornat. (2006), The camel has two humps. En www.cs.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

[15] S. J.Thompson, S. Hill (1995) Functional Programming Through the Curriculum. En Proceedings of the First international Symposium on Functional Programming Languages in Education. Eds. Lecture Notes In Computer Science, vol. 1022. Springer-Verlag, London, 85-102.

[16] S. Mariño, M.V. López. (2007) Aplicación del modelo b-learning en la asignatura Modelos y Simulación de las carreras de sistemas de la FACENA-UNNE". Edutec – Revista electrónica de Tecnología Educativa. Nro. 23.

[17] Y. P. Perez, L.M.Lopez, Multiparadigma en la enseñanza de la programación. Universidad Nacional del Comahue. Argentina.

[18] Z. B. Rosanigo, A.B. Paur, P. Bramati, H. Bramati (2006) Enseñar y aprender: un constante reto. Anales Congreso Argentino de Ciencias de la Información y las Comunicaciones – CACIC 2006.