Artículo Invitado

Evolución de un curso inicial de programación a un enfoque multiparadigma. Análisis y resultados

Evolution of a Programming Initial Course to a Multi Paradigm approach. Analysis and Results

Laura C. De Giusti[1,2], Victoria Sanz[1,2], Armando De Giusti[1,3]

[1] III-LIDI, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina
[2] CIC, Buenos Aires, Argentina
[3] CONICET, Argentina

ldgiusti@lidi.info.unlp.edu.ar, vsanz@lidi.info.unlp.edu.ar, degiusti@lidi.info.unlp.edu.ar

Recibido: 02/07/2019 | **Aceptado**: 25/11/2019

Cita sugerida: L. C. De Giusti, V. Sanz, A. De Giusti, "Evolución de un curso inicial de programación a un enfoque multiparadigma. Análisis y resultados," *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, no. 24, pp. 7-14, 2019. doi: 10.24215/18509959.24.e01

Resumen

En este trabajo se analiza la evolución de un curso inicial de programación, que ha sido reestructurado en dos asignaturas cuatrimestrales, incorporando en el segundo cuatrimestre trabajo experimental en tres paradigmas (imperativo, orientado a objetos y concurrente). Los temas principales incluyen una discusión sobre el impacto de este enfoque en el alumno de una carrera universitaria de informática, los criterios de selección de los paradigmas elegidos y aspectos salientes de la metodología y herramientas de implementación en el aula. Se analizan las demandas de la industria del software y el cambio tecnológico en la elección este enfoque, y se compara este nuevo curso con la solución clásica implementada anteriormente. Se presentan las ventajas y desafíos de este enfoque multiparadigma, luego de tres años de trabajo con más de 800 alumnos. Las conclusiones se enfocan en el impacto en el aprendizaje del alumno, considerando el efecto directo y el efecto indirecto sobre los cursos posteriores de la carrera.

Palabras clave: Programación; Algoritmos; Taller de programación; Paradigmas; Lenguajes.

Abstract

This paper analyzes the evolution of an introductory programming course, which has been restructured in two four-month courses, and includes in the second period practical sessions in three programming paradigms (imperative, object-oriented and concurrent). The main topics include a discussion of the impact of this approach on Computer Science students, the criteria for choosing the mentioned programming paradigms, the teaching methodology and strategies for implementation in the classroom. Also, the demands of the industry and the technological change that led to this new approach are analyzed, and this new course is compared with the previous traditional course. Furthermore, the advantages and challenges of this multi-paradigm approach, after 3 years of work with more than 800 students, are presented. The conclusions focus on the impact on student learning, considering the direct effect and the indirect effect on subsequent courses in the career.

Keywords: Programming; Algorithms; Programming workshop; Paradigms; Languages.

1. Introducción

En los planes de estudio de las carreras de Informática es habitual tener 1 primer curso anual (que puede dividirse en dos asignaturas cuatrimestrales) donde se aborda el aprendizaje y práctica inicial de programación. En lo que sigue de este trabajo, lo denominaremos genéricamente CS1. Los contenidos de este curso generalmente se enfocan en el análisis de problemas resolubles por computadora, su modelización, abstracción y expresión simbólica de la/s solución/es utilizando al menos un paradigma y lenguaje de programación [1, 2, 3, 4]. En el recorrido del curso se introducen las nociones de tipos de datos, estructuras de datos lineales y no lineales y conceptos importantes para la formación del alumno tales como reusabilidad, recursividad, legibilidad, corrección y eficiencia de las soluciones [5, 6, 7]. CS1 normalmente es el punto inicial del trayecto curricular[1] denominado "*Algoritmos y Lenguajes*" que, en el caso de las Licenciaturas en Informática y Sistemas de la UNLP, tiene de 6 a 8 asignaturas cuatrimestrales adicionales que cubren conceptos fundamentales de paradigmas y lenguajes que el alumno utilizará en su carrera universitaria.

A lo largo de los años en el ámbito académico se ha desarrollado una discusión sobre la elección del lenguaje y paradigma de programación más adecuado e incluso sobre si es conveniente introducir más de un paradigma (y su lenguaje asociado) en estos cursos universitarios iniciales de las carreras de Informática [8, 9, 10, 11]. Esta discusión tiene relevancia en la formación de profesionales de Informática porque es conocida la incidencia que tiene el aprendizaje del alumno en un curso CS1 para su evolución en la carrera e incluso en la posibilidad de abandono de los estudios [12, 13].

Dejando de lado una etapa inicial que va de los años 60 a los 80, en la que el paradigma imperativo y la programación estructurada fueron adoptados en forma generalizada en los estudios universitarios iniciales de programación, hay dos elementos actuales que condicionan/enriquecen la decisión a tomar: (1) la industria del software impulsa la formación en la programación y diseño orientado a objetos [14]; (2) el cambio en la arquitectura de los procesadores desde 2005 con la aparición de los "*multicores*" promueve la introducción en forma temprana de los conceptos de concurrencia y paralelismo [15, 16].

Por otro lado las características del alumno inicial de las carreras de Informática (al menos en Argentina) deben considerarse al planificar los objetivos, alcances e instrumentos pedagógicos a utilizar en este primer ciclo de aprendizaje de la programación o "*curso CS1*":

- La formación previa en Informática es heterogénea y limitada. El currículo de la Escuela Media no incluye en todos los casos asignaturas formativas en programación. Muchas veces el aprendizaje es "*instrumental*" con escasa conceptualización del modelo de problema que se resuelve. La formación de los docentes también es muy heterogénea y condiciona la adaptación al cambio tecnológico[2].
- Los nativos digitales son interactivos por naturaleza, trabajan por prueba y error, están preparados para aprender sobre diferentes tecnologías, son aficionados a vincularse colaborativamente en red, no están entrenados para la abstracción y requieren una alta motivación para mantener su concentración en el aprendizaje [17].
- Las competencias que se logran fortalecer a lo largo del primer año de una carrera universitaria de Informática y en particular como resultado del aprendizaje adquirido en un curso CS1 son muy importantes para el éxito en la carrera[3]. Desarrollar estas competencias require un esfuerzo por el aspecto motivacional del alumno, lo cual generalmente se logra con tareas experimentales de programación de complejidad baja/media que fortalezcan los conceptos teóricos y la formación buscada en el alumno. Esto exige una cuidadosa planificación del balance "*teórico-práctico*" de un curso CS1, sus mecanismos de seguimiento y evaluación, en lo posible teniendo en cuenta la actualización tecnológica y las demandas de la industria [18].

En este trabajo trataremos de reflejar sintéticamente los resultados evolutivos de un curso de programación para alumnos de primer año en la Facultad de Informática de la Universidad Nacional de La Plata, que comenzó siendo una materia anual llamada "*Programación de Computadoras*" en la década de los 80-90, evolucionó a otro curso anual de "*Algoritmos, Datos y Programas*" en el que se agregaron al paradigma imperativo y el lenguaje Pascal los conceptos de programación orientada a objetos y programación concurrente con un trabajo experimental limitado y la transformación en los últimos 3 años a dos cursos cuatrimestrales, uno de "*Conceptos de Algoritmos, Datos y Programas*" que se continúa en un "*Taller de Programación*" en el que los alumnos trabajan en máquina en 3 paradigmas: imperativo, orientado a objetos y concurrente/paralelo [19, 20]. El núcleo del trabajo es analizar la experiencia del "*Taller de Programación*" multiparadigma, luego de 3 ciclos completos con más de 800 alumnos atendidos y habiendo relevado el impacto de este curso sobre el aprendizaje del alumno y sobre los cursos sucesivos de segundo y tercer año de la carrera (Licenciatura en Informática y Licenciatura en Sistemas) donde se profundizan los conceptos introducidos en este Taller.

2. Primeros cursos de programación en carreras de Informática

2.1. Curso anual convencional

Podemos sintetizar los objetivos de un curso CS1 anual "*clásico*" en los siguientes puntos referidos a las competencias esperadas a alcanzar por el alumno:

- Analizar problemas resolubles con computadora, poniendo énfasis en la modelización, abstracción, y en la descomposición de los mismos. Obtener una expresión sintética y precisa de los problemas, con una documentación de la metodología de trabajo.
- Capacidad para el estudio, expresión simbólica, implementación y evaluación de algoritmos, orientando los mismos a la resolución de los módulos en que se descomponen los problemas, a partir de un paradigma (imperativo u orientado a objetos).
- Conocimiento y manejo de estructuras de control y estructuras de datos, tipos de datos y abstracción de datos.
- Conocimiento y aplicación de conceptos fundamentales en Informática tales como reusabilidad, recursión, legibilidad, corrección y eficiencia de los programas desarrollados.
- Lograr completar el ciclo "*del problema a su solución con computadora*", analizando simultáneamente algoritmos y datos.

Este conjunto de competencias/objetivos requieren el aprendizaje de una serie de temas teóricos (estructuras de control, estructuras de datos lineales y no lineales, tipos de datos, modularización, métodos de verificación y validación de algoritmos, recursión, métricas de eficiencia de algoritmos, etc.) los que se deben complementar y consolidar con trabajos experimentales en los que el alumno pase de la expresión simbólica de las soluciones a su implementación en un lenguaje real (lo que exige un conocimiento de la sintaxis y semántica del mismo) y la verificación de resultados contra lo especificado en el problema.

Tal como se indicó anteriormente este curso se resolvía en los años 80-90 con un paradigma de programación (imperativo) y un lenguaje de trabajo experimental (Pascal en nuestro caso) [21, 22].

2.2. Motivación para introducir múltiples paradigmas en primeros cursos de programación

La evolución de los cursos CS1 "*clásicos*" estuvo liderada por las dos líneas marcadas en la introducción:

- La primera está relacionada al cambio impuesto por la industria del software que requiere la formación en la programación y diseño orientado a objetos de los desarrolladores. Es opinión mayoritaria en el mercado laboral que el alumno tiene que adquirir habilidades en programación orientada a objetos en etapas tempranas de su carrera universitaria. Cualquier análisis de los lenguajes más utilizados actualmente en la industria ratifica este concepto.
- La segunda está relacionada a la evolución de la arquitectura de los procesadores, que ha cambiado fundamentalmente desde 2005 con la aparición de los "*multicores*", y por lo tanto promueve la introducción en forma temprana de los conceptos de concurrencia y paralelismo ya que el alumno y el profesional no se encontrarán más con el modelo de "*máquina Von Neumann*" secuencial que caracterizó el aprendizaje durante el siglo XX. Esto requiere un esfuerzo adicional por la mayor complejidad de la programación concurrente, respecto de la secuencial.

Asimismo, es interesante considerar que un enfoque multiparadigma nos permite una mejor articulación "*vertical*" con las asignaturas de segundo año, en las cuales se desarrolla el paradigma orientado a objetos y se ven diferentes lenguajes de programación, así como una mejor articulación "*horizontal*" con los conceptos de arquitectura de procesadores y de sistemas operativos, donde se analizan las arquitecturas actuales, la existencia de múltiples núcleos, la programación concurrente sobre los mismos y la administración de recursos compartidos por el sistema operativo[4]. En particular, el segundo año de las Licenciaturas en Informática y Sistemas de la UNLP presenta varias asignaturas correlativas al curso CS1 que incluyen la Programación Orientada a Objetos y los conceptos de Concurrencia.

2.3. Evolución a 2 cursos cuatrimestrales

Considerando lo anterior, en la Facultad de Informática UNLP se introdujeron cambios en el plan de estudios, y se reestructuró el curso anual "*clásico*" en dos cursos cuatrimestrales, uno de "*Conceptos de Algoritmos, Datos y Programas*" (CADP) centrado en los conceptos teóricos clásicos y que ejemplifica los mismos con el paradigma imperativo (utilizando los conocimientos previos del alumno y en particular los adquiridos en el módulo "*Expresión de Problemas y Algoritmos*" dictado durante el ingreso a la Facultad) y un segundo curso cuatrimestral teórico-práctico denominado "*Taller de Programación*" en el que los alumnos trabajan en máquina en 3 paradigmas: imperativo, orientado a objetos y concurrente/paralelo.

Los objetivos generales del "*Taller de Programación*" son:

- Ampliar el conjunto de estructuras de datos vistas en "*CADP*" con estructuras no lineales (árboles binarios) e introducir algoritmos fundamentales

sobre estructuras de datos lineales (listas simples y arreglos) y no lineales. Estos temas se desarrollaban en la segunda mitad del curso anual. Para el desarrollo experimental se utiliza el lenguaje Pascal, debido a que los alumnos ya trabajaron en este lenguaje en "*CADP*" y es adecuado para introducir los temas de la programación imperativa.

- Extender los conceptos de abstracción de datos para comprender la noción de objeto y analizar todas las características básicas de la programación orientada a objetos y los lenguajes asociados [23, 24]. Para la resolución de los trabajos experimentales, se adoptó el lenguaje de programación Java. Este mismo lenguaje es utilizado en materias correlativas de segundo año ("*Algoritmos y Estructuras de Datos*" y "*Seminario de Lenguajes*"), de esta manera se logra una mejor articulación. Asimismo, los conceptos de programación orientada a objetos se amplían en las materias "*Orientación a Objetos 1*" y "*Orientación a Objetos 2*" de segundo y tercer año respectivamente.
- Incorporar los temas conceptuales básicos de concurrencia (comunicación y sincronización) así como los problemas teóricos asociados (deadlock, fairness, prioridades). Para la parte práctica se utiliza el entorno y lenguaje de programación CMRE (*Concurrent Multi Robot Environment*) que permite expresar algoritmos concurrentes representando los múltiples procesadores que se comunican por elementos físicos (robots) que se mueven, comunican y sincronizan en un entorno visual virtual, con posibilidad de hacerlo por memoria compartida o por mensajes [25, 26]. Este entorno es utilizado por los alumnos ingresantes a la Facultad para resolver problemas secuenciales con un solo procesador o elemento físico (robot). Los conceptos de concurrencia se requieren para "*Introducción a los Sistemas Operativos*" de segundo año y se profundizan en tercer año en la asignatura "*Programación Concurrente*".

3. Taller de Programación multi-paradigma

3.1. Metodología y Organización

El Taller de Programación tiene como objetivo consolidar el aprendizaje de los fundamentos de cada paradigma y resolver problemas simples sobre máquina en cada uno de ellos, en un entorno real.

Este curso se estructura en 5 clases teórico-prácticas de 3 hs. por paradigma. En las clases el alumno trabaja interactivamente sobre máquina (configurada con todos los elementos requeridos), resuelve problemas en el aula y puede replicar el entorno fácilmente en su casa para continuar/profundizar las tareas.

El dictado del curso es secuencial por paradigma (Imperativo – Objetos – Concurrente) y al completar cada paradigma (módulo) hay una evaluación sobre máquina, con un problema concreto.

Según los resultados de las evaluaciones en el desarrollo del curso el alumno puede: (1) promocionar la asignatura (cursada y final) si aprueba los tres módulos; (2) obtener sólo la cursada en el caso de aprobar dos módulos quedando habilitado para rendir un examen final; (3) acceder a un recuperatorio, en caso de aprobar un sólo módulo, en el cual se evalúa cualquiera de los módulos desaprobados – aprobando esta instancia el alumno obtiene la cursada de la materia.

La realización de este curso multiparadigma, centrado en el trabajo experimental, requiere de un esfuerzo importante marcado por el número de docentes por grupo de alumnos y la infraestructura de equipamiento que se necesita. La Facultad de Informática de la UNLP ha asignado un equipo docente de 5 Profesores y 15 docentes auxiliares para atender unos 250 alumnos por año, divididos en grupos de 50. Al mismo tiempo ha desarrollado un equipamiento de Laboratorios Móviles con portables conectadas a Internet vía Wi-Fi que se pueden llevar a cualquier aula, de modo de trabajar con una máquina cada dos alumnos para el desarrollo del curso y una máquina por alumno para las evaluaciones.

Es fundamental una gran organización de la cátedra para que el alumno aproveche los tiempos en el aula y de interacción con los docentes. Hay que evitar que el alumno tenga dificultades secundarias (con su máquina, con la conectividad, en el acceso a los materiales, etc.) de modo que maximice el tiempo de aprendizaje.

Para el trabajo fuera del aula y con el objetivo de vincular a docentes y alumnos, se utiliza el entorno virtual de enseñanza y aprendizaje IDEAS[5] (disponible para los alumnos en todos los cursos de la carrera).

3.2. Contenidos

El objetivo principal del Taller es que el alumno aborde diferentes problemas e implemente soluciones correctas utilizando los tres paradigmas. En todos los casos el lenguaje de programación se elige en función de la facilidad de aprendizaje y su proyección futura. El trabajo experimental se complementa con temas teóricos que completan los dictados en el primer cuatrimestre (CADP) y se dictan los fundamentos de los paradigmas "*nuevos*" (objetos y concurrencia). Sintéticamente:

- En el módulo imperativo el alumno completa su aprendizaje abordando temas como: estructuras de datos no lineales, específicamente árboles binarios (definición, terminología, características, y operaciones) para lo cual es necesario introducir el concepto de recursividad,

implementación de algoritmos fundamentales sobre estructuras de datos estáticas y dinámicas: búsquedas, ordenación, merge.

- En el módulo orientación a objetos se desarrollan los conceptos básicos de estado y comportamiento de un objeto, la noción de clase e instancia así como el concepto de herencia y polimorfismo.
- En el módulo programación concurrente se explican los conceptos de proceso, comunicación y sincronización entre procesos y el empleo de memoria compartida y/o mensajes para esta comunicación.

4. Resultados y análisis

4.1. Impacto en la formación de los alumnos y desafíos para los docentes

Resulta claro que el cambio realizado a partir de 2015 ha generado un impacto sobre alumnos y docentes, con aristas positivas y dificultades concretas:

- Más allá de las cuestiones instrumentales de cada paradigma, los alumnos están obligados a desarrollar habilidades más generales, es decir entender el problema en sí, sus restricciones y posibles soluciones. Pensar en paradigmas alternativos para un problema (o una clase de problemas) es un aspecto formativo de especial interés en las carreras de Informática. El curso trata de abrir un panorama conceptual sobre las bondades relativas de cada paradigma, de modo que el alumno concluya el Taller con algunos criterios de selección del mismo, en función de la clase de problema a resolver.
- Al mismo tiempo para los alumnos la resolución concreta sobre máquina (utilizando un lenguaje de programación) es una motivación y una dificultad real, por los tiempos de aprendizaje y de experimentación. Incluso las evaluaciones sobre máquina pueden resultar "*a priori*" más complejas para el alumno. En el curso se enfatiza en las competencias relacionadas con la capacidad de autoaprendizaje y autoevaluación del mismo alumno.
- Para los docentes las exigencias (más allá del tiempo de atención e interacción en el aula) han sido el dominio de los paradigmas y lenguajes que se emplean en el Taller, la elaboración de trabajos experimentales que logren generar en el alumno las competencias buscadas y el trabajo mismo con máquinas, que resulta un desafío mayor que las soluciones "*en lápiz y papel*".
- Para una cátedra relativamente masiva (250 alumnos en 5 turnos diferentes) la coordinación de los docentes ha requerido un esfuerzo importante. El trabajo de taller teórico-práctico disminuye las distancias y favorece la interacción, pero exige un núcleo docente homogéneo y bien coordinado.

4.2. Resultados relacionados con el Taller de Programación

Luego de tres años y con más de 800 alumnos inscriptos en el Taller de Programación (todos los cuales habían superado el curso "*Conceptos de Algoritmos, Datos y Programas*" del primer cuatrimestre) podemos analizar algunos resultados cuantitativos y cualitativos:

- De los 280 alumnos que inician el curso, en promedio un 75% asiste, es decir se presenta a las evaluaciones de los 3 paradigmas/módulos. De este promedio de asistentes efectivos (210 alumnos por año de curso) un 81% aprueba los trabajos prácticos y promueve el final o queda habilitado para rendirlo. El 19% restante desaprueba el curso. Este resultado es claramente superior a la media de aprobación de los cursos del primer año de las Licenciaturas en Informática y Sistemas de la UNLP. En la Figura 1 se muestra resultados relacionados con la aprobación y desaprobación de trabajos prácticos del curso "*Taller de Programación*" para los años 2015, 2016 y 2017.

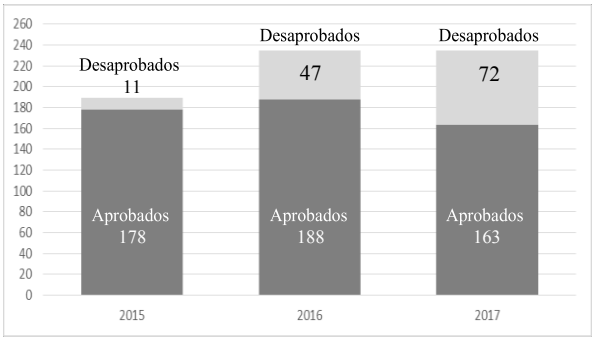

Figura 1. Cantidad de alumnos aprobados y desaprobados de la materia Taller de Programación

- La comparación de los resultados absolutos en términos de "*cursadas aprobadas*" del curso CS1 anual contra la combinación de 2 asignaturas cuatrimestrales no arrojó cambios significativos. Sin embargo, la solución adoptada desde 2015 genera un mayor número de alumnos "*promovidos*" que tienen sus 2 exámenes finales aprobados ("*Conceptos de Algoritmos, Datos y Programas*" y "*Taller de Programación*") al completar el primer año. Esto se debe a que para poder cursar "*Taller de Programación*" es necesario tener aprobada la cursada de "*Conceptos de Algoritmos, Datos y Programas*" del primer cuatrimestre. La misma correlación se aplica para rendir final.
- En la Figura 2 se muestra sintéticamente el

resultado de la encuesta realizada a alumnos sobre la dificultad que encontraron con la modalidad teórico-práctica del Taller multiparadigma. Se observa que un 26% de los alumnos consideró que el esfuerzo para aprobar el curso es superior al de otras asignaturas de 1er año, un 63% lo consideró normal y un 11% indicó que les resultó menos dificultoso que las otras asignaturas de 1er año (2 relacionadas con Matemáticas y 2 con Arquitectura de Computadoras).

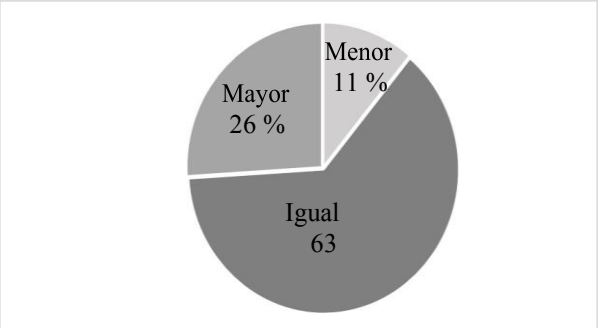

Figura 2. 280 Respuestas de alumnos que cursaron Taller de Programación durante los años 2016, 2017 y 2018 (expresado en porcentaje)

4.3. Resultados relacionados con asignaturas de 2do año y 3er año

- El impacto sobre 2do año se mide en diferentes asignaturas. Hemos analizado en detalle la evolución de las asignaturas "*Algoritmos y Estructuras de Datos*", "*Orientación a Objetos 1*" y "*Seminario de Lenguajes*", todas ellas correlacionadas con "*Taller de Programación*". El aumento en el porcentaje de trabajos prácticos aprobados en estas tres asignaturas es muy significativo, a partir de la introducción del Taller de Programación en 2015. Tal como se muestra en la Figura 3, las tres asignaturas han incrementado el porcentaje de alumnos que obtienen la cursada de 2015 a 2018 entre un 11% y un 34%.

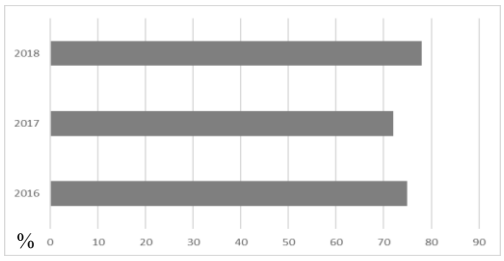

Figura 3. Porcentaje de respuestas positivas de los alumnos por año

- Con respecto a 3er año, nos interesó estudiar el impacto sobre el curso "*Programación Concurrente*". En principio se aprecia una mejora del orden del 7% en el porcentaje de aprobación de trabajos prácticos en 2017 y 2018 respecto de 2015.
- En la Figura 4 se muestra una encuesta de la percepción que tienen los alumnos de 2do año sobre la utilidad de haber aprendido más de un paradigma en "*Taller de Programación*". La utilidad percibida es de 75% (año 2016), 72% (año 2017), y 78% (año 2018).

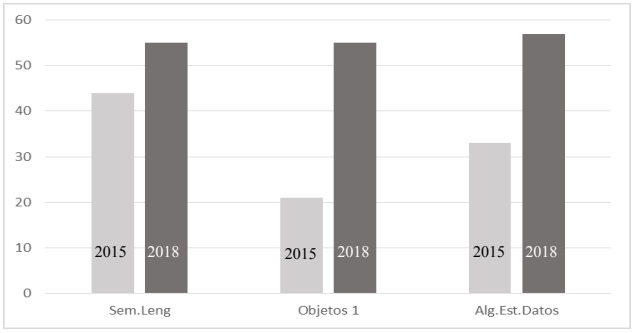

Figura 4. Porcentaje de Aprobados en 2015 y 2018 en las materias relacionadas con el Taller de Programación

4.3.1. Seguimiento personalizado

- Se hizo un estudio detallado de la trayectoria de alumnos que recorrieron exitosamente el Taller de Programación en 2015, 2016 y 2017 y cómo habían avanzado en sus carreras, poniendo el foco en las asignaturas correlativas ya mencionadas ("*Algoritmos y Estructuras de Datos*", "*Orientación a Objetos 1*", "*Seminario de Lenguajes*" y "*Programación Concurrente*"). En este estudio se incluye "*Introducción a los Sistemas Operativos*" que utiliza algunos conceptos básicos de concurrencia y se dicta en segundo año.
- Si tomamos el muestreo de los alumnos que promovieron Taller de Programación en 2015 nos encontramos que a 2018 el 97% de ellos obtuvieron la cursada de "*Orientación a Objetos 1*", el 93% la de "*Algoritmos y Estructuras de Datos*", 91,5% la de "*Introducción a los Sistemas Operativos*", 88% la de "*Seminario de Lenguajes*" y 65% la de "*Programación Concurrente*". Estos números son superiores al recorrido promedio de los alumnos y muestra el efecto beneficioso del aprendizaje en el Taller de Programación sobre las asignaturas relacionadas.

Conclusiones

Se analizó la evolución del curso inicial de programación (curso CS1) de la Facultad de Informática de la Universidad Nacional de La Plata (Argentina). En particular nos enfocamos en el Taller de Programación multiparadigma que se dicta en el segundo cuatrimestre, su implementación y resultados. Sintéticamente:

- Hay un impacto positivo en el aprendizaje y formación de los alumnos, que en particular se refleja sobre los cursos posteriores de la línea

Algoritmos y Lenguajes del plan de estudios.

- La motivación de los alumnos se incrementa claramente con el desafío de resolver un taller experimental con problemas en diferentes paradigmas.
- La elección de los tres paradigmas empleados (imperativo, orientado a objetos y concurrente) resulta lógica en función de los conocimientos previos del alumno, de las demandas de la industria del software y del cambio tecnológico. Esta elección facilita la articulación vertical y horizontal en el plan de estudios de la carrera.
- Los resultados cuantitativos y las encuestas con los alumnos son positivos respecto del modelo anterior centrado en un único paradigma.
- Hay un impacto indirecto sobre la actualización de los docentes del curso CS1 (ambos cuatrimestres) por las competencias a desarrollar con los alumnos.

Actualmente se han perfeccionado las herramientas requeridas para el desarrollo de los trabajos experimentales y se discute permanentemente el lenguaje a utilizar en cada caso, así como la extensión del enfoque multiparadigma a la Ingeniería en Computación que también se dicta en la Facultad.

Notas

[1] Puede acceder al documento curricular de la Red de Universidades Nacionales con Carreras de Informática (Argentina) desde http://redunci.info.unlp.edu.ar/docs/RecomendacionesCurriculares- LibroRedUNCI-2017-9-Completo.pdf

[2] Esto quedó reflejado en el informe del Ministerio de Educación de Argentina sobre las pruebas PISA 2012 en la Escuela Media (https://www.argentina.gob.ar/sites/default/files/informe_pisa_2012.pdf)

[3] Puede acceder a las competencias de la Licenciatura en Informática y Sistemas de la UNLP desde https://www.info.unlp.edu.ar/competencias-licenciatura-en-informatica/ y https://www.info.unlp.edu.ar/competencias-licenciatura-en-sistemas/

[4] Puede acceder a los planes de estudio de Licenciatura en Informática y Licenciatura en Sistemas (Facultad de Informática, UNLP) desde http://info.unlp.edu.ar/carreras-gradoarticulo/2015linuevo/ y http://info.unlp.edu.ar/carreras-gradoarticulo/plan-2015-licenciatura-en- sistema/ respectivamente.

[5] https://ideas.info.unlp.edu.ar

Referencias

[1] C. Leska, J. Barr, L.A. Smith King, "Multiple Paradigms in CS I", *ACM SIGCSE Bulletin*, vol. 28, no. 1, pp. 343-347, 1996.

[2] S. Perugini, "The design of an emerging/multi-paradigm programming languages course", *Journal of Computing Sciences in Colleges*, vol. 34, no. 1, pp. 52- 59, 2018.

[3] J. Reinfelds, "A three paradigm first course for CS majors", *ACM SIGCSE Bulletin*, vol. 27, no. 1, pp. 223-227, 1995.

[4] M.Vujošević-Janičić, D. Tošić, "The role of programming paradigms in the first programming courses", *The Teaching of Mathematics*, vol. 21, pp. 63-83, 2008.

[5] A. De Giusti, C. Madoz, E. Ibañez, "Propuesta de Enseñanza del curso 'Algoritmos, Datos y Programas'", April 8, 2018. [Online]. Available: http://info.unlp.edu.ar/wp-content/uploads/2018/09/Algoritmos-Datos-y-Programas.pdf [Accessed June 13, 2019]

[6] E. Giangrande Jr, "CS1 programming language options", *Journal of Computing Sciences in Colleges*, vol. 22, no. 3, pp. 153–160, 2007.

[7] M. Hertz, "What do "CS1" and "CS2" mean?: investigating differences in the early courses", in Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10), 2010, pp. 199-203.

[8] K. Becker, "Back to Pascal: retro but not backwards", *Journal of Computing Sciences in Colleges*, vol. 18, no. 2, pp. 17–27, 2002.

[9] R. Close, D. Kopec, J. Aman, "CS1: perspectives on programming languages and the breadth-first approach", *Journal of Computing Sciences in Colleges*, vol. 15, no. 5, pp. 228-234, 2000.

[10] Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, J. Paterson, "A survey of Literature on the Teaching of Introductory Programming", *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 204-223, 2007.

[11] T. Vilner, E. Zur, J. Gal-Ezer, "Fundamental concepts of CS1: procedural vs. object oriented paradigm - a case study", *ACM SIGCSE Bulletin*, vol. 39, no. 3, pp. 171-175, 2007.

[12] D. F. Shell, L. Soh, A. E. Flanigan, M. S. Peteranetz, "Students' Initial Course Motivation and Their Achievement and Retention in College CS1 Courses", in Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16), 2016, pp. 639-644.

[13] C. Stephenson, A. D. Miller, C. Alvarado, L. Barker, V. Barr, T. Camp, C. Frieze, C. Lewis, E. Cannon Mindell, L. Limbird, D. Richardson, M. Sahami, E. Villa, H. Walker, S. Zweben, *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions,* ACM, 2018.

[14] S. Cass, "The 2018 Top Programming Languages", *IEEE Spectrum*, July 31, 2018. [Online], Available: https://spectrum.ieee.org/at-work/innovation/the-2018- top-programming-languages [Accessed June 13, 2019]
[15] K. B. Bruce, A. Danyluk, T. Murtagh, "Introducing Concurrency in CS 1", in Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10), 2010, pp. 224-228.
[16] L. De Giusti, F. Leibovich, F. Chichizola, M. Naiouf, A. De Giusti, "Incorporando conceptos en la enseñanza de Concurrencia y Paralelismo utilizando el entorno CMRE", in Actas del XXI (CACIC 2015), 2015, pp. 1212-1221.
[17] S. Magana, *Disruptive Classroom Technologies: A Framework for Innovation in Education*, Corwin Press, 2017.
[18] Forte, M. Guzdial, "Motivation and Non-Majors in CS1: Identifying Discrete Audiences for Introductory Computer Science", *IEEE Transactions on Education*, vol. 48, no. 2, pp. 248-253, 2005.
[19] A. De Giusti, C. Madoz, G. Gorga, L. De Giusti, V. Ainchil, I. Rodriguez, L. Marrero, E. Ibañez, "Propuesta de Enseñanza del curso 'Conceptos de Algoritmos, Datos y Programas'", April, 8, 2018. [Online]. Available: http://info.unlp.edu.ar/wp-content/uploads/2018/05/Conceptos-de-Algoritmos- Datos-y-Programas.pdf [Accessed June 13, 2019]
[20] A. De Giusti, G. Gorga, L. De Giusti, V. Ainchil, W. Hasperué, V. Sanz, E. Ibañez, "Propuesta de Enseñanza del curso 'Taller de Programación'", April, 8, 2018. [Online]. Available: http://info.unlp.edu.ar/wp- content/uploads/2018/05/Taller-de-Programaci%C3%B3n.pdf [Accessed June 13, 2019]
[21] S. S. Brilliant, T. R. Wiseman, "The first programming paradigm and language dilemma", *ACM SIGCSE Bulletin*, vol. 28, no. 1, pp. 338–342, 1996.
[22] D. Gries, "What should we teach in an introductory programming course?", *ACM SIGCSE Bulletin*, vol. 6, no. 1, pp. 81-89, 1974.
[23] M. H. Goldwasser, D. Letscher, "Teaching an Object- Oriented CS1 with Python", *ACM SIGCSE Bulletin*, vol. 40, no. 3, pp. 42-46, 2008.
[24] K. Sanders, L. Thomas, "Checklists for Grading Object-Oriented CS1 Programs: concepts and misconceptions", *ACM SIGCSE Bulletin*, vol. 39, no. 3, pp. 166-170, 2007.
[25] J. Castro, L. De Giusti, G. Gorga, M. Sanchez, M. Naiouf, A. De Giusti, "ECMRE: Extended Concurrent Multi Robot Environment", in Actas del XXIII Congreso Argentino de Ciencias de la Computación (CACIC 2017), 2017, pp. 1133-1142.
[26] De Giusti, F. E. Frati, M. Sanchez, L. De Giusti, "LIDI Multi Robot Environment: Support software for concurrency learning in CS1", in Proceedings of the 2012 International Conference on Collaboration Technologies and Systems (CTS), 2012, pp. 294-298.

Información de Contacto de los Autores:

Laura C. De Giusti
50 y 120 s/n
La Plata
Argentina
ldgiusti@lidi.info.unlp.edu.ar
http://www.lidi.info.unlp.edu.ar

Victoria Sanz
50 y 120 s/n
La Plata
Argentina
vsanz@lidi.info.unlp.edu.ar
http://www.lidi.info.unlp.edu.ar

Armando De Giusti
50 y 120 s/n
La Plata
Argentina
degiusti@lidi.info.unlp.edu.ar
http://www.lidi.info.unlp.edu.ar

Laura C. De Giusti
Doctora en Ciencias Informáticas (Facultad de Informática – UNLP). Lic. en Informática (Facultad de Informática – UNLP). Profesor Asociado Dedicación Exclusiva en la Facultad de Informática, UNLP. Investigador Asociado a la CIC de la Prov. de Bs. As.

Victoria Sanz
Doctora en Ciencias Informáticas (Facultad de Informática – UNLP). Lic. en Informática (Facultad de Informática – UNLP). Profesor Adjunto Dedicación Exclusiva en la Facultad de Informática, UNLP. Investigador Asociado a la CIC de la Prov. de Bs. As.

Armando De Giusti
Especialista en Tecnología Informática aplicada en Educación (Facultad de Informática – UNLP). Ing. en Telecomunicaciones (Facultad de Ingeniería – UNLP). Calculista Científico (Facultad de Ciencias Exactas – UNLP). Profesor Titular Dedicación Exclusiva en la Facultad de Informática, UNLP. Investigador Principal de CONICET, Argentina.