Scrum como Herramienta Metodológica en el Entrenamiento Cooperativo de la Programación: De la Teoría a la práctica

Ángel R. Barberis , Lorena E. Del Moral Sachetti
Universidad Nacional de Salta. Facultad de Ciencias Exactas. Salta, Argentina.
barberis@cidia.unsa.edu.a, lorena_dms@cidia.unsa.edu.ar

Resumen

El proceso de enseñanza y aprendizaje de la programación de computadoras no es una tarea fácil e impacta fuertemente en las asignaturas que tienen entre sus objetivos llevar adelante la práctica de la programación en carreras informáticas. Las dificultades del proceso es un problema recurrente que se evidencia en los últimos años a pesar de los esfuerzos de las instituciones nacionales por mejorar la calidad educativa en carreras universitarias. En este último sentido, se han puesto en práctica numerosas soluciones que no resultaron efectivas. La falta de motivación en los alumnos y las dificultades en el desarrollo de habilidades que deben adquirir, provocan que el aprendizaje y la práctica de la programación sea una de las causas de deserción en los primeros años de las carreras informáticas. El presente trabajo expone una experiencia metodológica que resalta la enseñanza y práctica cooperativa en el marco de trabajo propuesto por Scrum, con el objeto de promover acciones cooperativas y colaborativas en equipo y la adquisición de habilidades propias del programador en el desarrollo ágil de software. Se muestran resultados experimentales sobre dos asignaturas: Programación Numérica y Cálculo Numérico de la carrera Licenciatura en Análisis de Sistemas de la Universidad Nacional de Salta.

Palabras claves: Enseñanza cooperativa, Scrum, programación ágil, prácticas ágil, rendimientos.

1. Introducción

En la actualidad, las carreras relaciona-das con las Ciencias Informáticas o de la Computación, se ven fuertemente impactadas por la alta tasa de deserción de estudiantes en asignaturas relacionadas con la programación, desde aquellas que introducen los primeros conceptos hasta aquellas que tienen entre sus objetivos llevar adelante la práctica de la programación de computadores. En un artículo de Tecnología del diario La Nación de 2013 [1], expresa que el 80% de los estudiantes de carreras informáticas abandonan sus estudios durante los primeros años, según un reporte del Ministerio de Educación de la Nación. Este hecho, no solo ocurre en Argentina, sino también en otras partes del mundo [2-4].

El arte de la programación es una tarea compleja y difícil de abordar académicamente [2, 5-7]. La complejidad del Proceso Educativo de la Programación radica en que éste demanda la interacción de habilidades tanto del profesor como la de los alumnos y exige la garantía de que el educador propicie un ambiente cooperativo para desarrollar en el educando otras habilidades como las psico-cognitivas necesarias [8-10] para que pueda abordar problemas multidisciplinarios en carreras informáticas.

En el reto de proponer una estrategia metodológica que aborde académicamente las prácticas de la programación es necesario tener en cuenta algunos valores del alumno en socie-dad, por ejemplo, el ayudar a aprender a otro, compartir ideas y recursos, y planificar coope-rativamente qué y cómo estudiar. Las caracte-rísticas emocionales del programador juegan un papel decisivo a la hora de desarrollar problemas complejos que demandan tiempo y esfuerzo. Mantener en

equilibro las variables de tipo psicológico y hacer del estudiante un participante activo en la adquisición del cono-cimiento resultan determinante al momento de introducir innovaciones académicas por la constante adaptación al cambio de currícula y la modificación dinámica del grupo social en el que se encuentra inmerso. Por lo tanto, la estrategia metodológica del Proceso Educativo de la Programación debe fomentar las habilida-des sociales y de comunicación en los estu-diantes, haciendo del hábito de ayudar, com-partir y cooperar, una norma inexcusable en el aula. En este sentido, es que varios investiga-dores [11-13] visualizan al desarrollo de soft-ware como una actividad cooperativa, en don-de la principal característica es el trabajo en equipo. El aporte principal de nuestro trabajo, es el compartir una experiencia metodológica que usa a Scrum como herramienta en un marco cooperativo que ayudó a subir el índice de rendimiento, y reducir la tasa de abandono en las asignaturas Programación Numérica (PN) y Cálculo Numérico (CN) de carreras Informáticas y Matemáticas de la Facultad de Ciencias Exactas de la Universidad Nacional de Salta (UNSa).

2 El Aprendizaje Cooperativo

El aprendizaje cooperativo [14] consiste en trabajar juntos para alcanzar objetivos comunes, maximizando el propio aprendizaje y el de los demás. Son necesarios cinco elementos fundamentales para que los grupos de trabajo cooperativos funcionen correcta-mente. La *interdependencia positiva* supone que cada integrante del grupo tiene la conciencia de que su esfuerzo lo beneficia y también a los demás. Sin este elemento, no hay cooperación; Los miembros del grupo asumen una *responsabilidad individual y grupal* me-diante la cual cada uno deberá cumplir con la tarea que le toca; y gracias a una *interacción estimuladora* los miembros comparten recur-sos, conocimientos, se motivan por pequeños logros y se alientan en los fracasos, buscando en todo momento el éxito en los demás. El trabajo cooperativo requiere que los alumnos aprendan *prácticas interpersonales y grupales* necesarias para formar parte de un grupo, ya que deben saber cómo comunicarse, como dominar las situaciones que se les presentan, tomar decisiones, manejar conflictos, entre otros. Por último, el equipo deberá realizar una *evaluación* para saber en qué medida han podido alcanzar sus objetivos, analizando los aspectos en que fallaron y en el que triunfaron, y reconociendo las acciones positivas y negativas de cada miembro del equipo. De esta manera, se podrán tomar acciones estimula-doras o correctivas, según corresponda.

El aprendizaje cooperativo incrementa la motivación y la participación gracias a la interacción entre profesores y alumnos; posibi-litando un intercambio continúo de ideas, el desarrollo de habilidades comunicativas y so-ciales y la superación de actitudes negativas. Los estudiantes al sentirse apoyados y en confianza, son capaces de consolidar su propio estilo de aprendizaje [15]. Esta metodología, encuentra sustento teórico en las investiga-ciones de Piaget, Vygotsky y en las teorías de aprendizaje del constructivismo social [16, 17].

Piaget destaca la importancia de la interacción entre los estudiantes y la resolución de problemas que favorece el desarrollo men-tal, colocando en primer plano las destrezas de los alumnos, al mismo tiempo que ejercitan la capacidad para resolver problemas.

Las teorías constructivistas enfatizan la importancia del papel activo de quien aprende, de sus motivaciones, y de su capacidad de construir redes de significados entretejiendo los saberes previos con los contenidos nuevos, y todos ellos en base a la continua interacción con su entorno. Por su parte para Vigotsky [18], el aprendizaje es un evento interpersonal de carácter dialéctico, que depende de las características individuales, de las del contexto (profesores y/o compañeros) y de las relaciones entre estos.

3. Marco Colaborativo de Scrum

Scrum es un marco de trabajo que permite encontrar prácticas emergentes en dominios complejos, como la gestión de proyectos de innovación [19]. No se trata de un proceso completo, ni tampoco es una meto-dología. En lugar de proporcionar una descrip-ción completa y detallada de cómo deben realizarse las tareas de un proyecto, genera un contexto relacional, interactivo y cooperativo, de inspección y adaptación constante para que los involucrados vayan creando su proceso de trabajo. Esto ocurre debido a que no existen ni mejores ni buenas prácticas en un contexto complejo. Es el equipo de trabajo quien encon-trará la mejor manera de resolver sus proble-máticas. Básicamente, Scrum como propuesta de trabajo ágil es [20]: un modo de desarrollo de carácter adaptable; orientado a las personas antes que a los procesos; y emplea desarrollo ágil iterativo e incremental.

El equipo de desarrollo se encuentra apoyado en tres roles básicos [19-21]: el Scrum Master, el Product Owner y el Equipo de Desarrollo Scrum. El Scrum Master es quien vela por la utilización de Scrum, la remoción de impedimentos y asiste al equipo a que logre su mayor nivel de performance posible. Puede ser considerado como un coach o un facilitador encargado de acompañar al equipo de desarrollo. El Product Owner es quien representa al negocio y es quien conoce los requerimientos del cliente y usuarios finales. Tiene la responsabilidad de conducir al equipo de desarrollo hacia el producto adecuado. Finalmente el Equipo de Desarrollo Scrum, se trata de un grupo de personas que forman un equipo multidisciplinar que cubre todas las habilidades necesarias para generar el resultado. Se auto-gestiona y auto-organiza, y dispone de atribuciones suficientes en la organización para tomar decisiones sobre cómo realizar su trabajo.

El progreso de los proyectos que utilizan Scrum se realiza y verifica a través de una serie de iteraciones llamadas Sprints [19-21]. Estos Sprints tienen una duración fija, pre-establecida de no más de un mes. Al comienzo de cada Sprint el equipo de desarrollo realiza un compromiso de entrega de una serie de funcionalidades o características del producto en cuestión. Al finalizar el Sprint se espera que las características comprometidas estén terminadas, lo que implica su análisis, diseño, desarrollo, prueba e integración al producto. En ese momento es cuando se realiza una reunión de revisión del producto construido durante el Sprint, donde el equipo de desarrollo muestra lo construido al Product Owner y a cualquier otra persona interesada. La retroalimentación obtenida en esta reunión puede ser incluida entre las funcionalidades a construir en futuros Sprints.

3.1. Principios de Scrum

Scrum es el modelo más utilizado en el mundo de las Metodologías Ágiles, por su sencillez y trabajo colaborativo. Este marco de trabajo propone los siguientes principios básicos a ser valorados [19-21]:

a. **Individuos e interacciones por sobre procesos y herramientas**. Scrum se apoya en la confianza hacia las personas, sus interacciones y los equipos. Los equipos identifican lo que hay que hacer y toman la responsabilidad de hacerlo, removiendo todos los impedi-mentos que obstaculicen su labor y estén a su alcance.

b. **Software funcionando por sobre documentación exhaustiva**. Scrum requiere que al final de cada Sprint se entregue un producto funcionando. La documentación es entendida, en Scrum, como un producto intermedio sin valor de negocio. Los equipos pueden documentar tanto como crean necesario, pero ninguno de estos documentos puede ser considerado como el resultado de un Sprint. El progreso del proyecto se mide en

base al software funcionando que se entrega iterativamente.
c. **Colaboración con el cliente por sobre la negociación de contratos**. El Product Owner es el responsable de la relación que existe con los usuarios finales. El Product Owner es parte del Equipo Scrum y trabaja colaborativamente con el resto de los individuos dentro del equipo para asegurarse que el producto construido tenga la mayor cantidad posible de valor al final de cada iteración.
d. **Respuesta al cambio por sobre el seguimiento de un plan**. Scrum, por diseño, se ase-gura que todo el mundo dentro de un equipo tenga toda la información necesaria para poder tomar decisiones coherentes y cruciales sobre el proyecto en cualquier momento. El progreso es medido al final de cada Sprint mediante software funcionando y la lista de caracterísi-cas pendientes que está visible continuamente y para todos los miembros del equipo. Esto permite que el alcance del proyecto cambie constantemente en función de la retroalimen-tación provista por el Product Owner. Fomen-tar el cambio es una ventaja competitiva.

3.2. Valores de Scrum

Además de los 4 principios mencionados en la sección anterior, Scrum se construye sobre la base de 5 pilares exclusivos, siendo estos los valores y las capacidades humanas [19]:
a. **Foco**. Los Equipos Scrum se enfocan en un conjunto acotado de características por vez. Esto permite que al final de cada Sprint se entregue un producto de alta calidad y, adicionalmente, se reduce el tiempo de entrega del producto concluido al usuario final.
b. **Coraje**. El trabajo en equipos, permite que los integrantes puedan apoyarse entre compañeros, y así tener el coraje de asumir compromisos desafiantes que les permitan crecer como profesionales y como grupo.
c. **Apertura**. Los Equipos Scrum privilegian la transparencia y la discusión abierta de los problemas. No se promueven los conflictos. La sinceridad se agradece y la información que se requiera está disponible para todos.
d. **Compromiso**. Los Equipos Scrum tienen mayor control sobre sus actividades, por eso se espera de su parte el compromiso profesional para el logro del éxito.
e. **Respeto**. Debido a que los miembros trabajan de forma conjunta, compartiendo éxitos y fracasos, se fomenta el respeto mutuo y la cordialidad.

4. De la Teoría a la Práctica

Durante los últimos años hemos identificado problemas intrínsecos a las asignaturas: Programación Numérica y Cálculo Numérico, analizando sus causas y evaluando la metodología docente empleada. Con el obje-tivo de solucionar estos problemas, en el año 2014 se ha puesto en práctica una nueva metodología docente que propone un marco cooperativo, basada principalmente en el uso de Scrum como herramienta para el entrenamiento de la programación. En el presente trabajo se expone la metodología, herramienta y dinámica usada. Al final se realiza un análisis y evaluación de los resultados obtenidos.

4.1 El Anhelo de un Ambiente Cooperativo

En la era de la información, no sólo importa la disponibilidad de la información, el conocimiento y los medios para comunicarlas, sino también, el modo en que ellos puedan ser aplicados en prácticas reales. El desarrollo de habilidades propias de un programador (automotivación, dedicación, autonomía, superación, etc) dota al alumno de capacidades multifacética para enfrentar a problemas interdisciplinario de diferentes grados de dificultad, que le supone un reto y un desafío que los incita a su resolución, logrando así, la expe-riencia de un buen programador. Las estimula-ciones cognitivas inherentes a la resolución de problemas, provocan un aprendizaje por descu-brimiento [18], que favorece el desarrollo mental, colocan en primer plano las destrezas de investigación,

los entrena en la generación de soluciones, y finalmente, doblegan las capa-cidades de programación de computadores. Es por ello que se enfatiza la importancia de la interacción social en la construcción de saberes en los estudiantes. Por lo que vale la pena promover entornos educativos cooperativos donde los alumnos tengan la oportunidad de crear conocimientos significativos gracias a la interacción con los demás. Particularmente en el funcionamiento académico de nuestra carre-ra de Informática, se pueden observar diferen-tes situaciones, actitudes y posturas frente al uso de estrategias didácticas para una práctica adecuada de la programación. Los factores más importantes observados es el empleo de estra-tegias colaborativas en el proceso educacional de la programación, en lugar de una cooperati-va. Un buen análisis de las diferencias concep-tuales de ambos términos se pueden ver en [22, 23]. Básicamente, se podría decir que antes de establecer *colaboraciones* (con personas que ya saben) es preferible crear *cooperaciones* (personas que aprenden entre ellas). En un ambiente de cooperación la colaboración es un ingrediente más. Por lo tanto, un ambiente cooperativo es el más idóneo para mejorar el Proceso Educativo en la Programación de computadores. De ésta manera, se reducirían los impactos negativos en aquellas asignaturas que entre, otros objetivos, proponen la práctica de la programación. Teniendo en cuenta la realidad observada, y la disponibilidad de un marco cooperativo que propicia un desarrollo ágil de software, resulta necesario crear una estrategia metodológica con el objetivo de mejorar la calidad educativa en la programa-ción, teniendo en cuenta valores humanos, que permitan a los alumnos convertirse en protago-nistas activos en un marco social en el que se integran al grupo-clase a aquellos estudiantes socialmente aislados o tímidos.

4.2 La Experiencia del Entrenamiento

Desde que se iniciaron las actividades de acreditación de la carrera LAS en la UNSa, se puso especial énfasis en el proceso educativo que se empleaba en las asignaturas de PN y CN, con el objeto de indagar la calidad de la enseñanza y determinar los índices de rendi-mientos y deserción en las materias objetos del estudio.

La asignatura de PN corresponde al 2do semestre de 2do año de LAS, mientras que CN, se dicta para las carreras de Licenciatura y Profesorado en Matemáticas (LyPM). En am-bas, se imparten contenidos del Análisis o Métodos Numéricos. La diferencia entre una y otra radica en el enfoque de las prácticas reali-zadas. La asignatura PN tiene una práctica fuertemente apoyada en el entrenamiento de la programación, mientras que CN para LyPM, tiene una práctica orientada hacia el desarrollo algorítmico con herramientas adecuadas para tales. Debido a esto, las prácticas son diferen-ciadas en tiempo y espacio. Las premisas del marco cooperativo son las mismas para ambos cursos, aunque el proceso organizativo es diferente.

Para poder implementar adecuadamen-te la estrategia cooperativa con los alumnos de PN, en el entrenamiento de la programación, se siguen los siguientes pasos:

Fase organizativa

1. Impartir una clase introductoria sobre el marco de trabajo Scrum. La clase no tiene porqué ser exhaustiva. Se expone claramente el marco colaborativo, los eventos y artefactos a utilizar.

2. Identificación de programadores entusiastas. El objetivo es distribuirlos tanto como sea posible en los grupos de trabajos que se forma-sen. Los programadores entusiastas son más activos y participativos, ya que demuestran mayores habilidades que sus compañeros. Por lo que la interacción con el resto del equipo logra elevar el nivel de los novatos y ayuda a consolidar la experiencia del entusiasta. De esta forma se pone en práctica el Tutoreo de Pares [24].

3. Organización en grupos de trabajo de entre 7 y 10 integrantes. No existe una cantidad ideal de miembros. El trabajo cooperativo del grupo debe ser equilibrado, participativo, multifacético y variado. Las tareas asignadas nunca deberán ser una sobrecarga para los miembros. Por lo tanto, teniendo en cuenta el volumen de trabajo que se le encomienda en cada sprint, se adopta una dimensión entre 7 y 10 individuos.

4. Entrenamiento a la socialización (iniciación a las *prácticas interpersonales y grupales*). En el primer sprint, el profesor de práctica desarrolla una dinámica de grupo con el objeto de provocar una interlocución amena de los miembros, procurando, especialmente, la parti-cipación de los individuos auto-marginados o tímidos.

5. Dar lineamientos de los artefactos de Scrum a utilizar. El profesor de la práctica en su rol de Scrum Master crea un ambiente dinámico de interacción con el objeto de que los alumnos fijen claramente los conceptos relacionados a los artefactos:

- Pila del producto: (Product Backlog) lista de requisitos de usuario, que a partir de la visión inicial del producto crece y evoluciona durante el desarrollo. Está formada por la lista de funcionalidades que el cliente desea obtener, ordenadas por la prioridad que él mismo le otorga a cada una.
- Pila del Sprint: (Sprint Backlog) lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto. Generalmente, refleja los requisitos vistos des-de el punto de vista del equipo de desarrollo. Está formada por la lista de tareas en las que se descomponen las funcionalidades especifica-das por el cliente.
- Incremento: resultado de cada sprint. El resultado consiste en una versión de software previsto, probado y en funcionamiento. Adi-cionalmente, se entrega un informe del desarrollo, en el que se describe la pila del producto y la del sprint, como así también, los casos de pruebas unitarias, funcionales y de integración.

Roles

a. *El profesor responsable* de la cátedra oficia de Product Owner, y es quién brinda los requerimientos de producto entregable de cada Sprint.
b. *Los docentes de prácticas* ejercen el rol del Scrum Master. Son quienes están en permanente interacción con los grupos de trabajos. Durante cada Sprint en consenso, se determina un líder de equipo para el Sprint siguiente. El líder de equipo es diferente para cada Sprint y su designación debe rotar entre todos los integrantes del grupo de trabajo, garantizando la participación total de sus miembros en este rol.
c. *Los alumnos divididos en grupos de trabajos* ocupan el rol del Equipo de desarrollo Scrum.

Dinámica

La transferencia de conocimiento, se realiza a través de dos clases teóricas y dos de prácticas semanales. En las clases teóricas, el profesor cumple un rol fundamental del proceso interactivo que sirve de soporte a la construcción del conocimiento. Expone un tema, indaga a la clase y construye el concepto con ideas guiadas, favoreciendo el aprendizaje por descubrimiento [18]. Una vez que se ha formulado el concepto, se reafirma el conocimiento con un ejemplo, procurando sea éste un caso de la vida real. Luego indaga nuevamente a la clase para que se propongan nuevos ejemplos frutos del razonamiento del alumnado. Al final de la clase teórica el alumnado llena un formulario de encuesta con carácter anónimo. El formulario consta de tres columnas. Cada columna se etiqueta con los símbolos ☺☹😐, donde el 1ro indica que le gustó la clase, el 2do que no le gustó y el 3ro es para indicar una sugerencia. El alumno puede marcar sólo una de las dos primeras pero no ambas. Este formulario también se llena en cada comisión de trabajos prácticos.

El programa temático consta de diez unidades y se imparte una guía de trabajos prácticos por cada unidad. La guía de trabajos prácticos consta de dos partes: 1) una propuesta de ejercicios de resolución numérica que el alumno debe resolver de manera individual; 2) la especificación del incremento de software requerido al término de la guía.

El desarrollo de la guía se concreta con un desarrollo individual de los ejercicios del análisis numérico, y con el trabajo cooperativo de un desarrollo ágil de software. Cada Sprint tiene una duración de una semana. Al inicio de cada guía, el equipo de desarrollo deberá trabajar con la Pila del Producto y desarrollar la Pila de Sprint, y trabajarán en función de los tiempos asignados a cada guía de práctica. Al final de cada sprint, el equipo de desarrollo se reúne con el Scrum Master (Profesor de Práctica) y realizan una retrospectiva. Ésta consiste en la realización de una reunión al final del sprint de revisión, en donde el equipo reflexiona sobre la forma en que desarrollaron sus trabajos, y se exponen los acontecimientos que sucedieron durante el sprint (*evaluación del desempeño grupal*). La retrospectiva no debe durar más de 10 minutos. Básicamente, se centra en el proceso del CÓMO se realizaron las labores asignadas, se identifican fortalezas y debilidades, y se planifica acciones de mejora para el siguiente sprint. Cada miembro del equipo responderá a tres preguntas básicas de respuestas precisas, concisas y claras. ¿Qué tenía que hacer? ¿Qué hice? y ¿Qué me falta hacer? Las respuestas dadas posibilitan que el equipo realice un autoanálisis del estado de compromiso de los miembros. No se acepta el incumplimiento de un sprint. La falta de compromiso de algunos de los miembros recaerá en un esfuerzo adicional repartido sobre las labores de los demás integrantes del equipo. Esto favorece la acción cooperativa, solidaria, colaborativa y humana con sus pares.

Cada dos sprint, el Product Owner (Profesor Responsable de cátedra) participará de las reuniones del equipo de desarrollo en la revisión del sprint, y también, de la retrospección. El incremento de software de cada sprint constituirá una evolución de lo realizado en el sprint anterior. Por lo que al final del semestre, cada equipo tendrá un software completo con todos los algoritmos de la currícula de la cátedra implementados.

En lo que respecta a CN, el marco cooperativo se implementa de la misma forma. No se exige el informe de desarrollo del software, pero sí un informe de conclusiones respecto de la guía práctica realizada.

4.3 Análisis de Resultados

La estrategia metodológica se implementó a partir del año 2014 en las clases prácticas de las asignaturas PN y CN. El objetivo primordial que se persigue es el reducir la tasa de deserción o abandono en las asignaturas, y elevar el rendimiento académico, bajo el principio de mejorar la calidad educativa. Durante el primer año, se implementó un marco colaborativo no cooperativo muy exigente como prueba piloto para analizar el impacto psico-cognitivo de los alumnos. La idea fue trabajar con los principios de Scrum, sin exigirles el aspecto cooperativo, con la exigencia de la presentación de todas las guías de trabajos prácticos resueltas en computa-doras, el incremento de software y el informe de desarrollo. Desde el punto de vista de la cátedra, los índices porcentuales de alumnos que regularizaron, que quedaron libres y el de abandono, mejoraron significativamente respecto de los índices de años anteriores. Desde el punto de vista de los alumnos, el grado de insatisfacción fue muy alto, como consecuencia de las exigencias en las resoluciones prácticas. Todas las disconformidades de los alumnos fueron expuestas en la última clase del semestre de ese año. Luego se analizaron las encuestas anónimas, los trabajos prácticos, los informes del desarrollo de software y los programas implementados. Se detectaron otros inconvenientes que se relacionaban con lo psicológico. Aproximadamente el 73,5 % de los alumnos entraban en la clasificación de

tímidos/introvertidos, ya que, en vez de sugerencias en la columna correspondiente, se visualizaba preguntas y consultas que podrían haberlas realizado directamente al profesor en clase. A pesar de las reiteradas aclaraciones respecto de cómo debían llenar las encuestas y las permanentes indagaciones del profesor, los alumnos siguieron con la misma postura de realizar consultas anónimas.

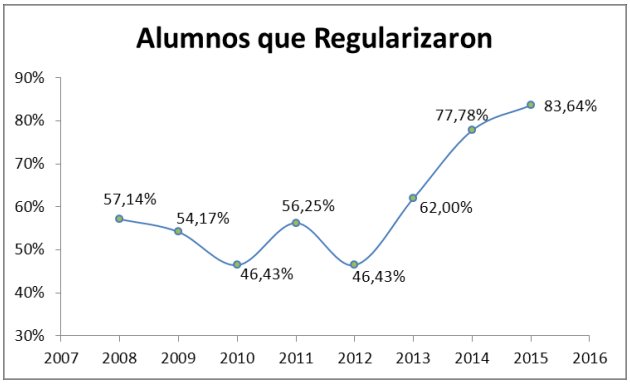

Fig. 1: Curva de alumnos que regularizaron las asignaturas entre los años 2008 y 2015.

En el año 2015, se tuvieron en cuenta los aspectos psicológicos y cognitivos de los alumnos. Se identificaron las causas de la desmotivación por las prácticas de programación y la falta de fijación de conceptos teóricos.

Como consecuencia, la cátedra propuso nuevos objetivos, y uno de ellos tenía que ver con las maneras de promover las prácticas de la programación de lenguajes.

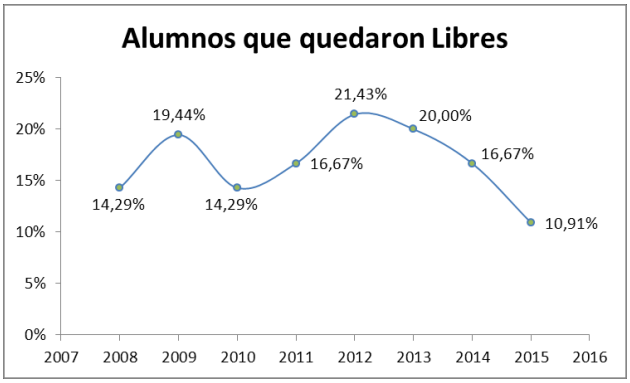

Fig. 2: Curva de alumnos que no regularizaron las asignaturas entre los años 2008 y 2015.

El cambio radical que marcó "*un antes y un después*" fue mirar a las prácticas de programación como un "entrenamiento" en el desarrollo de software a nivel de los alumnos de segundo año de la carrera.

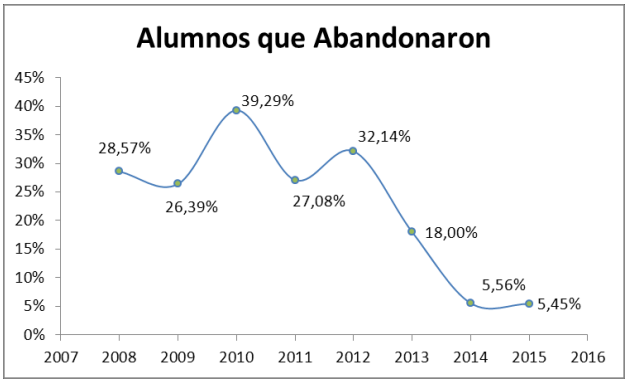

Fig. 3: Curva de alumnos que abandonaron las asignaturas entre los años 2008 y 2015.

Para ello, se necesitaba incorporar más elementos del marco de trabajo de Scrum en las prácticas de programación. Dar mayor soporte al trabajo cooperativo, a nivel *alumno-alumno*, *alumno-profesor*, *profesor-profesor*. El cambio en la estrategia educativa en las asignaturas implicó un mayor compromiso de los docentes, sobre todo en el modo de relacionarse con los alumnos. Tanto las clases teóricas como prácticas debían ser más interactivas entre *profesor-alumno* y *alumno-alumno* para favorecer una reestructuración cognitiva.

4.4 Conclusiones y Acciones futuras

Desde que surgió la iniciativa de mejorar la calidad educativa en asignaturas que tienen entre sus objetivos llevar adelante las prácticas de la programación, la estrategia metodológica descripta ha brindado resultados muy alentadores. Por un lado se logró mejorar la calidad cognitiva del alumnado, y por el otro, la adopción de nuevas experiencias y entrenamientos en la programación ágil de software. Al mismo tiempo, se contribuyó con la disminución de la tasa de deserción e incrementar los índices de rendimientos. Ver figuras de 1 a 3. El cambio de escenario, en el que se pone al alumno como principal protagonista en la interacción cooperativa en la educación, promueve un ambiente en el que se

desarrollan nuevas habilidades, no solo como programador, sino también como ser humano. Este ambiente propicia mayor diálogo entre los protagonistas, confianza, actitud activa y participativa, acciones que permite a la cátedra detectar rápidamente problemas cognitivos e instrumentar acciones correctivas. La retroali-mentación constante entre *profesor-profesor* y *profesor-alumno* ha sido un poco, la clave del éxito. Los resultados nos alentaron a abrir una línea de investigación sobre el entrenamiento en el desarrollo ágil de software en asignaturas que realizan prácticas de programación. En el marco de esta investigación, se están evalúan-do la factibilidad de nuevos eventos y artefac-tos de Scrum, para ir incorporando año a año nuevas relaciones entre la práctica ágil de la programación y los objetivos académicos de las asignaturas que no enseñan lenguajes de programación.

5. Referencias

[**1**] Perazo C. (**2013**) Tecnología. *Reporte del Ministerio de Educación*. 31/03/2016. Diario La Nación, Portal de Internet. http://www.lanacion.com.ar/1632045-el-80-de-los-estudiantes-de-carreras-informatica

[**2**] Lahtinen E., Ala-Mutka K. and Järvinen H.-M. (**2005**). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*. Vol. 37 (3):14-18.

[**3**] Costelloe E. (**2004**) Teaching Programming The State of the Art. *CRITE*. Technical Report. DC, Inst. of Tech. Tallaght, Dublin, Ireland.

[**4**] Bati T. B., Gelderblom H. and van Biljion J. (**2015**) Blended learning of programming in large classes: a reflection of students' experience from an Ethiopian University. *Transform 2015 Research Colloquium*. Educational Technology Inquiry Lab., University of Cape Town.

[**5**] Villalobos J. A., Casallas R. and Vela M. (**2007**) Una Solución Moderna e Integral al Problema de Enseñar Programación. *XXVII Reunión Nacional de FI y VI EIIEI*. Octubre de 2007, Cartagena de Indias, Colombia

[**6**] Sarpong K. A.-m., Arthur J. K. and Owusu Amoako P. Y. (**2013**). Causes of Failure of Students in Computer Programming Courses: The Teacher - Learner Perspective. *IJCA*. Vol. 77 (12):27-32.

[**7**] Rahmat M., Shahrani S., Latih R., Yatim N. F. M., Zainal N. F. A. and Rahman R. A. (**2012**). Major Problems in Basic Programming that Influence Student Performance. *Procedia - Social and Behavioral Sciences*. Vol. 59. pp. 287-296.

[**8**] Milne I. and Rowe G. (**2002**). Difficulties in Learning and Teaching Programming -- Views of Students and Tutors. *Education and Information Technologies*. Vol. 7 (1):55-66.

[**9**] Mancy R. and Reid N. (**2004**) Aspects of Cognitive Style and Programming. *Proceedings of 16th workshop of the PPIG*. Institute of Technology. Carlow, Ireland.

[**10**] Bennedsen J. and Caspersen M. E. (**2005**). Revealing the programming process. *ACM SIGCSE Bulletin*. Vol. 37 (1):186-190.

[**11**] Lovos E., González A. H., Mouján I., Bertone R. A. and Madoz M. C. (**2012**) Estrategias de enseñanza colaborativa para un curso de programación de primer año de la Licenciatura en Sistemas. *XVIII CACIC*. RedUNCI, UNS, Bahía Blanca.

[**12**] González A. H. and Madoz M. C. (**2014**) Desarrollo de actividades colaborativas en un curso inicial de programación de computadoras. *XX CACIC*. RedUNCI, UNLAM.

[**13**] Rosanigo Z. B. and Paur A. B. (**2006**) Estrategias para la enseñanza de Algorítmica

y Programación. *I Congreso de TE & ET*. RedUNCI, UNLP.

[**14**] Johnson D. W., Johnson R. T. and Holubec E. J. (**1999**) *El aprendizaje cooperativo en el aula*. Paidos Educador. Paidos, Argentina.

[**15**] Calzadilla M. E. (**2002**) Aprendizaje colaborativo y tecnologías de la información y la comunicación. Revista. *Iberoamericana de Educación*. OEI, Educación, Ciencia y Cultura.

[**16**] Alfageme Gonzalez M. B. (**2001**) Antecedentes de las ideas pedagógicas subyacentes en el aprendizaje cooperativo. Revista. *Anales de Pedagogía*. N° 19. pp 149-168; año 2001.

[**17**] Crook C. (**1998**) *Ordenadores y aprendizaje colaborativo*. Ministerio de Educación y Cultura. (1º Ed. np. 316) Ediciones Morata.

[**18**] Barrón Ruiz A. (**1993**). Aprendizaje por descubrimiento: principios y aplicaciones inadecuadas. *Enseñanza de las Ciencias: Investigación y experiencias didácticas*. Vol. 11 (1):3-11.

[**19**] Alaimo M. (**2013**) *Proyectos Ágiles con Scrum*. Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos. (1º Ed.) (np. 126) Kleer Agile Coaching & Training, Buenos Aires, Argentina.

[**20**] Palacio J. (**2007**) Flexibilidad con Scrum. *Principios de diseño e implantación de campos de Scrum*. (1º Ed.) np. 190. Navegápolis.com.

[**21**] Palacio J. (**2015**) Gestión de proyectos Scrum Manager. *(Scrum Manager I y II)*. (Versión 2.5.1º Ed.) np. 98. Scrum Manager®.

[**22**] Panitz T. (**1999**) *Collaborative versus cooperative learning: a comparison of the two concepts which will help us understand the underlying nature of interactive learning*.Artículo. Disponible en http://home.capecod.net/~tpanitz/tedsarticles/coopdefinition.htm. Accedido: 28/03/2016

[**23**] McInnerney J. M. and Robert T. S. (**2004**) Collaborative or cooperative learning? . *Online collaborative learning: Theory and practice*. pp. 203-214. Inform. Science Publishing, Hershey.

[**24**] Tudge J. (**1993**) Vigotsky, la zona de desarrollo próximo y la colaboración entre pares: connotaciones para la práctica del aula. en Moll, Luis C. (Comp) Vigotsky y la educación. Connotaciones y aplicaciones de la psicología sociohistórica en la educación. Aique, Argentina.