ReTEC

Revista Electrónica de Tecnología, Educación y Ciencia
ISSN: 2953-5654
http://retec.unsa.edu.ar
Universidad Nacional de Salta

Aprendizaje Significativo basado en el Entrenamiento de la Programación con Scrum

Angel R. Barberis , Lorena E. Del Moral Sachetti, Jorge Silvera

Universidad Nacional de Salta. Facultad de Ciencias Exactas
barberis@unsa.edu.ar, Lorena.dms.7@gmail.com, jsilvera@hotmail.com

Revista Electrónica de Tecnología, Educación y Ciencia,

Volumen 1, Número 3, pág. 115-125, jun, 2026. ISSN: 2953-5654

Disponible en http://retec.unsa.edu.ar/

Aprendizaje Significativo basado en el Entrenamiento de la Programación con Scrum

Angel R. Barberis, Lorena E. Del Moral Sachetti, Jorge Silvera

Universidad Nacional de Salta. Facultad de Ciencias Exactas
barberis@unsa.edu.ar, Lorena.dms.7@gmail.com, jsilvera@hotmail.com

Resumen. Aprender a programar es una tarea difícil y desafiante para los estudiantes de informática, lo que genera altas tasas de reprobación y abandono. Estas dificultades también afectan a otras materias relacionadas con la programación. Este trabajo presenta una experiencia basada en una revisión metodológica que permite a los alumnos entrenarse en el desarrollo ágil de software. Este enfoque supera la mera práctica de lenguajes de programación, promoviendo acciones cooperativas y colaborativas en equipo. Así, se facilita la adquisición de habilidades propias del programador profesional. Finalmente, se muestran los resultados experimentales de la aplicación de esta metodología en la asignatura Programación Numérica de la Licenciatura en Análisis de Sistemas de la Universidad Nacional de Salta.

Palabras claves*:* Enseñanza ágil, programación ágil, entrenamiento en la programación ágil.

1. Introducción

En la actualidad, las carreras relacionadas con las Ciencias Informáticas o de la Computación, se ven fuertemente impactadas por las altas tasas de deserción de estudiantes en asignaturas relacionadas con la programación de computadoras, desde aquellas que introducen los primeros conceptos [1] hasta aquellas que no la enseñan pero que tienen entre sus objetivos académicos poner en práctica del desarrollo de programas informáticos [2]. En los últimos 20 años se evidenciaron un alto porcentaje de deserción de alumnos en carreras informáticas, según reportes en diversos medios. La Cámara de empresas de Software y Servicios Informáticos (Cessi) en 2007 reportó que el 60 % de los alumnos de informática abandonaron la carrera en el primer año [3]; en 2013, la Secretaria de Políticas Universitarias del Ministerio de Educación de la Nación [4], reporta que el 80% de los estudiantes de carreras informáticas abandonan sus estudios durante los primeros años; y en 2019, la Licenciada en Sociología y especialista en Psicología Educacional Liliana Llamas, docente de la Universidad de Buenos Aires (UBA) [5], expresa que entre el 60 y 70 % de los inscriptos en una carrera universitaria abandonan sus estudios a mediado del primer año. Hoy en día, parece que la realidad no es diferente al de los años anteriores, ni al de otras partes del mundo, tal como lo presenta el artículo titulado "*Informática sigue siendo una de las carreras con más abandono*" publicado en julio de 2020 por el portal Webedia, que describe la realidad Española [6]. Este panorama, sumado a tantos otros artículos de investigación reportados en la literatura científica, muestra no sólo que los estudiantes experimentan una amplia gama de dificultades y deficiencias, sino también, que es una realidad recurrente tanto en Argentina como en otras partes del mundo [2, 7-9].

El arte de la programación de computadoras es una tarea compleja y difícil de abordar académicamente [2, 10-12]. La complejidad del Proceso Educativo de la Programación radica en que éste demanda la interacción de habilidades tanto del profesor como de los alumnos, y exige la garantía de que el educador propicie un ambiente cooperativo y colaborativo para desarrollar en el discente otras habilidades como las psico-cognitivas [13-15] y trabajo en equipo [16], entre otras, necesarias para el abordaje de problemas multidisciplinarios [17, 18]. Por lo tanto, es de suma importancia contar con una estrategia metodológica de enseñanza y

aprendizaje que propicie un ambiente de trabajo en grupo, en el que se pueda fomentar tanto, habilidades sociales como de comunicación, haciendo del hábito de ayudar, compartir y cooperar, una norma inexcusable en el aula. En este sentido, es que varios investigadores visualizan al desarrollo de software como una actividad cooperativa, en donde la principal característica es el trabajo en equipo [19, 20].

En la era del desarrollo tecnológico, no sólo importa la disponibilidad de la información, el conocimiento y los medios para comunicarla, sino también, el modo en que ellos puedan ser aplicados en prácticas reales. El desarrollo de habilidades inherentes a la programación de computadoras (creatividad [21], autoeficacia [22], resolución de problemas [23], razonamiento [24], trabajo en equipo [16], etc.) desarrolla en el alumno las capacidades multifacéticas que le permiten enfrentar problemas interdisciplinarios de diferentes grados de dificultad, que sumado a un buen entrenamiento, adquiere la experiencia de un buen programador. Las estimulaciones cognitivas inherentes a la resolución de problemas (actividades de exploración, análisis y búsqueda de soluciones) [23], estimulan un proceso de aprendizaje, que favorece el desarrollo mental, colocan en primer plano las destrezas de investigación, los entrena en la generación de soluciones, y con ello, los estudiantes se encaminan hacia el mayor desafío de doblegar las capacidades de programación de computadoras. La actividad de resolución de problemas en Programación necesita además de la habilidad técnica para sintetizar o resumir una solución [25, 26]. Esta habilidad junto al trabajo en equipo se puede desarrollar mediante la práctica constante o el entrenamiento en la programación.

El aporte principal de nuestro trabajo, es el compartir los resultados de la puesta en práctica de una metodología experimental basada en el desarrollo ágil de software que usa Scrum como herramienta en un marco cooperativo, centrado en el aprendizaje por proyecto, que propicia un ambiente en el que los alumnos se entrenan en la programación de aplicaciones computacionales. La metodología permitió a los alumnos convertirse en protagonistas activos en un marco social en el que se integran al grupo-clase a aquellos estudiantes socialmente aislados o tímidos, logrando así, mejorar el rendimiento académico y una reducción significativa de la tasa de abandono en la asignatura Programación Numérica de la carrera Licenciatura en Análisis de Sistemas, en la Universidad Nacional de Salta.

2. Marco de Referencia Conceptual

2.1. Aprendizaje Cooperativo

Uno de los ejes centrales de la metodología implementada fue fomentar un ambiente cooperativo en el que los estudiantes puedan desarrollar habilidades de trabajo en equipo, específicamente en el contexto de la programación de aplicaciones informáticas.

La cooperación es trabajar juntos para lograr objetivos compartidos. Dentro de las actividades cooperativas, los individuos buscan resultados que sean beneficiosos no solo para ellos mismos, sino también para todos los demás miembros del grupo. El aprendizaje cooperativo es el uso educativo de pequeños grupos en los que los estudiantes trabajan juntos para maximizar su propio aprendizaje y el de los demás [27].

El aprendizaje cooperativo aumenta la motivación y la participación gracias a la interacción entre docentes y estudiantes; permitiendo un intercambio continuo de ideas, el desarrollo de habilidades de comunicación y sociales, y la superación de actitudes negativas. Los estudiantes, al sentirse apoyados y seguros, son capaces de consolidar su propio estilo de aprendizaje [28].

En resumen, el trabajo cooperativo es una estrategia de gestión del aula que favorece la organización de los estudiantes en grupos heterogéneos para llevar a cabo tareas y actividades de aprendizaje. Esto implica agrupar a los estudiantes en pequeños equipos para promover el desarrollo de cada miembro.

2.2. Programación en Pares en un Entorno Cooperativo

Una técnica educativa que tiene elementos en común con el aprendizaje cooperativo es la programación en pares [29]. En esta forma de cooperación, dos programadores trabajan juntos en una computadora. En un momento dado, un miembro del equipo (el 'conductor') puede estar trabajando en la computadora: ya sea escribiendo un programa o modelando un diseño. El otro miembro (el 'navegante') puede estar observando activamente el trabajo del 'conductor', ayudando a resolver posibles errores, analizando soluciones alternativas, indagando sobre el conocimiento requerido, etc. Los roles de 'conductor' y 'navegante' se intercambian periódicamente entre los dos miembros del equipo. La programación en pares fue popularizada originalmente como parte de la metodología de desarrollo de software Programación Extrema [30]. Investigaciones en la literatura reportan que los programadores en pares producen código de mayor calidad en la mitad del tiempo en comparación con los programadores que trabajan solos [31]. También se ha encontrado que la técnica es efectiva para los estudiantes de programación, llevando a una mejora en el aprendizaje y la satisfacción del estudiante y a una reducción de la frustración en el desarrollo cognitivo [32].

El aprendizaje cooperativo utiliza métodos similares a la programación entre pares para ayudar a los estudiantes a aprender sobre programación y procesos de resolución de problemas. Sin embargo, en un entorno que promueve el aprendizaje cooperativo, se guía a los estudiantes a través de diferentes niveles de cooperación. Así, por ejemplo, en las etapas iniciales, el grupo completo puede hacer una lluvia de ideas para resolver un problema. En una etapa posterior, los estudiantes podrían trabajar en pares para resolver el problema y luego comparar sus soluciones con las desarrolladas por otra pareja del mismo grupo. Más tarde, otros ejercicios brindan a los estudiantes la oportunidad de trabajar en problemas por sí mismos, con la ayuda de otros miembros del grupo si es necesario. Con este enfoque incremental, se ofrecen incluso más ventajas que la programación estricta en pares. Al principio, todos en el grupo están aprendiendo a abordar una tarea de programación. Por lo tanto, es útil tener tantos puntos de vista diferentes como sea posible. A medida que desarrollan sus habilidades de programación y resolución de problemas, los estudiantes progresan para trabajar en pares. Finalmente, tienen la oportunidad de desarrollar confianza resolviendo individualmente (incluso con el apoyo del grupo).

2.3. Aprendizaje Cooperativo Basado en Problemas

Otras experiencias de aprendizaje cooperativo están diseñadas para separar y resaltar aspectos importantes de la programación y la resolución de problemas. Este es el caso del Aprendizaje Basado en Problemas (ABP), cuyo aprendizaje resulta del proceso de trabajar en la comprensión y resolución de problemas, donde el problema es un elemento importante en el proceso de aprendizaje. El marco teórico establece que las características del ABP son [33]: el aprendizaje está centrado en el alumno; el aprendizaje cooperativo ocurre en pequeños grupos de estudiantes; los profesores son facilitadores o guías; los problemas son la herramienta para el desarrollo de habilidades de resolución de problemas informáticos; la nueva información se adquiere mediante el aprendizaje autodirigido.

Así, el ABP marca una posición absolutamente diferente del aprendizaje basado en la enseñanza tradicional. En un entorno de ABP, los estudiantes reciben la descripción de un problema y ellos mismos identifican lo que necesitan saber, lo estudian y lo aplican para especificar la solución. Mientras que, en la enseñanza tradicional, los estudiantes reciben la descripción del problema, el profesor muestra lo que necesitan saber y los induce a determinar una solución. Los grupos cooperativos de resolución de problemas en un entorno de ABP suelen estar formados por dos a cuatro miembros.

El aprendizaje basado en problemas es muy adecuado para la ingeniería (como lo es para la medicina, donde se utiliza actualmente) porque ayuda a los estudiantes a desarrollar las habilidades y la confianza para explorar, analizar y especificar la solución adecuada para un problema dado. El proceso de construir modelos juntos en una interacción interpersonal cara a cara da como resultado un aprendizaje que es difícil de lograr de otra manera.

2.4. Marco Cooperativo Ágil

Promover un entorno cooperativo basado en un ABP no lo es todo, a menos que vaya acompañado de artefactos que faciliten su desarrollo. En este contexto, se decidió utilizar técnicas ágiles como Scrum para el desarrollo de aplicaciones de software, lo que permite a la cátedra de Programación de Métodos Numéricos formar a sus estudiantes en programación, más allá de una mera práctica de contenidos curriculares.

Scrum [34] es un marco de trabajo que permite fortalecer y consolidar las relaciones del equipo humano que interactúa cooperativamente en el desarrollo ágil de software. Propone un conjunto de prácticas y artefactos que posibilitan transformar un problema complejo en actividades simples de resolución inmediata y progresiva, generando al mismo tiempo un contexto relacional, interactivo y cooperativo de constante inspección y adaptación para que los involucrados puedan crear su propio estilo de trabajo. De esta manera, se crea un entorno en el que se proporciona al equipo los mecanismos necesarios para desarrollar buenas prácticas de trabajo en un contexto complejo.

En resumen, el marco de desarrollo Scrum busca enfocar al equipo en producir valor final de calidad, colaborando y cooperando en una comunicación interactiva para lograr mejoras continuas en el aprendizaje.

3. Estrategia Metodológica

Las características y principios del desarrollo ágil de Scrum proporcionan la dinámica central de la innovación pedagógica. Esto no solo brinda los mecanismos ideales para simular la realidad laboral en el aula, sino que también facilita la combinación de los mejores lineamientos de cada metodología activa utilizada.

A partir de la experiencia, se delinearon un conjunto de actividades, organizadas en etapas que pueden diferir o variar según el tipo de estudiantes, el enfoque de la asignatura y la cantidad de programación utilizada en las prácticas académicas.

Las etapas y momentos que conforman la propuesta se desarrollan según una secuencia temporal que no ocurre de manera lineal, sino por aproximaciones sucesivas. Las etapas son: Diagnóstico e indagación; Diseño y planificación; Implementación - Acción metodológica; y Evaluación - Reflexión.

3.1. Indagación y Diagnóstico

El docente realiza un análisis de la realidad educativa para desarrollar la propuesta pedagógica. Esto implica conocer en profundidad a los estudiantes para una adecuada implementación de las etapas siguientes. La actividad principal de esta etapa es la evaluación diagnóstica inicial para identificar el estado psico-cognitivo de los estudiantes. La evaluación se estructura en tres bloques. El primer bloque incluye ejercicios conceptuales necesarios para abordar los contenidos curriculares de la asignatura.

El segundo bloque está orientado a problemas numéricos simples. El tercer bloque está dirigido a identificar programadores entusiastas o estudiantes con habilidades particulares relacionadas con la programación. La información obtenida en este bloque contribuye a la formación de mejores criterios para la adaptación pedagógica tanto en el proceso de enseñanza-aprendizaje como en las prácticas de consolidación del conocimiento.

3.2. Diseño y planificación

Una vez que el docente define la realidad educativa en la que se desarrollará el proceso de aprendizaje y consolidación del conocimiento, se procede al diseño de la propuesta metodológica contextualizada. El desarrollo de una práctica académica basada en la formación ágil en programación requiere una adecuada planificación por parte de los docentes, basada en los elementos que conforman el proceso de formación académica (objetivos, contenidos, tiempos, instrumentos de evaluación, entre otros), es decir, el currículo. Para ello, es necesario contemplar los propósitos didácticos (objetivos y competencias) y los contenidos. Además, se debe dedicar un mayor esfuerzo al diseño de las estrategias metodológicas (agrupamientos, actividades, tiempos, materiales) y al sistema de evaluación. Cabe señalar que la adaptación de los contenidos a los estudiantes de un ciclo se desarrolla en paralelo y en sintonía con la metodología y la evaluación.

A su vez, los contenidos, los aspectos metodológicos y la modalidad evaluativa dependen de los objetivos y competencias a alcanzar. Entre las tareas que se desarrollan en esta etapa, destaca la formación de grupos de estudio, el diseño de actividades para la consolidación del conocimiento y la planificación temporal.

3.3. Implementación - Acción metodológica

La temporalización es fundamental en el diseño de las prácticas y la formación en programación ágil, pero no es fácil determinar a priori cuánto tiempo requieren las diferentes actividades de aprendizaje individual. La etapa de implementación es la que demanda más tiempo, porque se pone en marcha el diseño previamente definido. Los docentes cumplen los roles de Product Owner y Scrum Master, los cuales son vitales para la formación en programación dentro de un entorno Scrum. La acción pedagógica de la propuesta innovadora tendrá éxito en la medida en que los estudiantes sean instruidos en roles particulares. El programador entusiasta coordina las actividades cooperativas del grupo y el docente de la práctica instruye al entusiasta para que realice acciones equilibradas, participativas, multifacéticas y variadas, sin implicar una disminución de su rendimiento académico. Esto incluye la tutoría entre pares, las dinámicas de trabajo, la autoorganización y la decisión autónoma. A partir de ello, se desarrolla una simulación de la realidad laboral en el aula.

3.4. Evaluación - Reflexión

El diseño de la evaluación está estrechamente relacionado con la metodología de enseñanza utilizada. Dependiendo de cómo se considere la evaluación al diseñar el proceso, esta puede percibirse como un juicio o como una ocasión para aprender. En la propuesta metodológica, la evaluación se concibe como un proceso sistémico. En ella, el docente revisa el modelo pedagógico que sustenta la actividad formativa y selecciona las estrategias y herramientas que permitan verificar la evolución y los progresos reales alcanzados por los estudiantes. Deben coexistir evaluaciones grupales en su conjunto, junto con evaluaciones individuales. Esto permitirá que la evaluación sea coherente en toda su dimensión. La primera decisión que deben tomar los docentes es la importancia que le otorgarán a cada uno de estos dos tipos de evaluación. Empíricamente, se sabe que una mejor opción es que las actividades propuestas sean evaluadas mayoritariamente (es decir, más del 50%) con evaluación grupal. Por lo tanto, la evaluación individual podría no ser necesaria o tener menor relevancia, pero nunca ser más importante que la grupal. Sobre todo, si se implementan metodologías activas, apoyadas en el aprendizaje colaborativo y cooperativo con grupos de estudio. La importancia de la evaluación grupal e individual depende, obviamente, del criterio profesional del docente.

4. Resultados

La nueva metodología fue puesta en práctica durante el año 2024 en el contexto de la asignatura de Cálculo Numérico que se dicta en el segundo año del plan de estudio de la carrera. Esta no enseña la programación, pero sí, la usa como un recurso en las prácticas académicas para la implementación de aplicaciones numéricas. La asignatura desarrolla toda la praxis entre ocho y diez guías de trabajos prácticos. Cada guía responde a una o dos unidades temática del currículo, e incorpora, aproximadamente, un 80% de problemas relacionados con la programación algorítmica en computadoras, y un 20% con ejercicios de consolidación de conceptos.

Durante los años de 2016 y 2017, se hicieron los primeros experimentos en el que se utilizó únicamente la metodología ágil de Scrum en las clases prácticas de la asignatura. Los resultados experimentales mostraron un impacto positivo en el alumnado comparado con los de años previos, en el que se usó enseñanza tradicional basada en clases magistrales [35]. Se delinearon las actividades básicas de la metodología ágil que provocaba mayor motivación en el aprendizaje de los alumnos. Se especificaron mecanismos adaptativos en la enseñanza y se realizaron una evaluación pedagógica de la estrategia en 2015, que incluía mejoras respecto del experimento realizado en 2014 [36]. Los resultados de la evaluación metodológica fueron muy alentadores. El experimento se repitió durante el año 2018, y en 2019 se implementó una variante del año anterior.

Los resultados concluyentes al término de los años 2018 y 2019, fue que el plantel docente dedicaba gran esfuerzo para poner en práctica la metodología, como consecuencia de una capacitación débil en el desarrollo ágil con Scrum. Aun así, el índice del rendimiento académico pasó del 55 % al 85 % promedio de alumnos que aprobaron el cursado de la asignatura. El índice de abandono del cursado descendió al 5 % promedio. En los años 2022 y 2023 se retomaron el tradicional sistema de enseñanza basado en clases magistrales. Durante esos años, el plantel docente profundizó su capacitación en el desarrollo ágil con Scrum, y se perfiló la actual propuesta pedagógica, puesta en práctica a partir del año 2024. En los años 2020 y 2021 no se aplicó la metodología como consecuencia de la pandemia que afectó a todo el mundo.

Tabla 1. Registros anuales de 2013 al 2024 de alumnos matriculados en Programación Numérica, y las cantidades de regularizados, libres y los que abandonaron el cursado.

Año	Cant. Inscriptos	C. Regularizados	C. Libres	C. Deserción
2013	66	31	9	26
2014	58	33	9	16
2015	38	18	7	13
2016	74	56	14	4
2017	85	71	9	5
2018	78	67	7	4
2019	82	37	23	22
2022	67	38	9	20
2023	70	44	12	14
2024	79	71	5	3

En el año 2024, las estadísticas resultantes fueron: el 89.87% de los alumnos inscriptos regularizaron la asignatura, mientras que el 6.33% quedó libre, y el 3.8% abandonó el cursado. Estas estadísticas implican una mejora significativa del rendimiento académico respecto de los años anteriores (ver tabla 1).

Del análisis reflexivo del diagnóstico y de las evaluaciones del proceso, se puede advertir una actitud de reserva y poco participativo del alumnado en las etapas iniciales del cursado de la asignatura. La asignación de roles, división de tareas y organización fueron los factores de mayor impacto en el trabajo grupal. La administración adecuada de los comportamientos psicológicos del alumnado, la instrumentación del tutoreo de pares, y las prácticas centradas en la dinámica cooperativa, permitieron el desarrollo de nuevas habilidades, no solo como programador sino también como ser humano. Así, se generó un ambiente en el que se propició un mayor diálogo entre los pares, confianza, actitud reflexiva, activa y participativa. Todas estas acciones facilitaron a la cátedra detectar rápidamente problemas cognitivos y sociales, e instrumentar acciones correctivas y adaptativas.

5. Conclusiones

La implementación de una metodología como la descrita requiere necesariamente la formación previa del cuerpo docente en el desarrollo ágil de software con Scrum. Se deben tener conocimientos claros sobre el aprendizaje basado en problemas y las estrategias basadas en proyectos, así como dominar las actividades pedagógicas de aprendizaje colaborativo y cooperativo. La importancia de la formación previa de los docentes radica en la necesidad de contar con perfiles docentes que reaccionen rápidamente en la instrumentación de soluciones pedagógicas en situaciones no planificadas. La rapidez con que se desarrolla la dinámica metodológica, y los tiempos limitados para el desarrollo curricular de la asignatura, hacen que, en algunas situaciones, no puedan detectarse con inmediatez las posturas psico-culturales de los estudiantes que les impiden desarrollar habilidades sociales y colaborativas en el trabajo en

equipo. Superar estos escenarios implica un mayor riesgo en el abandono de la asignatura por parte del protagonista. Pueden plantearse inconvenientes mayores de marginalidad o integración en un grupo de estudio, que requieran el asesoramiento de un profesional psicopedagógico. Captar la atención de los estudiantes, generar en ellos el sentido de aprendizaje, crear espacios de reflexión y discusión del conocimiento, planificar actividades que induzcan el desarrollo del pensamiento crítico, son solo algunas de las tareas que demandan docentes profesionalizados, no solo en los temas del currículo de la asignatura, sino también en lo pedagógico. La etapa de diagnóstico e indagación es de suma importancia para el desarrollo de la propuesta metodológica. Esta proporciona información suficiente para planificar la estrategia que aumente la calidad educativa. Las características adaptativas y contextualizadas de la etapa de diseño aseguran el éxito de las etapas siguientes. La velocidad de reacción y los cambios adaptativos en marcha de la etapa de implementación favorecen el desarrollo académico de la realidad laboral en el contexto áulico.

Referencias

1. Bennedsen J. and Caspersen M. E. (**2019**). Failure rates in introductory programming: 12 years later. *ACM Inroads*. Vol. 10 (2):30–36. doi: 10.1145/3324888

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. doi: 10.1145/1151954.1067453

3. CESSI (2007) *Reporte de la Cámara de empresas de Software y Servicios Informáticos.* Diario La Nación. Tecnología, [Portal de Internet] [Accedido: 17/10/2020]. Available from https://www.lanacion.com.ar/tecnologia/el-60-de-los-alumnos-de-informatica-abandona-la-carrera-en-primer-ano-nid966300/.

4. Perazo C. (2013) *Reporte del Ministerio de Educación. Secretaría de Políticas Universitarias.* Diario La Nación. Tecnología, [Portal de Internet] [Accedido: 17/10/2020]. Available from http://www.lanacion.com.ar/1632045-el-80-de-los-estudiantes-de-carreras-informatica.

5. Llamas L. (2019) *Tras el receso invernal, abandona gran parte de los universitarios.* Editorial Río Negro S.A., [Portal de Internet] [Accedido: 17/10/2020]. Available from https://www.rionegro.com.ar/tras-el-receso-invernal-abandona-gran-parte-de-los-universitarios-1058778/.

6. Phillips A. (**2020**) Informática sigue siendo una de las carreras con más abandono. *Intentando entender los posibles motivos*. 01/07/2020. Xataka, Portal Webedia en Internet. https://www.xataka.com/otros/informatica-sigue-siendo-carreras-abandono-intentando-entender-posibles-motivos

7. 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.

8. Gallego Durán F. J., Satorre-Cuerda R., Compañ-Rosique P. and Villagrá-Arnedo C. (**2018**). Explicando el bajo nivel de programación de los estudiantes. *ReVisión*. Vol. 11 (1):33-42. http://www.aenui.net/ojs/index.php?journal=revision

9. Canedo E. D., Santos G. A. and Leite L. L. (**2018**). An Assessment of the Teaching-Learning Methodologies Used in the Introductory Programming Courses at a Brazilian University. *Informatics in Education*. Vol. 17 (1):45-59. doi: 10.15388/infedu.2018.03

10. Villalobos J. A., Casallas R. and Vela M. (**2007**) Una Solución Moderna e Integral al Problema de Enseñar Programación. *XXVII Reunión Nacional de Facultades de Ingeniería y VI EncuentroIberoamericano de Instituciones de Enseñanza de la Ingeniería*. Octubre de 2007, Cartagena de Indias, Colombia

11. 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. *International Journal of Computer Applications (IJCA)*. Vol. 77 (12):27-32. doi: 10.5120/13448-1311

12. 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. doi: http://dx.doi.org/10.1016/j.sbspro.2012.09.277

13. 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. doi: 10.1023/a:1015362608943
14. Mancy R. and Reid N. (**2004**). Aspects of Cognitive Style and Programming. *Proceedings of 16th workshop of the Psychology of Programming Interest Group (PPIG 2004).* Institute of Technology. Carlow, Ireland. http://www.ppig.org/workshops/ppig-2004-16th-annual-workshop
15. Bennedsen J. and Caspersen M. E. (**2005**). Revealing the programming process. *ACM SIGCSE Bulletin*. Vol. 37 (1):186-190. doi: 10.1145/1047124.1047413
16. Sancho-Thomas P., Fuentes-Fernández R. and Fernández-Manjón B. (**2009**). Learning teamwork skills in university programming courses. *Computers & Education*. Vol. 53 (2):517-531. doi: 10.1016/j.compedu.2009.03.010
17. Roberts F. S. (**2011**). The Challenges of Multidisciplinary Education in Computer Science. *Journal of Computer Science and Technology*. Vol. 26 (4):636-642. doi: 10.1007/s11390-011-1164-1
18. Houstis E. N., Rice J. R., Ramakrishnan N., Drashansky T., Weerawarana S., Joshi A. and Houstis C. E. (**1998**). Multidisciplinary Problem-Solving Environments for Computational Science. *Advances in Computers*. pp. 401-438. Elsevier doi: 10.1016/S0065-2458(08)60209-0
19. Janz B. D. (**1999**). Self-directed teams in IS: correlates for improved systems development work outcomes. *Information & Management*. Vol. 35 (3):171-192. doi: 10.1016/S0378-7206(98)00088-3
20. Lewandowski A. and Bourguin G. (**2006**). A New Framework for the Support of Software Development Cooperative Activities Vol. 5 pp. 36-43. doi: 10.5220/0002493100360043
21. Hershkovitz A., Sitman R., Israel-Fishelson R., Eguíluz A., Garaizar P. and Guenaga M. (**2019**). Creativity Inside and Outside Programming Learning. *Proceedings of the 9th International Conference on Learning Analytics & Knowledge.* pp. 293-298. Association for Computing Machinery. Tempe, Arizona, USA.
22. Tsai M.-J., Wang C.-Y. and Hsu P.-F. (**2018**). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research*. Vol. 56 (8):1345-1360. doi: 10.1177/0735633117746747
23. Kotovsky K. (**2003**). Problem Solving – Large/Small, Hard/Easy, Conscious/Nonconscious, Problem-Space/Problem-Solver: The Issue of Dichotomization. *The Psychology of Problem Solving*. pp. 373-384. Cambridge University Press. Cambridge. doi: 10.1017/CBO9780511615771.013
24. Fox R. and Farmer M. (**2011**). The Effect of Computer Programming Education on the Reasoning Skills of High School Students. *Paper of The 2011 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'11).* July 18-21, 2011. WorldComp 2011 Proceedings. http://worldcomp-proceedings.com/proc/p2011/FEC2843.pdf
25. Mudgett D. R. (**2014**). Teaching and Learning in Technical IT Courses. *Innovative Practices in Teaching Information Sciences and Technology: Experience Reports and Reflections*. pp. 31-42. Springer International Publishing. Cham. doi: 10.1007/978-3-319-03656-4_4
26. López-Cruz O., Mora A. L., Sandoval-Parra M. and Espejo-Gavilán D. L. (**2017**). Teaching Computer Programing as Knowledge Transfer: Some Impacts on Software Engineering Productivity. *Proceedings of Trends and Applications in Software Engineering.* pp. 145-154. Springer International Publishing. Cham.
27. Johnson D. W., Johnson R. T. and Holubec E. J. (**1999**). *El aprendizaje cooperativo en el aula*. Paidos Educador (Book 144). Paidos, Argentina.
28. García R., Traver J. A. and Candela I. (**2019**). *Aprendizaje cooperativo: Fundamentos, características y técnicas*. Colección AcciónSocial. 2 Ed. ICCE (Instituto Calasanz de Ciencias de la Educación), Madrid.
29. Faja, S. (2014). Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses. Information Systems Education Journal (ISEDJ), 12(6), 36-45. https://files.eric.ed.gov/fulltext/EJ1140923.pdf
30. Back K. and Andres C. (**2005**). *Extreme Programming Explained: embrace change*. 2ª Ed. Addison-Wesley Professional, Boston, MA.
31. Williams L., Wiebe E., Yang K., Ferzli M. and Miller C. (**2002**). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education*. Vol. 12 (3):197-212. doi: 10.1076/csed.12.3.197.8618
32. Mentz E., van der Walt J. L. and Goosen L. (**2008**). The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education*. Vol. 18 (4):247-260. doi: 10.1080/08993400802461396
33. Barrows H. S. (**1996**). Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning*. Vol. 1996 (68):3-12. doi: 10.1002/tl.37219966804

34. Alaimo M. (**2013**). *Proyectos Ágiles con Scrum*. Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos. 1 Ed. (np. 126) Kleer Agile Coaching & Training, Buenos Aires, Argentina. http://www.kleer.la/es/publicamos/scrum

35. Barberis A. R. and Del Moral L. E. (**2016**). Scrum como Herramienta Metodológica en el Entrenamiento Cooperativo de la Programación: De la Teoría a la Práctica. *Proceedings of XI Congreso de Tecnología en Educación y Educación en Tecnología 2016 (TE&ET 2016).* pp. 365-374. Red UNCI. Universidad de Morón, Argentina.

36. Del Moral L. E. and Barberis A. R. (**2016**). Evaluación de una Propuesta Metodológica para el Entrenamiento de la Programación. *Proceedings of 4to Congreso Nacional de Ingeniería en Informática / Sistema de Información (CoNaIISI 2016).* Confedi. Universidad Católica de Salta, Argentina.