Práctica Pedagógica basada en el Entrenamiento del Desarrollo Ágil de la Programación

Pedagogical Practice based on Training of the Agile Development of Programming

Angel R. Barberis[1], Lorena E. Del Moral Sachetti[2], Jorge Silvera[1], Eusebio Méndez[2]

[1]Facultad de Ciencias Exactas – Universidad Nacional de Salta (UNSa)
Av. Bolivia 5.150 – Capital – Salta – Argentina

[2]Sede Regional Orán – Universidad Nacional de Salta
Alvarado 751 – San Ramón de la Nueva Orán – Salta – Argentina.

barberis@unsa.edu.ar, lorena.dms.7@gmail.com, jsilvera@hotmail.com, eusebio.mendez@gmail.com

Resumen. *Este artículo presenta una estrategia metodológica para ser aplicada durante la práctica académica de la programación de computadoras. Se trata de una estrategia de aprendizaje activa, participativa y centrada en el alumno para ser desarrollada durante las clases prácticas de una asignatura que usa la programación de computadora como un recurso primordial en la praxis. Esta metodología permite a las cátedras entrenar a sus alumnos en el desarrollo ágil de aplicaciones informáticas como un enfoque superador, a la mera práctica de fijación de contenidos. Al mismo tiempo, permite promover en el estudiante, acciones cooperativas y colaborativas en el trabajo en grupo, que les facilita la adquisición de habilidades propias del programador profesional.*

Abstract. *This article presents an active, participatory and student-centered learning methodology to be developed during practical classes of a subject that uses computer programming as a primary resource in practice. This strategy allows the chairs to train their students in the agile development of computer applications as a superior approach, to the mere practice of fixing content, allowing at the same time, to promote in the student, cooperative and collaborative actions in group work, which facilitates the acquisition of skills of the professional programmer.*

1. Introducción

El principal reto que afrontan los sistemas educativos del mundo entero es la calidad. En torno a ello, desarrollan su función primaria de preparar y formar a ciudadanos para una sociedad cambiante, en constante evolución, incierta y compleja [Pérez Gómez, 2010]. Esta sociedad cambiante ha evolucionado hasta convertirse en la Sociedad del Conocimiento, acompañada de un fuerte desarrollo tecnológico [Alfonso Sánchez, 2016]. Esta nueva sociedad del siglo XXI, influenciada por las tecnologías en sus

diversas formas, demanda nuevas estrategias de formación académica para captar la atención de sus individuos, y generar en ellos, motivación para seguir aprendiendo en la constante evolución hacia el futuro. Actualmente, el sistema educativo universitario se encuentra en una etapa de transición y adaptación a los nuevos cambios tecnológicos, políticos y sociales como consecuencia de una reacción tardía a las nuevas demandas de una sociedad globalizada [Fernández March, 2006]. Consecuentemente, todavía existe en las universidades un sistema educativo basado en clases magistrales y centradas en el docente, conformando una metodología obsoleta que propicia un ambiente de enseñanza y aprendizaje pasivo, y muy poco efectivo [Ackoff and Greenberg, 2008]. Esta realidad incide severamente en carreras tecnológicas.

En los primeros años de estudios universitarios en carreras informáticas, los alumnos experimentan una amplia gama de dificultades y deficiencias relacionadas con el aprendizaje y praxis de la programación de computadoras. Estas dificultades están relacionadas con una escasa práctica de la programación de aplicaciones informática útiles, un incipiente desarrollo de habilidades del pensamiento crítico, algorítmico y de resolución de problemas [Robins, 2019]. La programación vista como una habilidad es muy difícil de aprender, y aún mucho más, de dominar. Por lo que, después de los cursos introductorios, varios estudiantes todavía tienen dificultades para escribir programas simples y leer el flujo algorítmico de un código fuente [Havenga, Breed et al., 2013], entre otras habilidades. Ante estas dificultades, el estudiante se siente desmotivado [Gomes and Mendes, 2014], con una acentuada pérdida de interés por el aprendizaje [Bennedsen and Caspersen, 2008], que los induce al abandono del cursado de la asignatura, o bien, a alcanzar un bajo rendimiento académico. Estos inconvenientes impactan severamente en asignaturas de los primeros años que no enseñan la programación de computadoras, pero sí la contemplan entre los objetivos de la práctica académica [Barberis, Del Moral Sachetti et al., 2021].

Ante este panorama, surge la necesidad de formalizar una estrategia metodológica que permita generar en el alumno un interés genuino por aprender, practicar y entrenarse en la programación de computadoras. Una manera de motivar a los estudiantes en la programación es introducirlos en un ambiente de desarrollo ágil de aplicaciones informáticas. Así, las cátedras que usan la programación en sus prácticas académicas, podrán contar con un recurso que les permita entrenar a sus alumnos en el desarrollo ágil de software, como un enfoque superador a la mera praxis de fijación de contenidos. La metodología innovadora se sustenta en una adecuación de Scrum al servicio académico. Permite en el estudiantado, promover acciones cooperativas y colaborativas en el trabajo en equipos, la adopción de capacidades de auto organización, comunicación y motivación por el aprendizaje. Al mismo tiempo, les facilita la adquisición de habilidades del pensamiento crítico, lógico y de resolución de problemas, como así también otras, propias del programador profesional. Básicamente, la estrategia consiste en la apropiación de los conceptos del Aprendizaje Cooperativo Basado en Problemas. Bajo el marco referencial del desarrollo ágil de Scrum, la estrategia se diseña para visualizar lo que el alumno es capaz de hacer con el aprendizaje que va adquiriendo, y resaltar aspectos importantes de la programación y la resolución de problemas. En pocas palabras, se trata de llevar la realidad laboral al aula, en un ambiente de estudio activo, simplificado y controlado por los docentes.

2. Trabajos Relacionados

La agilidad permite el desarrollo rápido de programas informáticos mediante la adopción de modelos iterativos e incrementales donde se intercalan los análisis, el diseño y las actividades de construcción. Hay varias metodologías con estas características, todas las cuales se basan en un conjunto de principios recopilados en el manifiesto ágil, entre los cuales, se encuentra Scrum [Pressman and Maxim, 2015]. La literatura discute ampliamente la filosofía, principios y prácticas de metodologías ágiles [Pressman and Maxim, 2015; Fuertes and Sepúlveda, 2016; Lasa Gómez and Alvarez García, 2018]. En este campo, la investigación empírica se ha centrado más en entornos de la ingeniería de Software que en los escenarios educativos [Fuertes and Sepúlveda, 2016]. Aun así, las prácticas de las metodologías ágiles en el campo de la enseñanza también han llamado la atención de algunos investigadores. Por ejemplo, uno de los proyectos más difundido es eduScrum, que adapta Scrum a la educación secundaria en Holanda. Básicamente, eduScrum [Wijnands and Stolze, 2019] es un marco de trabajo para un proceso de educación activa, colaborativa y co-creativa. Permite a los estudiantes hacer tareas de acuerdo con un ritmo fijo, planificar sus actividades y hacer un seguimiento de su propio progreso. El docente determina las tareas a realizar, los aconseja y los motiva. Un proyecto similar que combina dos metodologías ágiles Scrum y Kanban al servicio de la educación es Blueprint [Godoy, Cruz et al., 2019], que bajo el patrocinio de Blueprint Education, propone un marco de enseñanza activa para escuelas secundarias. Blueprint Education[1], es una organización sin fines de lucro que se especializa en opciones académicas para estudiantes no tradicionales. En el enfoque educativo, los estudiantes utilizan Scrum como una forma atractiva y autoorganizada de trabajar de manera colaborativa y dinámica. Así, las escuelas guiadas por los proyectos de eduScrum y Blueprint Education usan Scrum para ayudar a grupos de estudiantes a aprender de forma más efectiva y desarrollarse de una manera divertida.

En el ámbito Universitario, se reportan experiencias del uso de Scrum en la enseñanza de algunas asignaturas. Así por ejemplo, se usó la metodología ágil para dinamizar y acercar los beneficios del trabajo activo en la enseñanza de la asignatura Gestión y Organización de Empresas [Gómez, 2020]. Otros trabajos de investigación, además de describir la experiencia con alumnos de una facultad de educación, también resaltan los beneficios que aporta Scrum en la interacción entre pares y en el trabajo en grupo, dotando al estudiante de habilidades de organización, autonomía y capacidades de implicarse en proyectos comunes [Onieva López, 2018]. También se derivó un enfoque para enseñar la programación basada en desarrollo ágiles; que mediante la utilización de práctica de prueba y error se promovió el pensamiento crítico y la comunicación entre los estudiantes, reportándose como resultado, una fuerte motivación del estudiantado por aprender [Kofune and Koita, 2014].

3. Estrategias Pedagógicas que componen la Metodología Innovadora

Atraer y sostener la atención de los estudiantes tanto en clases teórica como en las prácticas académicas es uno de los mayores retos que debe afrontar cualquier docente. Cada individuo participante de una clase académica tiene intereses, ritmos de

[1] http://www.blueprinteducation.org/

aprendizaje y niveles de concentración distintos. Para diseñar una metodología adecuada a estos alumnos, se disponen de varias estrategias pedagógicas que forman el grupo de *metodologías activas*. Por lo que usando una, o una combinación de éstas, se podrá lograr una intervención personalizada y que se enriquezca por sus diferencias [Bernal González and Martínez Dueñas, 2017]. Las metodologías activas son estrategias educativas que pone al alumnado en el centro del proceso de enseñanza-aprendizaje. Se las define como procesos interactivos de la educación que se centra en la comunicación activa y en la interconexión entre el profesorado, los estudiantes y el material didáctico. Estas metodologías toman como principal punto de partida los intereses del alumnado. Intentan inducir al estudiantado a adquirir las competencias necesarias para el desarrollo diario y su vida laboral. Fomentan la participación activa, lo cual mejora notablemente los niveles de atención e interés [Silva Quiroz and Maturana Castillo, 2017]. La figura docente se convierte en un facilitador y guía del aprendizaje, contextualizando los conocimientos a situaciones reales del mundo actual, y reorientando las necesidades de los alumnos a lo largo del proceso [Peralta Lara and Guamán Gómez, 2020].

Se sabe que cada estrategia activa es *buena para determinadas situaciones* del proceso de enseñanza-aprendizaje. El uso exclusivo de una única estrategia no logra la diversidad de metas y objetivos que profesores y alumnos buscan alcanzar. Pues, el conjunto de variables, como número y características de los alumnos, enfoque de la materia, y otras, como las sociales y culturales, condicionan la pertinencia de un determinado método. Así, la innovación que se propone, aborda un conjunto de estrategias activas que aportan los mejores beneficios en la combinación metodológica. De esta manera, se logra un recurso pedagógico apropiada para el contexto en el que se desarrollan las clases prácticas de la programación de computadoras.

3.1. Aprendizaje Cooperativo

Uno de los ejes centrales de la metodología puesta en práctica fue el propiciar un ambiente cooperativo en el que los alumnos puedan desarrollar habilidades propias del trabajo en equipo, específicamente en el contexto de la programación de aplicaciones para computadoras.

Más que una metodología es una estrategia de aprendizaje que se combina con otras metodologías activas. Se caracteriza por la organización de la clase en grupos mixtos y heterogéneos. En estos grupos, los alumnos trabajan de forma conjunta y coordinada para resolver tareas académicas, tratando de profundizar en su aprendizaje. Es un proceso educativo que valora especialmente la importancia de las relaciones interpersonales y la necesidad de socialización e integración de todos y cada uno de los estudiantes [García, Traver et al., 2019].

La cooperación es trabajar en conjunto para lograr objetivos compartidos. Dentro de las actividades cooperativas, los individuos buscan resultados que sean beneficiosos no sólo para ellos mismos, sino también, para todos los demás miembros del grupo. El aprendizaje cooperativo es el uso educativo de grupos pequeños en el que los estudiantes trabajan juntos para maximizar su propio aprendizaje y el de los demás [Johnson, Johnson et al., 1999].

El aprendizaje cooperativo, cuidadosamente estructurado, involucra a personas que trabajan en equipos para lograr un objetivo común, en condiciones que involucran

tanto la *interdependencia positiva* (todos los miembros deben cooperar para completar la tarea) como la *responsabilidad individual y grupal* (cada miembro es responsable del resultado final de la tarea que se le asigna). Gracias a una *interacción estimuladora* los miembros comparten recursos, conocimientos, se motivan por pequeños logros y se alientan en los fracasos, buscando en todo momento el éxito de los demás. El trabajo cooperativo requiere que los alumnos aprendan *prácticas interpersonales y grupales*. Estas son necesarias para formar parte de un grupo, y tener habilidades de trabajo en equipo, ya que deben saber cómo comunicarse, cómo dominar las situaciones que se les presentan, tomar decisiones, manejar conflictos, entre otros. Por último, en el marco cooperativo, el equipo debe realizar una *evaluación* para saber en qué medida han podido alcanzar sus objetivos. Para ello, analizan los aspectos en que fallaron y en el que triunfaron, y reconocen las acciones positivas y negativas de cada miembro del grupo. De esta manera, se pueden tomar acciones estimuladoras o correctivas, según corresponda [Johnson, Johnson et al., 1991].

El aprendizaje cooperativo incrementa la motivación y la participación gracias a la interacción entre profesores y alumnos. Esta interacción posibilita un intercambio continúo de ideas, el desarrollo de habilidades comunicativas y sociales, y la superación de actitudes negativas. Los estudiantes al sentirse apoyados y en confianza, son capaces de consolidar su propio estilo de aprendizaje [García, Traver et al., 2019].

En síntesis, el trabajo cooperativo es una estrategia de gestión del aula que privilegia la organización del alumnado en grupos heterogéneos para la realización de las tareas y actividades de aprendizaje. Algunos de los principales beneficios del aprendizaje cooperativo radican en el aumento de la motivación por el aprendizaje en general, y por la realización de las distintas tareas académicas. Se fomentan actitudes de implicación e iniciativa. Aumenta el grado de comprensión de lo que se hace en clase. Profundiza y mejora el trabajo en equipo, así como sus resultados. Y fomenta habilidades sociales, tales como, interacción, respeto, comprensión, solidaridad y resolución de conflictos.

De esta manera, el aprendizaje cooperativo es una estrategia de gran valor a tener en cuenta, ya que mejora de forma significativa los procesos educativos gracias a la colaboración e interacción entre los alumnos.

3.2. Programación por Pares en un Ambiente Cooperativo

Una técnica educativa que tiene elementos comunes con el aprendizaje cooperativo es la programación por pares [Williams, 2011]. En esta forma de cooperación, dos programadores trabajan juntos en una computadora. En cualquier momento particular, 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 "navegador") puede estar observando activamente el trabajo del "conductor", ayudando a resolver posibles fallas, analizando alternativas de solución, indagando sobre los conocimientos requerido, etc. Los roles de "conductor" y "navegador" se intercambian periódicamente entre los dos miembros del equipo. La programación por pares se popularizó originalmente como parte de la metodología de desarrollo de software Extreme Programming [Back and Andres, 2005]. Las investigaciones en la literatura, reportan que los programadores por pares producen código de mayor calidad en la mitad del

tiempo, en comparación con los programadores que trabajan solos [Williams, Wiebe et al., 2002]. También se ha encontrado que la técnica es efectiva para los aprendices de la programación, lo que lleva a un mejor aprendizaje y satisfacción de los estudiantes, y reduce la frustración en el desarrollo cognitivo [Braught, Eby et al., 2008; Mentz, van der Walt et al., 2008].

El aprendizaje cooperativo utiliza métodos similares a la programación por pares para ayudar a los estudiantes a aprender sobre los procesos de programación y resolución de problemas [Mentz, van der Walt et al., 2008]. Sin embargo, en un ambiente que propicie un aprendizaje cooperativo, la programación por pares, guía a los estudiantes a través de diferentes niveles de cooperación. Así por ejemplo, en las primeras etapas, el grupo completo puede realizar una lluvia de ideas para resolver un problema. En una etapa posterior, los estudiantes podrían trabajar en parejas 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 los problemas por sí mismos, con asistencia de otros miembros del grupo si es necesario. Con este enfoque incremental se ofrece incluso más ventajas que la programación estricta por pares. Al principio, todos en el grupo están aprendiendo a abordar una tarea de programación. Por tanto, es útil tener tantos puntos de vista diferentes como sea posible. A medida que se desarrollan sus habilidades de programación y resolución de problemas, los estudiantes progresan para trabajar en parejas. Finalmente, tienen la oportunidad de desarrollar confianza resolviendo problemas individualmente (aún con el apoyo del grupo).

Incluir más de dos miembros en un grupo cooperativo permite mayor flexibilidad en la asignación de roles relacionados con el trabajo en equipo y el desarrollo de programas. Algunas organizaciones grupales involucran un número sustancial de roles, que serían difíciles de implementar en un entorno estricto de programación por pares.

3.3. Aprendizaje Cooperativo Basado en Problemas

Una de las metodologías activas más conocidas y utilizadas es el Aprendizaje Basado en Problemas (ABP) [Wood, 2003]. Consiste en el diseño, programación e implementación de un conjunto de tareas asociadas a una misma temática. Se trata de un método educativo muy participativo que sirve para trabajar un tema concreto de forma transversal. En clase se plantea un problema del tipo proyecto que los alumnos deben resolver y encontrar la mejor solución. Para lograr el reto, los estudiantes deben trabajar en equipo a través de la investigación y recopilación de información, discurriendo, analizando y debatiendo. Implica diversas actividades constructiva, autodirigida, colaborativa y contextual [Yew and Goh, 2016]. Puede ser vista como un proceso en el que los estudiantes se enfrentan a problemas de diversa índole en grupos reducidos, compartiendo durante esa experiencia la posibilidad de practicar y desarrollar habilidades, de observar y reflexionar sobre actitudes y valores que en el método de enseñanza “convencional” difícilmente podrían ponerse en acción [Bawamohiddin and Razali, 2017].

El desarrollo de un ABP es complejo, y presenta las siguientes características:

- En un ABP normalmente se trabaja de forma cooperativa, aunque también puede combinarse con tareas individuales de investigación o creación.
- Con este proyecto educativo, se pretende que los aprendizajes estén conectados con los propios estudiantes de tres formas:
  - *Contextual*: El problema a resolver, debería ser un proyecto que conecta al estudiante con la realidad percibida.
  - *Motivacional*: Se procura trabajar con temáticas actuales e interesantes para el grupo clase.
  - *Productiva*: El aprendizaje debe tener una utilidad práctica e inmediata para el alumnado, de manera que le sirva en su día a día para desarrollar otras acciones.
- El producto final con el que siempre finaliza el proyecto debería ser creado, ejecutado y presentado normalmente en grupo.
- Los alumnos se convierten en los protagonistas de sus propios aprendizajes, potenciando el trabajo en equipo, el pensamiento crítico y la investigación, así como la autonomía y la responsabilidad individual.

El proceso culmina con una presentación del producto o ejecución de la solución, que refleja lo que el alumnado es capaz de hacer con los aprendizajes que ha adquirido durante el proyecto.

3.4. Marco del Desarrollo Ágil: Scrum

El propiciar un ambiente cooperativo sobre la base de un ABP no lo es todo, a menos que vaya acompañado de un marco referencial que facilite su desarrollo. En este contexto, se utiliza la técnica del desarrollo ágil de software conocida como *Scrum* para la programación de aplicaciones informáticas. La dinámica de trabajo que propone Scrum, combinada adecuadamente con un conjunto de actividades que potencian el desarrollo de un *aprendizaje cooperativo basado en problemas*, dan lugar a una metodología activa que permite *entrenar* a estudiantes en la programación ágil de aplicaciones informática, como un enfoque superador a la mera práctica de contenidos curriculares.

Desde el punto de vista informático, Scrum se percibe como un marco de trabajo que permite fortalecer y afianzar las relaciones del equipo humano que interactúa cooperativamente en el desarrollo ágil de software [Alaimo, 2015]. Propone un conjunto de prácticas y artefactos que posibilita transformar un problema complejo en simples actividades, de resolución inmediata y progresiva. Genera al mismo tiempo, un contexto relacional, interactivo y cooperativo, de inspección y adaptación constante para que los involucrados vayan creando su propio estilo de trabajo. De esta manera, se propicia un ambiente en el que se dota al equipo humano de los mecanismos necesarios para que desarrollen buenas prácticas de trabajo en un contexto complejo. Tal como se expresa en [Palacio, 2007], Scrum como propuesta de trabajo ágil es básicamente: un modo de

desarrollo de carácter adaptable; orientado a las personas antes que a los procesos; y emplea desarrollo ágil iterativo e incremental.

La dinámica de Scrum se encuentra apoyada en tres roles básicos [McKenna, 2016]: el Product Owner, el Scrum Master y el Equipo de Desarrollo Scrum. El Product Owner es la persona que conoce los requerimientos del cliente y usuarios finales. Tiene la responsabilidad de conducir al equipo de desarrollo hacia el producto adecuado e interactúa con ellos a través del Scrum Master. El Scrum Master es el líder del equipo de desarrollo que vela por la utilización adecuada de las prácticas 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, protegiéndolo y aislándolo de interrupciones para garantizar su productividad. Finalmente, el Equipo de Desarrollo Scrum, se trata de un grupo de personas que forman un equipo multidisciplinario que cubre todas las habilidades necesarias para generar el resultado o producir valor. Se autogestiona y autoorganiza, y dispone de atribuciones suficientes en la organización para tomar decisiones sobre cómo realizar su trabajo.

En Scrum, el progreso de los proyectos de software se realiza y verifica a través de una serie de iteraciones llamadas Sprints [Palacio, 2007; Alaimo, 2015; McKenna, 2016]. Estos tienen una duración fija y preestablecida. Al comienzo de cada Sprint el equipo de desarrollo realiza un compromiso de entrega, que puede ser una serie de funcionalidades o características del producto en cuestión. Al finalizar la iteración se espera que las características comprometidas estén terminadas, lo que implica su análisis, diseño, desarrollo, prueba e integración del producto. La metodología Scrum, prevé reuniones donde se realizan el proceso de retroalimentación, recopilación y aclaración de los requisitos. Estas reuniones se denominan *Scrum Diario*, *Planificación de Sprint*, *Retrospectiva Sprint*, y la más usada, *reunión de revisión*. El Scrum Diario se realiza al comienzo de cada sesión de trabajo. En ella cada participante describe lo que se ha completado, lo que esperan completar, y se encuentran los impedimentos. La Planificación del Sprint se concreta al comienzo de las iteraciones y consta de dos partes: en la primera, el equipo se compromete a alcanzar un conjunto de objetivos y en la segunda, se identifican tareas específicas. En la reunión de revisión cada equipo de trabajo presenta los requisitos completados y los que aún deben completarse. La Retrospectiva Sprint tiene lugar al final de cada iteración. La finalidad es que cada equipo se focalice en las lecciones aprendidas durante el desarrollo realizado en esa iteración.

Desde la visión pedagógica, Scrum es una metodología activa que se utiliza para fomentar el trabajo colaborativo en el aula a través de proyectos que involucran a toda la clase. Tiene por objetivo lograr la autonomía y la responsabilidad a través del aprendizaje auto dirigido, la calidad de la educación, lo que redunda en mejoras de las calificaciones y, estudiantes motivados en el aprendizaje.

Básicamente, Scrum es el marco referencial más idóneo para propiciar el entrenamiento de los alumnos en la programación de aplicaciones informáticas. Procura que el estudiante organizado en equipos de trabajo se oriente a producir valor final de calidad, colaborando y cooperando en una comunicación interactiva para lograr mejoras continuas del aprendizaje.

4. Innovación en la Práctica Curricular de la Programación

4.1. La metodología

Las características y principios del desarrollo ágil de Scrum brindan la dinámica central de la innovación pedagógica. Ésta no sólo provee los mecanismos idóneos para simular la realidad laboral en el aula, sino que facilita la combinación de los mejores lineamientos de cada metodología Activa utilizada.

La propuesta metodológica se desarrolla de manera puntual en el marco de ciertas actividades. No obstante ello, tiene un carácter sistemático y holístico, pues constituye un enfoque de enseñanza y de consolidación de los aprendizajes. Cabe destacar, que no existe una regla única o general que exprese la forma de diseñar e implementar una metodología de enseñanza y aprendizaje para el universo heterogéneo de estudiantes que una asignatura pueda tener. De la investigación empírica se delinearon un conjunto de actividades organizadas en etapas que pueden diferir o variar según el tipo de alumnado, el enfoque de la asignatura y de la cantidad de programación que se utilice en las prácticas académicas.

Las etapas y momentos que integran la propuesta metodológica se desarrollan según una secuencia temporal que no ocurre de manera lineal, sino por aproximaciones sucesivas.

En términos generales e independientemente de la metodología utilizada, el docente realiza un diagnóstico de la situación inicial de los estudiantes antes de iniciar la etapa de diseño. Esta no es cerrada, sino que implica permanentes indagaciones debido a la diversidad de situaciones que pueden experimentar los estudiantes durante el cursado de la asignatura. La propuesta metodológica involucra un conjunto de factores en permanente cambio, lo que exige flexibilidad y reflexión sobre la práctica durante todo el proceso de aprendizaje. Así, las actividades que se proponen según el momento de su desarrollo en el proceso, se ubican en algunas de las etapas de: *Diagnóstico e indagación*, *Diseño y planificación*, *Implementación – Acción metodológica*, y *Evaluación – Reflexión*.

4.1.1. Diagnóstico e indagación

Al iniciar el ciclo de formación el docente realiza, indefectiblemente, un análisis de la realidad educativa en la que se desarrollará la propuesta pedagógica. El diagnóstico implica, por encima de todo, conocer en profundidad al estudiantado para una adecuada implementación de las etapas siguientes. El docente debe identificar para cada uno de sus estudiantes, cuáles son sus principales intereses, las expectativas, motivaciones, su situación socioemocional, experiencias personales y saberes previos relacionados con los contenidos curriculares de la asignatura. Cabe destacar, que las dificultades de aprendizajes experimentados en asignaturas previas, hacen que un porcentaje del alumnado adquieran la sensación de una mayor dificultad en la asimilación de nuevos conceptos del currículo. Esta sensación de inferioridad acentúa la falta de motivación en el estudiante, que los conduce a una auto marginación del grupo activo de la clase, y en muchos casos, con el consiguiente abandono del cursado. Por ello, es de vital

importancia identificar las características de los estudiantes en el momento inicial del cursado de la asignatura, para facilitar así, un ajuste adecuado de la propuesta didáctica.

La actividad principal de esta etapa es la *evaluación diagnóstica inicial*. Con ella se trata de identificar el estado psico-cognitivo real de los estudiantes, sobre las competencias, técnicas, habilidades y capacidades en el planeamiento y resolución de problemas computacionales adquiridos previamente. La evaluación se estructura en tres bloques. El primero, incluye ejercicios conceptuales necesarios para el abordaje de los contenidos curriculares de la asignatura. El segundo, está orientado hacia la resolución de problemas numéricos simples, que permiten fácilmente identificar las competencias genéricas adquiridas previamente. El tercer bloque, se encamina hacia la identificación de programadores entusiastas o alumnos con habilidades particulares relacionadas con la programación. El bloque contempla resolución de problemas de programación simple en el que se evalúa: estrategia o formas de abordaje del problema, análisis y evaluación de soluciones alternativas y, el estilo de programación utilizada en la implementación de la solución final del problema. El programador entusiasta reúne en mayor medida las habilidades y competencias para el análisis, diseño y codificación de un programa para computadoras.

La segunda actividad está relacionada con la instrumentación de una encuesta que indaga sobre las características de la personalidad del estudiante. Se busca identificar motivaciones, expectativas, relaciones de compañerismo afines, el que se destaca académicamente, el líder natural, el tímido y el aislado, entre otros. La información así obtenida es de vital interés para la heterogeneidad en la conformación de los grupos de estudios en la etapa de diseño.

En general, 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 de saberes.

4.1.2. Diseño y planificación

Una vez que el docente define la realidad educativa en la que se desarrollará el proceso de aprendizaje y consolidación de saberes, se realiza el diseño de la propuesta metodológica *contextualizada*. El desarrollo de una práctica académica basado en el entrenamiento ágil de la programación exige una adecuada planificación por parte de los docentes, a partir de los elementos que integran el proceso de formación académica (objetivos, contenidos, tiempos, instrumentos de evaluación, entre otros), es decir, el currículum. Para ello, se hace necesario contemplar los propósitos didácticos (objetivos y competencias) y los contenidos. Además, hay que dedicar mayor esfuerzo al diseño de las estrategias metodológicas (agrupamientos, actividades, tiempos, materiales) y al sistema de evaluación. Cabe destacar que la adecuación de contenidos a los estudiantes de un ciclo se desarrolla paralelamente 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 lograr. Entre las tareas que se desarrollan en esta etapa, se destaca, *la formación de grupos de estudios*, *el diseño de actividades para la consolidación de saberes,* y *la planificación temporal*.

• *Formación de grupos de estudios*

Hacer que los estudiantes realicen tareas de aprendizaje en grupo es una de las estrategias cooperativas más poderosas para abordar muchos de los problemas que se tienen en la enseñanza a nivel universitario. Una de las formas de aprendizaje cooperativo que favorece el entrenamiento ágil de la programación es organizar a los alumnos en grupos base. Se trata de grupos que se constituyen al inicio del curso, con el objetivo principal de que cada alumno tenga, en primera instancia, a unos compañeros para compartir y resolver dudas sobre el curso. La formación o creación de grupos base constituyen el escenario central en la que cada estudiante se desarrolla. Por lo tanto, son un elemento esencial para el entrenamiento de los alumnos en la programación de aplicaciones informáticas.

En la conformación de los grupos se debe tener en cuenta la heterogeneidad como elemento facilitador, ya que la resolución de diversas tareas exige distintas capacidades que se complementan. Cuanta más diversidad de capacidades, sensibilidades, enfoques, género, más fácil será alcanzar los objetivos perseguidos. Por ello, la determinación de la realidad socioemocional de los estudiantes en el diagnóstico inicial es de gran utilidad, en función del cual es posible determinar con mejor criterio los agrupamientos. Así, la conformación de grupos de estudiantes se realiza con diferentes perfiles (tímidos, extrovertidos, con dificultades para el estudio, líderes naturales, estudiosos, desmotivados, etc.). Se asegura que los alumnos pasivos y con dificultades en el aprendizaje no se reúnan en un mismo grupo homogéneo. En todos los casos, se busca que los alumnos con dificultades socioeducativos se beneficien de las conductas favorables de los activos y sobresalientes, al integrarse en diferentes grupos. Al mismo tiempo, el criterio ayuda a que los estudiantes destacados desarrollen aún más sus capacidades en el accionar cooperativo y colaborativo. La dimensión del grupo de trabajo recomendado oscila entre 4, 6 y 8 alumnos (según el tamaño de la clase), para favorecer la práctica de la programación en pares como recurso interno al equipo de trabajo. Se procura haya al menos un Programador Entusiasta en cada grupo de estudio.

La identificación de programadores entusiastas en el diagnóstico inicial es de vital interés. Tiene el propósito de ser distribuidos tanto como sea posible en los distintos equipos de trabajos o grupos de estudios. Los programadores entusiastas son más activos, participativos, y demuestran mayores habilidades que sus compañeros. La interacción con el resto del grupo, y particularmente con el integrante pasivo, eleva el nivel de participación y motivación de los novatos. Este accionar ayuda a consolidar la experiencia del entusiasta en la programación de computadoras. De esta manera se logra una integración positiva necesaria para la conformación de grupos colaborativos y cooperativos en la dinámica de Scrum, gracias a la efectividad del tutoreo de pares.

• *Diseño de Actividades para la consolidación de saberes*

Las actividades de consolidación de saberes en las prácticas académicas de la programación, determinan el clima del aula y la cultura del aprendizaje. Para desarrollar el interés y la motivación del alumnado se debe ofrecer un amplio rango de actividades que enriquezcan las experiencias generadoras de aprendizaje. El éxito académico estará dado por la planificación cuidadosa en el uso de las diversas metodologías y técnicas

didácticas activas. Es decir, desde un enfoque procedimental, los abordajes por proyectos, estudio de casos, las estrategias de aprendizaje basado en problemas, etc., deben ser bien planificados en la conjugación con un ambiente de desarrollo ágil de software. Cabe destacar, que en los enfoques de aprendizaje y modalidades de enseñanza procedimentales, los estudiantes, fundamentalmente, aprenden haciendo junto a otros. Por lo tanto, se debe enfatizar la importancia que tiene la estructuración y secuenciación de las actividades de aprendizaje. El profesorado debe considerar en el desarrollo de las actividades, las distintas fases a transitar, los productos a solicitar y los saberes que se busca consolidar.

La *adecuación pedagógica contextualizada* al entrenamiento de la programación consiste, básicamente, en seleccionar la actividad, metodología y técnica didáctica ajustada a las características de los alumnos del ciclo. Teniendo en cuenta la realidad educativa inicial, se desarrollan dos o tres proyectos o casos de estudios diferentes, que se conforman con los mismos ingredientes conceptuales brindados por el currículo. Durante la planificación de esta actividad, se pone especial atención en los alumnos pasivos y marginales, en los conceptos fallidos y en las competencias requeridas. Los proyectos informáticos son desarrollados por los grupos de estudios a lo largo de todo el ciclo de la asignatura. Para cada proyecto práctico se fija un objetivo académico, se especifican los conceptos requeridos y la estrategia a ser aplicada por cada grupo base. También se especifican los objetivos de trabajo en equipo y se describen las habilidades interpersonales que se desea utilizar o afianzar en función del enfoque o estrategia de abordaje de los proyectos seleccionados. En todos los casos, no se descuida el cumplimiento de los objetivos generales de la asignatura.

- *Planificación temporal*

Al igual que la formación de grupos y diseño de actividades, la temporalización es esencial a la hora de planificar las prácticas académicas activas en el aula. Los tres elementos están estrechamente relacionados y articulados, y es importante que el profesorado los considere de forma conjunta.

Si bien la duración de cada proyecto en la práctica es igual al ciclo de cursado de la asignatura, hay que determinar la duración de cada módulo componente. Para cada módulo que forma un proyecto, se determinan los objetivos específicos, los conceptos curriculares necesarios y el tiempo de realización. Este último es importante para la dinámica del desarrollo ágil de software mediante Scrum. El tiempo de realización de cada módulo, permite al equipo de desarrollo Scrum (grupo de estudio) planificar los sprint, sprint diario, la distribución de tareas dentro del equipo y las entregas de productos. También se adecúan las expectativas de logros y criterios de evaluación específicas al módulo.

4.1.3. Implementación – Acción metodológica

La temporalización es fundamental en el diseño de las prácticas y entrenamiento en la programación ágil. Sin embargo, no es nada sencillo determinar a priori cuánto tiempo toman las distintas actividades de aprendizaje. Generalmente, cuando se aprende con

otros se tiende a modificar el tiempo de aprendizaje individual. Un estudiante suele tardar menos tiempo en aprender algo de forma individual que en grupo. Es decir, con el aprendizaje individual algunos estudiantes aprenden más rápido, y con el aprendizaje en grupos cooperativos y colaborativo todos aprenden mejor. Por ello, la propuesta metodológica antepone el desarrollo conjunto en grupos de trabajo al individual, sin renunciar a que cada estudiante tenga sus propios aprendizajes.

La etapa de implementación es la que más tiempo demanda, pues en ella se pone en marcha el diseño definido previamente. Este es el momento cuando el estudiantado se transforma en protagonista principal, ya que debe desarrollar acciones concretas para aprender junto a otros. Ellos son quienes deben implicarse, comprometerse y motivarse con la tarea y con el equipo, durante todo el ciclo de cursado de la asignatura. El proceso no sólo depende del estudiante sino también de los docentes, por su rol de facilitador y guía del aprendizaje. Y desde la perspectiva Scrum, los docentes cumplen los roles de Product Owner y Scrum Master cuyas funciones son vitales para el entrenamiento de la programación en un ambiente del desarrollo ágil de aplicaciones informáticas. Durante esta etapa pueden producirse algunos desajustes que serán superados mediante adaptaciones oportunas.

La acción pedagógica de la propuesta innovadora tendrá éxito en la medida que se instruya a los estudiantes en los roles particulares y en los principios de la dinámica Scrum. Para ello se contemplan actividades de preparación dirigidas a los estudiantes según sus roles dentro del grupo de estudio, y a la clase en general.

Se procura *instruir al Programador Entusiasta en la función de Líder de equipo*. El programador entusiasta coordina las actividades cooperativas del grupo. Para ello, el docente de la práctica instruye al entusiasta para un accionar equilibrado, participativo, multifacético y variado, sin que por ello, implique una disminución de su rendimiento académico. Ante la falta de este perfil en el grupo, el puesto puede ser ocupado por un estudiante motivado, responsable o estudioso. Básicamente, se los instruye en el tutoreo de pares y en la forma de evaluar el rendimiento del grupo mediante rúbricas específicas para cada proyecto áulico.

Por otro lado, se *expone al conjunto clase los lineamientos del trabajo en equipo*. Se da instrucción sobre la dinámica de trabajo, auto organización y toma de decisión autónoma. Se motiva a los alumnos para que "fabriquen" su propio contenido del saber nuevo, descubrimiento, indagando en la información existente, cotejándola, valorándola, y preparándola para su comprensión. De esta manera, se logra que el material didáctico disponible para la práctica les resulte significativo en relación a otros conocimientos adquiridos. Se les expone las diferentes estrategias de aprendizaje significativo, para que la conozcan y aprendan a realizar un proceso de metacognición. Así, podrán indagar lo que se sabe, lo que se desea aprender, lo que aún falta incorporar y, relacionar esos conocimientos de modo sistemático y no arbitrario. En general, se busca que el estudiante esté comprometido en su propio proceso de aprendizaje, que tenga una participación activa en la clase, que conozca sus objetivos y cómo lograrlo, en un todo de acuerdo con el *aprender a aprender*.

De manera más específica, se *explica las tareas del trabajo cooperativo con Scrum y la interdependencia positiva*. Se expone los principios y características del desarrollo ágil con Scrum. Se describe las exigencias profesionales del desarrollo de

software y la adaptación de ésta a un ambiente simplificado de la realidad laboral en el aula. Se enfatiza en la responsabilidad individual de los alumnos dentro de la dinámica de funcionamiento con Scrum.

Con el objeto de garantizar la correcta aplicación de la propuesta metodológica, el docente resuelve un caso práctico que pone en evidencia la dinámica del trabajo con Scrum en el contexto áulico. Para asegurar la autonomía en el trabajo en equipo, y sobre todo, el proceder correcto durante la dinámica de Scrum, el profesor propone un proyecto de complejidad simple. Esto facilitará en los alumnos una rápida asimilación de las actividades en el entrenamiento de la programación ágil.

La realidad socioemocional de los alumnos suele ser muy variada, por lo que resulta necesario *entrenarlos para la socialización*. Se aprovecha la etapa inicial de cada sprint, para que el profesor desarrolle una dinámica de grupo motivadora con el objeto de promover una interlocución amena de los miembros del equipo. Se procura, especialmente, la participación de los individuos auto-marginados, tímidos o pasivos.

Siempre en el contexto del proyecto que desarrolla el profesor con el conjunto de la clase, *se entrena a los alumnos para enfrentar situaciones de contingencia*. El marco cooperativo de Scrum prevé revisiones periódicas de la marcha del proyecto. En este contexto, se representa una situación hipotética en el que se resalta el beneficio del recurso de retrospección. Esta consiste en exponer las dificultades que experimenta cada integrante del equipo en la concreción de las tareas asignadas. El equipo reflexiona sobre la forma en que trabaja; se asumen nuevas responsabilidades y se replanifican las actividades retrasadas. Las tareas pueden ser estudiar y conceptualizar los temas teóricos, uso del entorno de programación, programación de algoritmos, diseño de casos de pruebas, colaborar con las tareas de otro compañero, etc. De esta manera, también se afianzan las actitudes cooperativas y colaborativas de los miembros del equipo.

Después de las actividades de preparación del grupo clase, se pone en acción la propuesta metodológica. En la dinámica se conjugan todas las características seleccionadas de las metodologías activas descriptas previamente, bajo el marco referencial de Scrum.

En el aula, el profesor responsable de la asignatura ejerce rol de Product Owner. Es quien determina qué se debe aprender, supervisa la calidad de los resultados y los evalúa. Por otro lado, los Equipos Scrum están formados por cuatro, seis u ocho alumnos, que se auto organizan y son multidisciplinares. Uno de ellos adopta el rol de Líder de equipo, y es elegido por el profesor. El rol de Scrum Master es asignado al profesor responsable de las prácticas académicas, y tiene la responsabilidad de formar el Equipos Scrum teniendo la heterogeneidad de habilidades y perfiles. Además, durante el desarrollo del proyecto, se asegurará de la correcta ejecución de los procesos.

El trabajo comienza adquiriendo una visión general del proyecto y de los tiempos de realización. Los estudiantes identifican funciones y tareas a realizar y, manifiestan sus aportaciones con notas adhesivas pegadas en un tablero. El profesor, por su parte, ejerce de guía pero dejando que sean los alumnos quienes trabajen de forma autónoma. Cada Equipo Scrum crean sus tareas y las organizan en el tiempo asignado a cada módulo del proyecto. Los estudiantes saben cómo y por qué tienen que hacer algo, y adquieren conciencia del grado de importancia que tiene la tarea que hacen, tanto para

ellos mismo como para el equipo. Así, cada uno va descubriendo cuáles son sus cualidades e identifica las de sus compañeros. Se adquieren habilidades como la planificación, la reflexión, la escucha y la retroalimentación. Y se fomenta la idea del trabajo en equipo como vía para conseguir el mejor resultado posible en el desarrollo del proyecto, como así también, en lo académico.

En el inicio de cada módulo, el equipo scrum debe trabajar en la creación de la Pila del Producto (la lista de objetivos específicos del producto), y desarrollar la Pila de Sprint, conocido también, como lista de tareas. Trabajarán en función de los tiempos asignados a cada módulo del proyecto. 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, en donde el equipo reflexiona sobre la forma en que desarrollaron sus tareas, y evalúan el desempeño individual y grupal. Generalmente, la retrospectiva no dura más de 15 minutos. Básicamente, se centra en el proceso del CÓMO se realizaron las labores asignadas, se identifican fortalezas y debilidades, y se planifican acciones de mejora para el siguiente sprint. El resultado de la retrospectiva posibilita 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. El Product Owner (Profesor Responsable de cátedra) puede participar 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 completamente funcional, con todos los requerimientos de cada módulo del proyecto implementados.

4.1.4. Evaluación – Reflexión

En la innovación metodológica es fundamental que la evaluación sea parte integral del proceso de aprendizaje y de consolidación de los saberes. Su aplicación continua aporta información útil para estudiantes, profesores e instituciones, y propicia la discusión sobre las deficiencias detectadas en el aprendizaje. Esto último, induce rápidamente a poner en marcha acciones correctivas. Así, la evaluación puede verse como un proceso continuo, integral y participativo, que permite identificar una problemática, analizarla y explicarla mediante información relevante [Meriño Almaguer, Lorente Rodríguez et al., 2011].

El diseño de la evaluación guarda estrecha relación con la metodología de enseñanza utilizada. En función de cómo la evaluación sea considerada al diseñar el proceso, puede ser percibida como un juicio o como una ocasión para aprender. En la propuesta metodológica, la evaluación es vista como un proceso sistémico. En ella, el docente revisa el modelo pedagógico que sustenta la actividad formativa, y selecciona las estrategias y herramientas que permiten constatar la evolución y el progreso real alcanzado por los estudiantes. Deben coexistir evaluaciones del grupo en su conjunto, a la vez que evaluaciones individuales. Ello permitirá que la evaluación sea coherente en toda su dimensión. La primera decisión que debe tomar el profesorado es la importancia

que dará 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 tanto, la evaluación individual puede no ser necesaria o tener menos relevancia, pero nunca ser más importante que la grupal. Sobre todo, si se implementa metodologías activas, sustentadas en aprendizajes colaborativos y cooperativos con grupos de estudios. La importancia que tenga la evaluación grupal y la individual depende, obviamente, del criterio profesional del docente.

Más allá de cómo se considere la evaluación grupal e individual, también es necesario que los docentes expliciten los criterios para su éxito. Los estudiantes deben conocer lo que se espera que hagan para poder organizar el trabajo del grupo. Es fundamental que los criterios de evaluación estén estructurados, de modo que el alumno (a nivel individual) pueda alcanzarlos sin perjudicar a sus compañeros de grupo. Además de la asimilación de contenidos y de las actividades procedimentales, la evaluación debe considerar otras dimensiones de aprendizajes actitudinales, tales como la escucha activa de lo intercambiado con el resto del equipo, o la crítica a las ideas y no a las personas.

La nueva metodología propone aplicar evaluaciones compartidas, que son aquellas en las que el equipo valora lo aportado por el individuo, la clase evalúa al equipo y el profesor cada producción individual. Así, es oportuno evaluar los trabajos del grupo, tanto desde el punto de vista global, como en relación con lo que ha aportado cada uno de sus miembros individualmente. Debido a los diversos tipos de evaluación, se considera usar distintos instrumentos para llevar a cabo dicha evaluación. Junto a ello, también es relevante facilitar al estudiantado las herramientas de autoevaluación y coevaluación ya que, de este modo, conocerán los criterios que se tomarán en cuenta a la hora de valorar el trabajo. Entre estos instrumentos, y en el ámbito metodológico que se aplica se considera relevante el uso de rúbricas como el recurso más idóneo para reflejar la evolución académica de los discentes. Cabe indicar que la mayor dificultad que debe enfrentar el docente, es el traducir las valoraciones y reflexiones sobre los aprendizajes de los estudiantes en calificaciones. El criterio psicopedagógico del profesor es esencial en este tipo de decisiones. Por ello se resalta la necesidad de contar con una cultura de reflexión sobre la práctica educativa e integrada como práctica habitual de todos los docentes.

Entre las primeras actividades de esta etapa se realiza una *adecuación de las expectativas de logros y criterios de evaluación generales*. Mientras se define el enfoque pedagógico de la práctica a seguir en la etapa de diseño, paralelamente se definen los criterios de evaluación, tanto grupal como individual. La disponibilidad de estos y las expectativas de logros permitirá el armado de las rúbricas para los docentes y para los alumnos.

Las expectativas de logros pueden ser concebidas como metas que pueden modificarse, simplificarse o ampliarse durante el proceso si varían las condiciones previstas. Éstas, deben ser redactadas en términos del alumno de acuerdo al grado de maduración, y a la propuesta de casos de estudios elaborada como consecuencia del diagnóstico áulico inicial. Básicamente, las expectativas de logro y los criterios de evaluación, dirigen el aprendizaje de modo intencional, y deben ser comunicadas a los

alumnos, para que ellos puedan ser partícipe de sus logros, y de lo que aún falta por obtener, en un proceso de metacognición que les ayudará a aprender a aprender.

Durante la acción metodológica del entrenamiento en la programación, se *supervisa el aprendizaje de los estudiantes durante el trabajo en equipo, en función de los roles asignados y de la complejidad del caso de estudio*. El profesor responsable de la asignatura oficia el rol de Product Owner, el profesor responsable de la práctica académica el rol de Scrum Master, y cada grupo de estudio oficia el rol de Equipo Scrum, con un líder de grupo que es el programador entusiasta. El Scrum Master es el responsable del éxito cada Equipo Scrum. Es el encargado de facilitar todos los elementos procedimentales que requiera el equipo, como así también, alentar sobre las buenas prácticas actitudinales del grupo. El profesor observa y recopila datos de forma sistemática sobre cada grupo mientras trabaja. Cuando es necesario, interviene para ayudar a los estudiantes a completar la tarea con precisión y a trabajar en conjunto de manera eficaz. El líder de equipo puede solicitar la asistencia en cualquier momento del Scrum Master, ante una situación de contingencia que no pueda ser controlado por el grupo de estudio. El Product Owner se reúne con el Scrum Master al inicio y final de cada módulo del proyecto, para evaluar la marcha metodológica e instrumentar cambios adaptativos de ser necesario. Todas las acciones evaluativas se reflejan, por un lado, en la rúbrica del docente llenado por el responsable de la práctica (Scrum Master), y por el otro, en la rúbrica del alumno, llenado por el Líder de Equipo. Al final del período de desarrollo del módulo en estudio, se realiza una puesta en común, donde cada equipo de trabajo expone sus resultados, logros e inconvenientes al grupo clase. La clase evalúa la efectividad del equipo en el trabajo conjunto y plasma sus reflexiones en una rúbrica de evaluación entre pares.

Al término del desarrollo de tres o cuatro módulos componentes del proyecto áulico, *se realiza una evaluación formativa o de proceso*. La actividad permite conocer y valorar los niveles de avance en el desarrollo de las competencias. Tiene por objeto realizar “sobre la marcha” ajustes adecuados y pertinentes a las prácticas pedagógicas, readaptar las actividades o simplemente mejorar la aplicabilidad de la metodología. Se diseñan rúbricas en función de las expectativas de logros que se buscan alcanzar. La rúbrica es realizada por el profesorado y por cada grupo de trabajo involucrado en el proyecto.

Finalmente, al concluir el ciclo de cursado de la asignatura se *realiza la evaluación sumativa o final*, cuyo objetivo central es la valoración del logro de los aprendizajes. La evaluación se lleva a cabo a partir de las evidencias reunidas a lo largo del desarrollo del proyecto didáctico. Se considera el resultado o producto final de cada módulo desarrollado, y su integración en el proyecto informático final. Así también, determina el grado en el que se han alcanzado los objetivos educativos y las competencias adquiridas en el entrenamiento de la programación. Una vez concluido el análisis y revisión de las rúbricas por parte del profesorado, se proponen actividades para favorecer el proceso de reflexión en el alumno acerca de lo que aprendió y cómo lo aprendió, para ayudarle a relacionar los nuevos aprendizajes con otros, y para valorar los logros alcanzados con referencia a la situación de inicio y a la final. Cómo última acción del docente, se traduce la evaluación cualitativa en cuantitativa con la que se determina si el alumno ha alcanzado los objetivos de la asignatura.

Además de la evaluación y reflexión sobre el proceso de aprendizaje y consolidación de saberes en los alumnos, también es *importante evaluar el impacto del nuevo proceso metodológico aplicado*. Las variables que influyen fuertemente en los perfiles psico-cognitivos de los alumnos son numerosas, al igual que variados, y difíciles de atender a todas en cada año en que se dicte la asignatura. La falta de interés por la superación académica, falta de compromiso para asumir el rol de un estudiante activo, dificultad para expresarse y hacerse entender son sólo algunos de los grandes inconvenientes difíciles de revertir en el proceso educativo, y que repercuten en la aplicación de cualquier estrategia metodológica. Por ello, con esta tarea se busca disponer de acciones objetivas, que permita evaluar el esfuerzo docente aplicado en todas las etapas del proceso metodológico, por un lado, y el rendimiento académico de los estudiantes, por el otro. Se tiene en cuenta las desviaciones de la predicción académica en la adecuación versus la realidad enfrentada, y la velocidad de reacción en la adaptación dinámica de la propuesta metodológica ante las necesidades pedagógicas requeridas por los alumnos del momento.

Es importante resaltar la necesidad de evaluar la efectividad de la aplicación de la nueva metodología para las prácticas académicas. El objetivo es contar en cada período lectivo, con estadísticas de variables como: esfuerzo docente vs cantidad de alumnos, rendimiento académico, porcentaje de abandono, metas alcanzadas, objetivos académicos cumplidos, entre otros. Esto permitirá realizar predicciones y adecuaciones pertinentes en la aplicación de la metodología en el año siguiente.

5. Resultados

La nueva metodología fue puesta en práctica durante el año 2019 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 algún lenguaje para computadoras, y un 20% con ejercicios de consolidación de conceptos.

Durante los años de 2014 y 2015, 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 comparados con los de años previos, en el que se usó enseñanza tradicional basadas en clases magistrales [Barberis and Del Moral, 2016]. 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 [Del Moral Sachetti and Barberis, 2016]. Los resultados de la evaluación metodológica fueron muy alentadores. El experimento se repitió durante el año 2016.

Uno de los resultados concluyentes del experimento de 2016, fue que el plantel docente dedicaba gran esfuerzo para el éxito de la estrategia metodológica ágil como

consecuencia de una capacitación débil en el desarrollo ágil de 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 2017 y 2018 no se repitieron los experimentos en las clases prácticas, retornando al tradicional sistema de enseñanza basado en clases magistrales. Durante esos años, el plantel docente que conduce la investigación se capacitó plenamente en el desarrollo ágil con Scrum, y se perfilaron la actual propuesta pedagógica que se puso en práctica durante el año 2019. En el año 2020 no se aplicó la metodología como consecuencia de la pandemia que hasta hoy 2021 sigue afectando a todo el mundo.

Tabla 1. Indicadores de mayor impacto obtenidos en la evaluación diagnóstica.

Indicador	Cantidad de alumnos	Porcentaje
Alumnos Inscriptos	79	100 %
Alumnos Evaluados	74	96 %
Resolución correcta de problemas (2 categorías: Numéricos y de programación)		
Sólo Numéricos Básicos	27	36.5 %
Sólo De programación	14	18.92 %
Numéricos Básicos y De programación	8	10.8 %
Resolución incorrecta / No resueltos	25	33.78 %
Recursos utilizados por los alumnos en la programación		
Únicamente en papel	31	41.89 %
Únicamente en computadora	20	27.03 %
Únicamente en algún dispositivo móvil	11	14.86 %
Computadora y dispositivos móvil	5	6.76 %
Computadora, dispositivo móvil y en papel	7	9.46 %
Conocimiento sobre las etapas de resolución de problemas de programación.	22	29.73 %
Formas en que los alumnos estudian		
Únicamente en forma individual	39	52.7 %
Únicamente en grupo	14	18.92 %
Individual y en grupo	21	28.38 %

La innovación metodológica puesto en práctica en el año 2019 se inició con una evaluación diagnóstica inicial. Entre los resultados obtenidos, se muestran en la tabla 1 los indicadores de mayor impacto. En general, el diagnóstico resaltó que los alumnos tenían muy poca habilidad, tanto en la programación como en la resolución de problemas numéricos. Sobre la base de 74 estudiantes evaluados, 27 alumnos (36.5%) resolvieron correcta y únicamente los problemas numéricos. Si a estos estudiantes se le adicionan los 25 que no resolvieron ningunos de los problemas (numéricos y de programación) o los hicieron incorrectamente, totalizan 52, que equivalen al 70.27 % de aquellos que no lograron demostrar sus habilidades en programación. Del 29.73 % (22

estudiantes) que sí lo hicieron, sólo un alumno podía considerarse un Programador entusiasta.

Un indicador importante en la nueva metodología de entrenamiento en la programación, es el que responde a la pregunta de qué recurso utilizaron cuando aprendían a programar y cuando se les enseñaba un lenguaje de programación. Sobre el total de evaluados, el 41.89 % lo hizo únicamente usando escritura en papel. Esta estadística es muy desalentadora para la aplicación de la propuesta metodológica, ya que un alto porcentaje de alumnos no adquirieron ninguna habilidad requerida para la programación de computadoras. De igual forma ocurre con el 31.08 % (14.86 + 6.76 + 9.46) de los estudiantes que adquirieron algunas habilidades, y de forma no adecuada, al usar un dispositivo móvil. A este panorama crítico, se le adiciona el hecho de que sólo un 29.73% de los discentes conoce las distintas etapas que se considera en la resolución de problemas informáticos con programación.

Dado que la propuesta metodológica se sustenta en el trabajo en equipo, con estrategias de aprendizajes colaborativos y cooperativos, fue importante conocer la forma en que los alumnos estudian fuera del ámbito de la clase. El diagnóstico mostró que el 52.7 % estudiaba o prefería desarrollar su formación de manera individual.

Estas estadísticas, al igual que otras de menor impacto, mostraron una realidad educativa muy crítica, que el equipo docente debía revertir con la nueva metodología de entrenamiento en la programación. Las actividades de adecuación, diseño, preparación, y puesta en marcha de la nueva estrategia constituyó un gran desafío para el cuerpo docente. La etapa de implementación inició con 6 clases (dos primeras semanas) de nivelación académica. Paralelamente se instruyó en el rol de líder de equipo a los alumnos más destacados en la evaluación diagnóstica para la conformación de los equipos de estudios. A partir de entonces se continuó con las actividades programadas para la metodología. Las expectativas de logros se adecuaron para evaluar el 70% de las actividades en grupo, y el 30% restantes a las individuales. La clase se dividió inicialmente en 9 grupos de 8 integrantes cada uno, y un grupo de 7. Al término del ciclo de cursado se tenía 8 grupos de 8 integrantes y 2 grupos de 6, por el abandono de 3 alumnos.

Se planificaron tres proyectos informáticos diferentes: 1. Aplicación para dispositivos móviles que asista al estudiante en todos los temas relacionados al cálculo numérico, 2. Aplicación en computadora para el docente, que le permita diseñar problemas numéricos con comprobación de resultados, y 3. Aplicación en computadora para ingenieros en carreteras. Los tres proyectos se estructuraban de 8 módulos a ser desarrollados durante el cursado de la asignatura. Cada módulo implicaba el uso incremental de los contenidos del currículo de la asignatura.

Además de las rúbricas llenadas al término de cada módulo, la estrategia evaluativa consistió de dos evaluaciones formativas (al término de los primeros cuatro módulos, y al final de los cuatros restantes), y de una sumativa al final del cuatrimestre. La primera fue instrumentada con problemas numéricos básicos y de complejidad intermedia. Esta modalidad de evaluación con exigencia intermedia permitió a los estudiantes reflexionar sobre la importancia de realizar autoevaluaciones al final de cada módulo del proyecto. Los resultados mostraron que el 90% de los alumnos que asistían regularmente a clases aprobaron el examen, mientras que el 10% restante fue a

recuperar. En el recuperatorio aprobaron el 100% de los examinados. Así, la estrategia de evaluación aplicada generó en los alumnos confianza, seguridad en sí mismo y adquirieron conciencia de la importancia de autoevaluarse en el estudio progresivo.

La segunda evaluación formativa incluyó problemas de la vida real con una complejidad media, y ejercicios de aplicación del razonamiento crítico y conceptual. Esta vez, el 67% de los estudiantes aprobaron el examen, mientras que el 33% fue a recuperar. En el recuperatorio aprobaron el 60% de los examinados. Luego, la evaluación sumativa fue aprobado por el 100% de los participantes (alumnos que aprobaron las dos evaluaciones formativas o sus respectivas recuperaciones). Estadísticamente, 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 2). La forma de evaluar (de menor rigor a mayor), y autoevaluarse de manera permanente, presupone que el alumno asume un papel más activo en el proceso de aprendizaje, y que la propia evaluación es una instancia de aprendizaje adicional en la consecución de las competencias genéricas y profesionales.

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 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.

Tabla 2. Registros anuales de 2008 al 2019 de alumnos matriculados en Cálculo Numérico, y las cantidades de regularizados, libres y los que abandonaron el cursado.

Año	C. Inscriptos	C. Regulares	C. Libres	C. Abandono
2008	59	34	8	17
2009	82	45	15	22
2010	66	31	9	26
2011	58	33	9	16
2012	38	18	7	13
2013	70	44	12	14
2014	74	56	14	4
2015	85	71	9	5
2016	78	67	7	4
2017	82	37	23	22
2018	67	38	9	20
2019	79	71	5	3

El cambio en la estrategia educativa en la asignatura implicó un mayor compromiso docente, sobre todo en el modo de relacionarse con los alumnos, y en las formas de preparar e impartir las clases. El enfoque centrado en el aprendizaje de los estudiantes favoreció la reestructuración cognitiva. Así, se logró mejorar la calidad cognitiva del estudiantado, sumar nuevas experiencias en la programación ágil de software, y alcanzar las competencias genéricas necesarias para la resolución de problemas numéricos. Al mismo tiempo, se contribuyó con la disminución de la tasa de abandono (figura 1) e incrementar los índices de rendimientos respecto de años anteriores (figura 2 y 3).

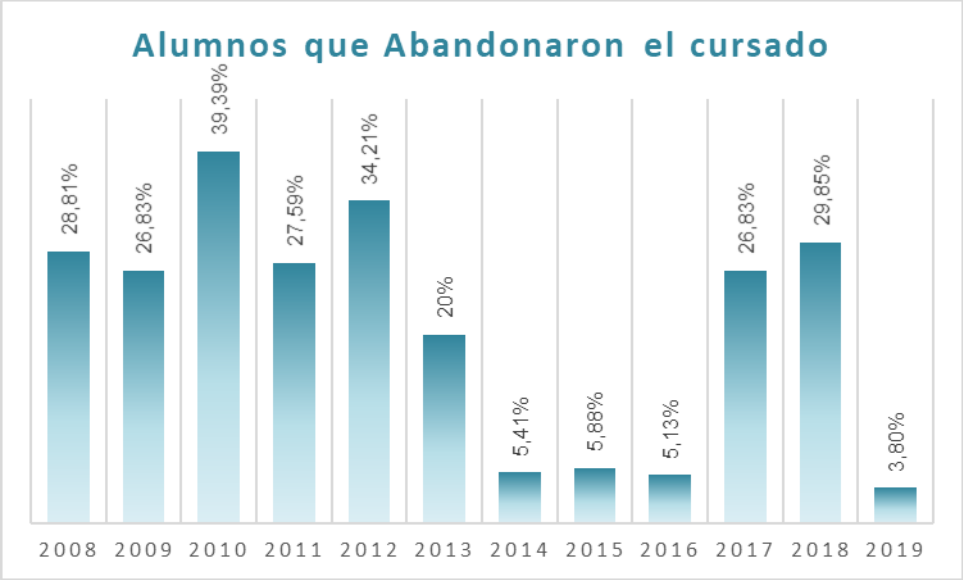

Figura 1. Curva que muestra la tendencia del índice de abandono en el cursado

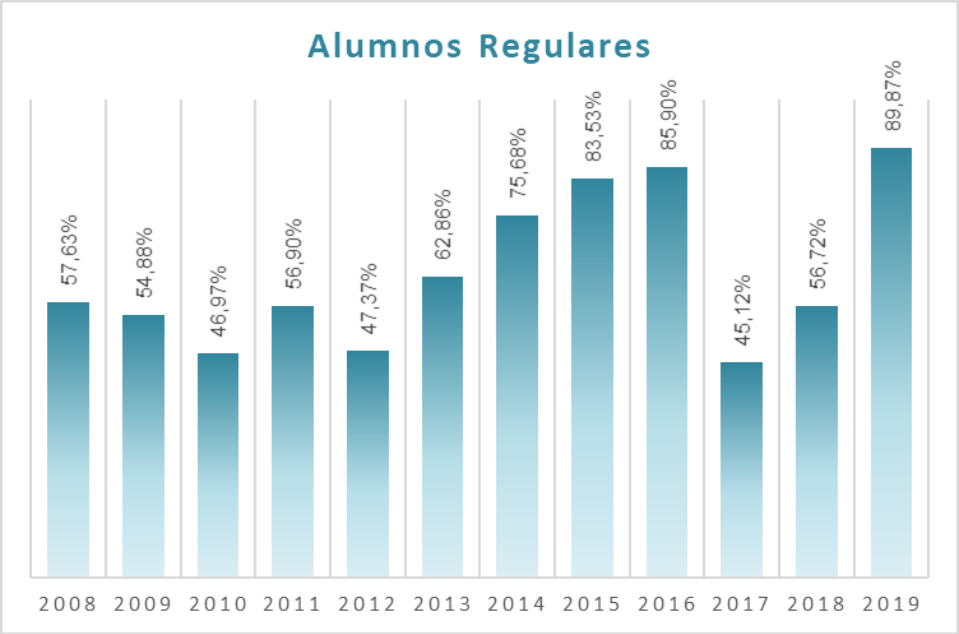

Figura 2. Curva que muestra la tendencia del porcentaje de alumnos aprobaron el cursado de la asignatura entre los años 2008 y 2019.

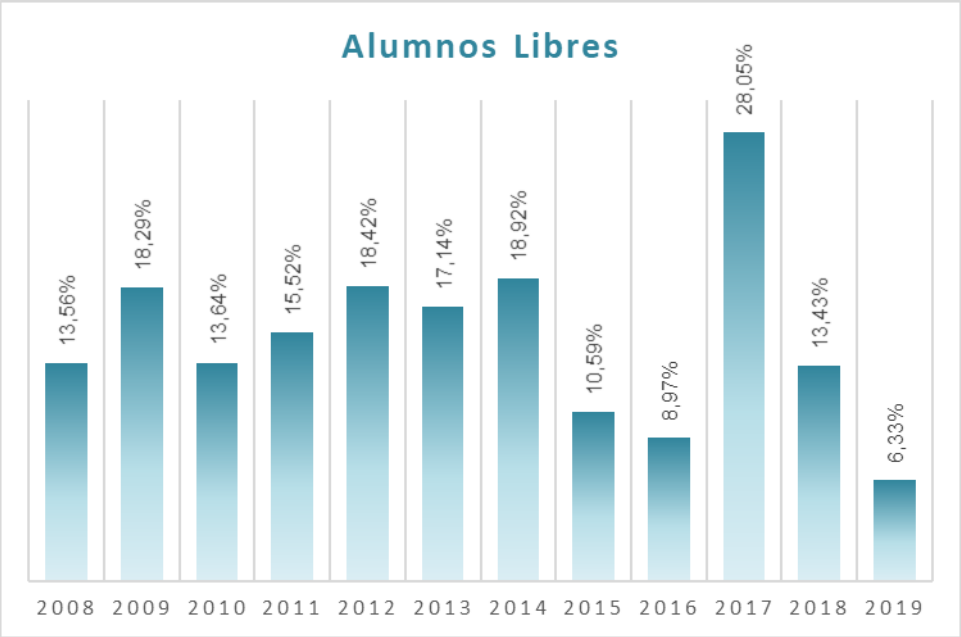

Figura 3. Curva que muestra la tendencia del porcentaje de alumnos que no aprobaron el cursado de la asignatura entre los años 2008 y 2019.

6. Conclusión

La puesta en marcha de una metodología como la descripta requiere necesariamente la capacitación previa del cuerpo docente en el desarrollo ágil de software con Scrum. Debe tener conocimientos claros sobre las estrategias de Aprendizajes basados en problemas y basados en proyectos. Dominar las actividades pedagógicas del aprendizaje colaborativo y cooperativo. La importancia de la capacitación previa del profesorado radica en la necesidad de contar con perfiles docentes que reaccionen rápidamente en la instrumentación de soluciones pedagógicas ante una situación no prevista. La velocidad con que se desarrolla la dinámica metodológica, y los tiempos acotados para el desarrollo curricular de la asignatura, hacen que, en algunas situaciones, no puedan detectarse en la mediatez posicionamientos psico-culturales de los alumnos que les impiden desarrollar habilidades sociales y colaborativas en el trabajo en equipo. Pasar por alto estos escenarios implica un mayor riesgo en el abandono del cursado por parte del protagonista. Pueden suscitarse inconvenientes mayores de marginalidad o de integración a un grupo de estudio que requieran el asesoramiento de un profesional psicopedagógico. Captar la atención del alumnado, generar en ellos la sensación de seguridad en el aprendizaje, crear espacios de reflexión y discusión de saberes, planificar actividades que induzcan al desarrollo del pensamiento crítico, son solo algunas de las tareas que demandan docentes profesionalizado, no sólo en los temas del currículo de la asignatura sino también, en lo pedagógico.

La etapa del diagnóstico e indagación es de suma importancia para el desarrollo de la propuesta metodológica. Esta brinda la información suficiente para planificar la estrategia que eleva la calidad educativa. Las características adaptativa y contextualizada de la etapa de diseño aseguran el éxito de las etapas siguientes. La velocidad de reacción

y de cambios adaptativos en la marcha de la etapa de implementación, favorecen el desarrollo académico de la realidad laboral en el contexto áulico. Este escenario entrena al alumno en la toma de decisión y la resolución de conflictos. Lo que potencia el trabajo en quipo cuando se desarrolla aplicaciones informáticas.

La disponibilidad de criterios que permiten crear evaluaciones como una instancia de aprendizaje para el alumnado, no sólo ayuda a mejorar el rendimiento académico de ellos, sino también, a elevar la calidad de los aprendizajes. Por ello, en la etapa de evaluación y reflexión se destaca la necesidad de una labor concienzuda del docente profesionalizado.

En términos generales, la acción metodológica facilitó la adquisición en los alumnos de una correcta conceptualización de los contenidos específicos de la asignatura. Como así también, el desarrollo de habilidades profesionales en la programación. La estrategia logró que los alumnos tímidos, introvertidos o pasivos sean más participativos. Al integrarse a un equipo de trabajo y sentirse involucrados en el desarrollo de un proyecto de utilidad real, generó entusiasmo por el estudio y por la programación, haciendo que la interacción con sus pares resultara estimuladora.

El trabajo en grupo cooperativo, facilitó la detección de diferentes habilidades en los alumnos. Por ejemplo, algunos mostraron más predisposición a realizar pruebas de software, a diseñar casos de estudios, y otros a integrar componentes, etc. Esto ayudó a los discentes a descubrir su propio perfil, teniendo en cuenta habilidades y capacidades propias. Con el entrenamiento en el desarrollo ágil de la programación, los estudiantes pudieron desarrollar aún más el pensamiento crítico y computacional, y concientizarse de las competencias débiles que debían fortalecer.

La metodología fomentó el trabajo en equipo de los estudiantes acercando la realidad laboral a las aulas, debido a que la producción de software no es una tarea solitaria sino una actividad desarrollada en equipos multidisciplinarios.

Finalmente, entre los resultados obtenidos resaltan la mejora en el rendimiento académico de los estudiantes y la disminución de la tasa de abandono en el cursado.

7. Trabajo Futuro

La metodología propuesta en este artículo es de utilidad en las prácticas académicas de cualquier asignatura que use la programación como una componente fuerte en la praxis. Los enfoques metodológicos que guían el desarrollo de las clases teóricas de una asignatura, pueden ser diferentes a los aplicados en la praxis académica. En este sentido, las investigaciones conducidas por los autores están orientadas hacia el desarrollo de una estrategia para la enseñanza de conceptos teóricos que sea compatible con la expuesta en este artículo. Se busca una estrategia basada en metodologías activas que permita desarrollar clases teóricas participativas con énfasis en la programación, y en el desarrollo ágil con Scrum. Se prevé que la propuesta siga aplicándose en cuanto se autorice la vuelta a la enseñanza presencial en el ámbito universitario. La aplicación en años sucesivos permitirá perfeccionar las actividades y etapas que conforman la metodología.

8. Referencias

Ackoff, R. L. and Greenberg, D. (2008). Turning Learning Right Side Up Putting Education Back on Track. United States, Pearson Education, Inc. 1st Ed.

Alaimo, M. (2015). Proyectos Ágiles con Scrum. Buenos Aires, Argentina, Kleer Agile Coaching. 2da. Ed.

Alfonso Sánchez, I. R. (2016). "La Sociedad de la Información, Sociedad del Conocimiento y Sociedad del Aprendizaje. Referentes en torno a su formación." *bibliotecas anales de investigación,* 12(2), 231-239. http://revistas.bnjm.cu/index.php/BAI/article/view/179/189

Back, K. and Andres, C. (2005). Extreme Programming Explained: embrace change. Boston, MA, Addison-Wesley Professional. 2da. Ed.

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". XI Congreso de Tecnología en Educación y Educación en Tecnología 2016 (TE&ET 2016), Universidad de Morón, Argentina. http://sedici.unlp.edu.ar/handle/10915/54603

Barberis, A. R., Del Moral Sachetti, L. E., Silvera, J. A., Méndez, E. and Rojas, N. (2021). "Rediseño Educativo para la Enseñanza y Aprendizaje del Cálculo Numérico". XXIII Edición del Workshop de Investigadores en Ciencias de la Computación (WICC), Universidad Nacional de Chilecito (UNdeC). La Rioja, Argentina. http://sedici.unlp.edu.ar/handle/10915/120524

Bawamohiddin, A. B. and Razali, R. (2017). "Problem-based Learning for Programming Education." *International Journal on Advanced Science, Engineering and Information Technology,* 7(6), 2035-2050. doi: 10.18517/ijaseit.7.6.2232

Bennedsen, J. and Caspersen, M. E. (2008). Exposing the Programming Process. In Reflections on the Teaching of Programming: Methods and Implementations. J. Bennedsen, M. E. Caspersen and M. Kölling, páginas: 6-16. Berlin, Heidelberg, Springer Berlin Heidelberg.

Bernal González, M. d. C. and Martínez Dueñas, M. (2017). "Metodologías Activas para la Enseñanza y el Aprendizaje." *Revista Panamericana de Pedagogía,* 0(25), 101-106. https://revistas.up.edu.mx/RPP/article/view/1695

Braught, G., Eby, L. M. and Wahls, T. (2008). "The effects of pair-programming on individual programming skill." *ACM SIGCSE Bulletin,* 40(1), 200-204. doi: 10.1145/1352322.1352207

Del Moral Sachetti, L. and Barberis, Á. R. (2016). "Scrum como herramienta metodológica en el entrenamiento cooperativo de la programación: proceso evaluativo". XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016) - V Workshop de Innovación en Educación en Informática (WIEI), Universidad de San Luis, Argentina. http://sedici.unlp.edu.ar/handle/10915/56299

Fernández March, A. (2006). "Metodologías activas para la formación de competencias." *Educatio Siglo XXI,* 24(0), 35-56. https://revistas.um.es/educatio/article/view/152

Fuertes, Y. and Sepúlveda, J. (2016). "Scrum, Kanban and Canvas in the commercial, industrial and educational sector - A literature review." *Published in Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS),* 6(1), 46-50.

García, R., Traver, J. A. and Candela, I. (2019). Aprendizaje cooperativo: Fundamentos, características y técnicas. Madrid, España, ICCE (Instituto Calasanz de Ciencias de la Educación). 2da. Ed.

Godoy, C. P., Cruz, A. F., Silva, E. P., Santos, L. M., Zerbini, R. S. and Pahins, C. A. L. (2019). "Blueprint Model: A new Approach to Scrum Agile Methodology". 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE). doi: 10.1109/ICGSE.2019.00014

Gomes, A. and Mendes, A. (2014). "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations". 2014 IEEE Frontiers in Education Conference (FIE), Madrid, España. doi: 10.1109/FIE.2014.7044086

Gómez, S. M. (2020). "Aplicación de las Metodologías Ágiles al proceso de enseñanza-aprendizaje universitario." *Revista d'Innovació Docent Universitària,* 2020(12), 62-73. doi: 10.1344/RIDU2020.12.7

Havenga, M., Breed, B., Mentz, E., Govender, D., Govender, I., Dignum, F. and Dignum, V. (2013). "Metacognitive and problem-solving skills to promote self-directed learning in computer programming: Teachers' experiences." *SA-eDUC Journal,* 10(2), 1-14.

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

Johnson, D. W., Johnson, R. T. and Smith, K. A.; (1991). "*Cooperative learning: Increasing college faculty instructional productivity*". **Report** 4. ASHE-ERIC Higher Education. The George Washington University. Washington, DC. https://eric.ed.gov/?id=ED343465

Kofune, Y. and Koita, T. (2014). "Empirical Studies of Agile Software Development to Learn Programming Skills." *Systemics, Cybernetics and Informatics,* 12(3), 34-37. http://www.iiisci.org/journal/CV%24/sci/pdfs/HB618EY14.pdf

Lasa Gómez, C. and Alvarez García, A. (2018). Métodos Ágiles: Scrum, Kanban, Lean. Madrid, España, Anaya Multimedia. 2da. Ed.

McKenna, D. (2016). The Art of Scrum. Berkeley, CA, Apress. 1st Ed.

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,* 18(4), 247-260. doi: 10.1080/08993400802461396

Meriño Almaguer, Y., Lorente Rodríguez, A. E. and Maribona, M. G. (2011). "Propuesta de instrumentos de evaluación para entornos virtuales de aprendizaje: una experiencia en la universidad de las ciencias informáticas." *Revista de Informática Educativa y Medios Audiovisuales,* 8(15), 1-8.

Onieva López, J. L. (2018). "Scrum como Estrategia para el Aprendizaje Colaborativo a través de Proyectos. Propuesta Didáctica para sus Implementación en el Aula Universitaria." *Redes y colaboración en educación: Nuevas formas de participación y transformación social, Colaboración,* 22(12), 509-527. doi: 10.30827/profesorado.v22i2.7735

Palacio, J. (2007). Flexibilidad con Scrum. *Principios de diseño e implantación de campos de Scrum*. (1º Ed.) np. 190. Navegápolis.com. http://www.scrummanager.net/files/flexibilidad_con_scrum.pdf Accedido: 29/07/2020

Peralta Lara, D. C. and Guamán Gómez, V. J. (2020). "Metodologías activas para la enseñanza y aprendizaje de los estudios sociales." *Sociedad & Tecnología,* 3(2), 2-10. doi: 10.51247/st.v3i2.62

Pérez Gómez, Á. I. (2010). "Nuevas exigencias y escenarios para la profesión docente en la era de la información y de la incertidumbre." *Revista Interuniversitaria de Formación del Profesorado,* 24(2), 17-36. https://www.redalyc.org/articulo.oa?id=27419198002

Pressman, R. S. and Maxim, B. R. (2015). Software Engineering: A Practitioner's Approach. New York, McGraw-Hill Education. 8va. Ed.

Robins, A. V. (2019). Novice Programmers and Introductory Programming. In The Cambridge Handbook of Computing Education Research. A. V. Robins and S. A. Fincher, páginas: 327-376. Cambridge, Cambridge University Press.

Silva Quiroz, J. and Maturana Castillo, D. (2017). "Una propuesta de modelo para introducir metodologías activas en educación superior." *Innovación educativa (México, DF),* 17(73), 117-131. http://www.scielo.org.mx/scielo.php?pid=S1665-26732017000100117&script=sci_abstract

Wijnands, W. and Stolze, A. (2019). Transforming Education with eduScrum. In Agile and Lean Concepts for Teaching and Learning: Bringing Methodologies from Industry to the Classroom. D. Parsons and K. MacCallum, páginas: 95-114. Singapore, Springer Singapore.

Williams, L. (2011). Pair programming. Taylor and Francis Group., In Phillip A. Laplante, editor. Encyclopedia of Software Engineering. Volume II.

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,* 12(3), 197-212. doi: 10.1076/csed.12.3.197.8618

Wood, D. F. (2003). "Problem based learning." *British Medical Journal (BMJ),* 326(7384), 328-330. doi: 10.1136/bmj.326.7384.328

Yew, E. H. J. and Goh, K. (2016). "Problem-Based Learning: An Overview of its Process and Impact on Learning." *Health Professions Education,* 2(2), 75-79. doi: 10.1016/j.hpe.2016.01.004