https://doi.org/10.69639/arandu.v13i1.2108

Metodologías activas para la formación de ingenieros de software en la educación superior

Active Methodologies for the Training of Software Engineers in Higher Education

Maria Teodolinda Ortega Ovalle
maria.ortegao@up.ac.pa
https://orcid.org/0009-0000-3629-9751
Universidad de Panamá
Panamá

Artículo recibido: 18 febrero 2026-Aceptado para publicación: 20 marzo 2026
Conflictos de Interés: Ninguno que declarar

RESUMEN

La formación de ingenieros de software en la educación superior exige enfoques pedagógicos que promuevan el aprendizaje significativo, la resolución de problemas y el desarrollo de competencias profesionales alineadas con las demandas de la industria tecnológica. Las metodologías activas se han consolidado como estrategias efectivas para fomentar la participación estudiantil, el pensamiento crítico y la aplicación práctica del conocimiento en contextos reales. Este artículo analiza las principales metodologías activas utilizadas en la enseñanza de la Ingeniería de Software, sus fundamentos teóricos, su impacto en el aprendizaje y los desafíos asociados con su implementación en entornos presenciales, híbridos y virtuales. A partir de una revisión analítica de la literatura reciente, se identifican oportunidades para fortalecer la formación profesional mediante enfoques centrados en el estudiante, el trabajo colaborativo y la integración de herramientas tecnológicas avanzadas.

Palabras clave: metodologías activas, ingeniería de software, educación superior, aprendizaje colaborativo, aprendizaje basado en proyectos

ABSTRACT

The training of software engineers in higher education requires pedagogical approaches that promote meaningful learning, problem-solving, and the development of professional competencies aligned with the demands of the technology industry. Active methodologies have emerged as effective strategies to foster student engagement, critical thinking, and the practical application of knowledge in real-world contexts. This article analyzes the main active methodologies used in Software Engineering education, their theoretical foundations, their impact on learning, and the challenges associated with their implementation in face-to-face, hybrid, and virtual environments. Based on an analytical review of recent literature, opportunities are

identified to strengthen professional training through student-centered approaches, collaborative work, and the integration of advanced technological tools.

Keywords: active methodologies, software engineering, higher education, collaborative learning, project-based learning

INTRODUCCIÓN

La formación de ingenieros de software en la educación superior enfrenta un escenario de transformación constante, impulsado por la acelerada evolución tecnológica, la globalización del trabajo y la creciente complejidad de los sistemas informáticos. En este contexto, las instituciones educativas se ven obligadas a replantear sus modelos pedagógicos para responder a las demandas de una industria que requiere profesionales capaces de resolver problemas reales, trabajar en equipos multidisciplinarios, adaptarse a entornos cambiantes y aprender de manera autónoma a lo largo de su vida profesional. Las metodologías tradicionales, centradas en la transmisión unidireccional de contenidos, han demostrado ser insuficientes para desarrollar las competencias técnicas, cognitivas y socioemocionales que exige el campo de la Ingeniería de Software.

En respuesta a estas necesidades, las metodologías activas han emergido como enfoques pedagógicos que sitúan al estudiante en el centro del proceso de aprendizaje, promoviendo su participación activa, la reflexión crítica, la experimentación y la aplicación práctica del conocimiento. Estas metodologías, entre las que destacan el Aprendizaje Basado en Proyectos (ABP), el Aprendizaje Basado en Problemas (PBL), el Aprendizaje Colaborativo, el Aprendizaje Experiencial y las simulaciones profesionales, se alinean con los principios del constructivismo, el aprendizaje situado y el aprendizaje experiencial, teorías que reconocen que el conocimiento se construye a partir de la interacción con el entorno y la resolución de problemas significativos.

En el ámbito de la Ingeniería de Software, estas metodologías adquieren una relevancia particular debido a la naturaleza iterativa, colaborativa y orientada a la solución de problemas que caracteriza al desarrollo de software. La ingeniería de requisitos, el diseño de arquitecturas, la programación, las pruebas, la gestión de proyectos y la integración continua son procesos que requieren no solo conocimientos técnicos, sino también habilidades de comunicación, liderazgo, pensamiento crítico y trabajo en equipo. Por ello, la implementación de metodologías activas permite simular escenarios reales de la industria, fomentar la autonomía del estudiante y fortalecer competencias transversales esenciales para el ejercicio profesional.

Además, la creciente digitalización de la educación superior y la expansión de los entornos virtuales e híbridos han abierto nuevas oportunidades para integrar tecnologías que potencien las metodologías activas. Plataformas colaborativas, repositorios de código, entornos de desarrollo en la nube, simuladores de proyectos ágiles y herramientas de comunicación sincrónica y asincrónica permiten diseñar experiencias de aprendizaje más flexibles, personalizadas y orientadas a la práctica profesional. Sin embargo, también plantean desafíos relacionados con la capacitación docente, la disponibilidad de recursos tecnológicos, la gestión del tiempo académico y la necesidad de garantizar la participación activa de los estudiantes.

En este artículo se analiza de manera exhaustiva el papel de las metodologías activas en la formación de ingenieros de software en la educación superior, destacando sus fundamentos teóricos, sus beneficios, sus limitaciones y las oportunidades que ofrecen para mejorar la calidad del aprendizaje. A partir de una revisión analítica de la literatura reciente, se examina su aplicación en diferentes modalidades educativas y se proponen recomendaciones para su implementación efectiva en programas de Ingeniería de Software.

METODOLOGÍA

El presente estudio se desarrolló mediante una revisión analítica y narrativa de literatura especializada, con el propósito de identificar, describir y examinar las metodologías activas aplicadas a la formación de ingenieros de software en la educación superior. Se adoptó un enfoque cualitativo de carácter interpretativo, orientado a comprender las tendencias, prácticas y resultados reportados en investigaciones recientes, así como los desafíos y oportunidades que estas metodologías presentan en contextos presenciales, híbridos y virtuales.

Para la recopilación de información se consultaron diversas bases de datos académicas de alto impacto, entre ellas Scopus, Web of Science, IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect y Google Scholar. Estas fuentes fueron seleccionadas debido a su relevancia en el ámbito de la ingeniería, la tecnología educativa y la educación superior. La búsqueda se centró en estudios publicados entre 2018 y 2024, escritos en inglés o español, revisados por pares y relacionados específicamente con la implementación de metodologías activas en programas de Ingeniería de Software o disciplinas afines. Se priorizaron investigaciones que evaluaran el impacto de estas metodologías en el aprendizaje, el desarrollo de competencias profesionales, la participación estudiantil y la experiencia formativa en general.

El proceso de selección de estudios se llevó a cabo mediante la identificación de palabras clave como active learning, software engineering education, project-based learning, problem-based learning, collaborative learning, experiential learning, simulation y higher education. A partir de estas búsquedas se realizó una primera depuración basada en la pertinencia temática y la disponibilidad de información completa. Posteriormente, se efectuó una lectura detallada de los estudios seleccionados para determinar su relevancia y consistencia metodológica.

El análisis de la información se desarrolló en varias etapas. En primer lugar, se realizó una lectura exploratoria para identificar patrones generales y enfoques recurrentes en la literatura. Luego, se procedió a una codificación temática que permitió organizar los hallazgos en categorías conceptuales relacionadas con los tipos de metodologías activas, las competencias que promueven, los resultados obtenidos en diferentes contextos educativos, los desafíos de implementación y las herramientas tecnológicas que las acompañan. Finalmente, se integraron los hallazgos mediante un proceso de síntesis y triangulación conceptual, lo que permitió

contrastar resultados, identificar tendencias emergentes y reconocer vacíos en la literatura que requieren mayor investigación.

El alcance de este estudio se centra en ofrecer una visión amplia, actualizada y fundamentada sobre el uso de metodologías activas en la formación de ingenieros de software. No obstante, se reconocen ciertas limitaciones inherentes a los estudios de revisión, como la dependencia de la disponibilidad de literatura reciente, la variabilidad en los enfoques metodológicos de los estudios analizados y la imposibilidad de generalizar completamente los resultados debido a la diversidad de contextos educativos. A pesar de ello, la revisión realizada proporciona una base sólida para comprender el estado actual de las metodologías activas en la educación superior y su potencial para transformar la enseñanza de la Ingeniería de Software.

Marco teórico

La formación de ingenieros de software en la educación superior se sustenta en un conjunto de teorías pedagógicas que han evolucionado a lo largo del tiempo y que hoy constituyen la base de los modelos educativos centrados en el estudiante. Las metodologías activas, como el Aprendizaje Basado en Proyectos, el Aprendizaje Basado en Problemas, el Aprendizaje Colaborativo, el Aprendizaje Experiencial y las simulaciones profesionales, encuentran su fundamento en corrientes teóricas que conciben el aprendizaje como un proceso dinámico, constructivo y situado. Estas teorías no solo explican cómo aprenden los estudiantes, sino que también orientan el diseño de experiencias formativas que buscan desarrollar competencias complejas, especialmente relevantes en disciplinas como la Ingeniería de Software.

El constructivismo, una de las bases más influyentes, sostiene que el conocimiento no se transmite de manera pasiva, sino que se construye activamente a partir de la interacción del individuo con su entorno. Piaget enfatiza la importancia de la actividad mental del estudiante, quien reorganiza y reinterpreta la información para generar nuevos esquemas cognitivos. En el contexto de la Ingeniería de Software, esta perspectiva se refleja en actividades que requieren que los estudiantes analicen problemas reales, diseñen soluciones, experimenten con tecnologías y reflexionen sobre sus decisiones. La construcción de software implica procesos iterativos de análisis, diseño, implementación y evaluación, lo que se alinea de manera natural con los principios constructivistas.

Por su parte, el enfoque socioconstructivista de Vygotsky destaca el papel de la interacción social y el lenguaje en la construcción del conocimiento. La noción de Zona de Desarrollo Próximo (ZDP) es especialmente relevante para comprender cómo los estudiantes pueden alcanzar niveles superiores de desempeño mediante la colaboración con compañeros y la guía de docentes o tutores. En la Ingeniería de Software, donde el trabajo en equipo es esencial, esta teoría respalda la implementación de metodologías que promuevan la comunicación, la negociación, la co-construcción de soluciones y la resolución conjunta de problemas. Las actividades colaborativas, los proyectos grupales y las simulaciones de equipos de desarrollo

reflejan este enfoque, permitiendo que los estudiantes aprendan no solo de los contenidos, sino también de las interacciones con otros.

El aprendizaje situado, propuesto por Lave y Wenger, plantea que el conocimiento se adquiere de manera más efectiva cuando se desarrolla en contextos auténticos y significativos. Según esta perspectiva, aprender implica participar en comunidades de práctica donde los individuos se involucran en actividades reales y progresivamente adquieren mayor responsabilidad. En la formación de ingenieros de software, esta teoría se materializa en experiencias que simulan entornos profesionales, como proyectos ágiles, prácticas en empresas, laboratorios de desarrollo y actividades que replican procesos reales del ciclo de vida del software. Estas experiencias permiten que los estudiantes comprendan no solo los aspectos técnicos, sino también las dinámicas sociales, organizacionales y éticas del trabajo profesional.

El aprendizaje experiencial, basado en el modelo de Kolb, propone que el aprendizaje es un ciclo continuo que integra la experiencia concreta, la observación reflexiva, la conceptualización abstracta y la experimentación activa. Este enfoque es especialmente pertinente para la Ingeniería de Software, donde los estudiantes deben enfrentarse a problemas reales, reflexionar sobre sus decisiones, comprender los fundamentos teóricos que sustentan sus acciones y aplicar nuevamente lo aprendido en nuevas situaciones. Las metodologías activas que incorporan prácticas, laboratorios, simulaciones y proyectos permiten recorrer este ciclo de manera natural, fortaleciendo la comprensión profunda y la transferencia del conocimiento.

El aprendizaje colaborativo constituye otro pilar fundamental del marco teórico. Esta perspectiva sostiene que el aprendizaje se potencia cuando los estudiantes trabajan juntos para alcanzar objetivos comunes, compartiendo conocimientos, habilidades y responsabilidades. En la Ingeniería de Software, donde la mayoría de los proyectos se desarrollan en equipos multidisciplinarios, esta teoría adquiere una relevancia especial. Las metodologías activas que promueven el trabajo en equipo permiten desarrollar competencias como la comunicación efectiva, la gestión de conflictos, la toma de decisiones conjunta y la responsabilidad compartida, todas ellas esenciales para el ejercicio profesional.

Finalmente, las teorías del aprendizaje autorregulado y del aprendizaje autónomo complementan este marco teórico al destacar la importancia de que los estudiantes gestionen su propio proceso de aprendizaje. En un campo tan dinámico como la Ingeniería de Software, donde las tecnologías evolucionan rápidamente, la capacidad de aprender de manera independiente, identificar necesidades formativas, planificar estrategias y evaluar el propio desempeño es crucial. Las metodologías activas fomentan estas habilidades al situar al estudiante como protagonista de su aprendizaje, brindándole oportunidades para tomar decisiones, asumir responsabilidades y reflexionar sobre su progreso.

En conjunto, estas teorías proporcionan un marco sólido para comprender por qué las metodologías activas son especialmente adecuadas para la formación de ingenieros de software.

Su énfasis en la participación activa, la colaboración, la reflexión, la autenticidad de las tareas y la integración entre teoría y práctica responde a las demandas de una disciplina que requiere profesionales capaces de enfrentar problemas complejos, adaptarse a entornos cambiantes y trabajar eficazmente en equipos. Además, la incorporación de tecnologías educativas y entornos virtuales amplía las posibilidades de implementar estas metodologías, permitiendo diseñar experiencias de aprendizaje más flexibles, personalizadas y alineadas con las prácticas de la industria del software.

RESULTADOS

El análisis de la literatura permitió identificar un conjunto de metodologías activas que se aplican de manera recurrente en la formación de ingenieros de software en la educación superior. Estas metodologías se implementan con distintos niveles de profundidad, en diversas modalidades educativas y con el apoyo de herramientas tecnológicas que facilitan la colaboración, la simulación de entornos profesionales y la integración entre teoría y práctica. Los resultados se organizan en torno a cinco enfoques principales: Aprendizaje Basado en Proyectos, Aprendizaje Basado en Problemas, Aprendizaje Colaborativo, Aprendizaje Experiencial y Simulaciones profesionales. A continuación, se presentan los hallazgos más relevantes, acompañados de tablas que sintetizan las características, beneficios y desafíos identificados en la literatura.

Aprendizaje Basado en Proyectos (ABP)

El Aprendizaje Basado en Proyectos se posiciona como la metodología activa más utilizada en la enseñanza de la Ingeniería de Software. La literatura coincide en que esta metodología permite integrar de manera efectiva los contenidos teóricos con la práctica profesional, ya que los estudiantes desarrollan proyectos reales o simulados que abarcan diversas etapas del ciclo de vida del software. Los proyectos suelen involucrar actividades como análisis de requisitos, diseño de arquitecturas, programación, pruebas, documentación y presentación de resultados, lo que favorece el desarrollo de competencias técnicas y transversales.

Los estudios revisados destacan que el ABP incrementa la motivación estudiantil, mejora la retención del conocimiento y fortalece habilidades como la comunicación, el liderazgo y la gestión del tiempo. Sin embargo, también se identifican desafíos relacionados con la carga de trabajo, la coordinación entre equipos y la necesidad de una evaluación equilibrada que considere tanto el desempeño individual como el grupal.

Tabla 1

Características del Aprendizaje Basado en Proyectos en Ingeniería de Software

Aspecto	Descripción
Enfoque principal	Desarrollo de proyectos reales o simulados
Competencias desarrolladas	Programación, diseño, pruebas, gestión de proyectos, comunicación
Beneficios	Mayor motivación, aprendizaje profundo, integración teoría-práctica
Desafíos	Carga de trabajo elevada, coordinación de equipos, evaluación compleja

Aprendizaje Basado en Problemas (PBL)

El Aprendizaje Basado en Problemas se utiliza ampliamente para abordar situaciones complejas que requieren análisis crítico, investigación autónoma y toma de decisiones fundamentadas. En Ingeniería de Software, esta metodología se aplica en cursos relacionados con ingeniería de requisitos, arquitectura de software, pruebas y mantenimiento, donde los estudiantes deben resolver problemas abiertos que no tienen una única solución correcta.

Los estudios muestran que el PBL mejora la capacidad de los estudiantes para identificar problemas, formular hipótesis, justificar decisiones técnicas y evaluar alternativas. Asimismo, fomenta la autonomía y la capacidad de aprender de manera independiente, habilidades esenciales en un campo donde las tecnologías evolucionan rápidamente. No obstante, algunos estudiantes experimentan dificultades iniciales debido a la falta de estructura y a la necesidad de asumir un rol más activo en su aprendizaje.

Tabla 2

Aplicaciones del Aprendizaje Basado en Problemas en Ingeniería de Software

Aspecto	Descripción
Tipos de problemas	Requisitos ambiguos, fallos en sistemas, decisiones de arquitectura
Competencias desarrolladas	Pensamiento crítico, análisis, toma de decisiones
Beneficios	Mayor autonomía, comprensión profunda, habilidades de investigación
Limitaciones	Curva de adaptación, necesidad de guía docente constante

Aprendizaje Colaborativo

El aprendizaje colaborativo se presenta como un componente esencial en la formación de ingenieros de software, dado que la industria exige la capacidad de trabajar en equipos multidisciplinarios. La literatura destaca que esta metodología fomenta la comunicación efectiva, la negociación, la responsabilidad compartida y la construcción colectiva del conocimiento. Las actividades colaborativas suelen incluir proyectos grupales, discusiones guiadas, revisiones por pares y resolución conjunta de problemas.

Los estudios analizados evidencian que el aprendizaje colaborativo mejora la cohesión del grupo, incrementa la motivación y favorece el desarrollo de habilidades socioemocionales. Sin embargo, su implementación requiere estrategias claras para gestionar conflictos, distribuir responsabilidades y evaluar el desempeño individual dentro del grupo.

Tabla 3

Elementos del Aprendizaje Colaborativo en Ingeniería de Software

Aspecto	Descripción
Actividades comunes	Proyectos grupales, revisiones por pares, debates técnicos
Competencias desarrolladas	Comunicación, negociación, trabajo en equipo
Beneficios	Mayor motivación, cohesión grupal, aprendizaje social
Desafíos	Gestión de conflictos, desigualdad en la participación

Aprendizaje Experiencial

El aprendizaje experiencial, basado en el ciclo de Kolb, se implementa mediante actividades que permiten a los estudiantes aprender a partir de la acción, la reflexión y la aplicación. En Ingeniería de Software, esta metodología se traduce en prácticas profesionales, laboratorios de programación, simulaciones de proyectos ágiles y actividades que replican procesos reales del desarrollo de software.

Los estudios revisados indican que el aprendizaje experiencial mejora la comprensión de conceptos complejos, fortalece la capacidad de aplicar conocimientos en situaciones reales y aumenta la confianza del estudiante en sus habilidades. Sin embargo, su implementación requiere recursos tecnológicos, infraestructura adecuada y docentes capacitados para guiar el proceso reflexivo.

Tabla 4

Ciclo del Aprendizaje Experiencial aplicado a Ingeniería de Software

Etapa	Aplicación
Experiencia concreta	Desarrollo de prototipos, prácticas en empresas
Observación reflexiva	Análisis de errores, revisión de código
Conceptualización abstracta	Relación con teorías de diseño, patrones, metodologías
Experimentación activa	Implementación de mejoras, pruebas iterativas

Simulaciones y Entornos Profesionales

Las simulaciones permiten recrear escenarios reales de la industria del software, como la gestión de proyectos ágiles, la integración continua, el control de versiones y la resolución de conflictos en equipos de desarrollo. Estas experiencias ofrecen un entorno seguro donde los estudiantes pueden experimentar, cometer errores y aprender sin las consecuencias que tendría en un contexto profesional real.

La literatura destaca que las simulaciones mejoran la capacidad de los estudiantes para tomar decisiones bajo presión, gestionar recursos, adaptarse a cambios y trabajar en entornos dinámicos. No obstante, su implementación requiere herramientas tecnológicas especializadas y una planificación cuidadosa para garantizar que las actividades sean auténticas y significativas.

Tabla 5

Simulaciones utilizadas en la formación de ingenieros de software

Tipo de simulación	Descripción
Proyectos ágiles	Simulación de sprints, retrospectivas y planificación
Integración continua	Uso de pipelines automatizados y repositorios colaborativos
Control de versiones	Resolución de conflictos, fusiones y manejo de ramas
Gestión de equipos	Roles, liderazgo, negociación y resolución de conflictos

DISCUSIÓN

Los resultados obtenidos a partir del análisis de la literatura permiten comprender con mayor claridad el papel que desempeñan las metodologías activas en la formación de ingenieros de software en la educación superior. En conjunto, las evidencias muestran que estas metodologías

no solo favorecen el aprendizaje significativo, sino que también contribuyen al desarrollo de competencias profesionales esenciales para el ejercicio de la ingeniería en contextos reales. La discusión que sigue integra estos hallazgos con los fundamentos teóricos previamente expuestos, destacando las implicaciones pedagógicas, los desafíos de implementación y las oportunidades de mejora para las instituciones educativas.

En primer lugar, se observa que el Aprendizaje Basado en Proyectos (ABP) se consolida como la metodología más alineada con la naturaleza de la Ingeniería de Software. La disciplina exige la capacidad de integrar conocimientos teóricos con habilidades prácticas, trabajar en equipos multidisciplinarios y gestionar procesos complejos a lo largo del ciclo de vida del software. El ABP permite recrear estas condiciones de manera auténtica, lo que explica su impacto positivo en la motivación, la autonomía y la adquisición de competencias técnicas. Sin embargo, la literatura también advierte que su implementación requiere una planificación rigurosa, una adecuada selección de proyectos y mecanismos de evaluación que reconozcan tanto el desempeño individual como el colectivo. Esto implica que las instituciones deben invertir en capacitación docente, diseño curricular y recursos tecnológicos que permitan sostener proyectos de alta complejidad.

Por otro lado, el Aprendizaje Basado en Problemas (PBL) emerge como una metodología especialmente útil para desarrollar el pensamiento crítico y la capacidad de resolver problemas abiertos, características fundamentales en la ingeniería. La resolución de problemas ambiguos, la toma de decisiones fundamentadas y la capacidad de justificar soluciones técnicas son habilidades que se fortalecen mediante el PBL. No obstante, su efectividad depende en gran medida del acompañamiento docente, ya que los estudiantes pueden experimentar dificultades iniciales al enfrentarse a escenarios poco estructurados. Esto sugiere la necesidad de estrategias de andamiaje que permitan una transición gradual desde metodologías tradicionales hacia enfoques más autónomos.

El aprendizaje colaborativo, por su parte, se presenta como un componente transversal a todas las metodologías activas analizadas. La industria del software se caracteriza por el trabajo en equipos multidisciplinarios, donde la comunicación, la negociación y la responsabilidad compartida son esenciales. La literatura confirma que las actividades colaborativas fortalecen estas competencias y mejoran la cohesión grupal, pero también señala desafíos relacionados con la gestión de conflictos, la distribución equitativa de tareas y la evaluación del aporte individual. Esto implica que los docentes deben diseñar estrategias claras de organización del trabajo en equipo, así como mecanismos de evaluación que reconozcan la diversidad de roles y responsabilidades dentro del grupo.

El aprendizaje experiencial, fundamentado en el ciclo de Kolb, se revela como una metodología particularmente efectiva para integrar teoría y práctica. Las actividades que permiten a los estudiantes experimentar, reflexionar, conceptualizar y aplicar nuevamente lo aprendido

generan un aprendizaje profundo y duradero. En Ingeniería de Software, esto se traduce en prácticas profesionales, laboratorios de programación, simulaciones de proyectos ágiles y actividades que replican procesos reales del desarrollo de software. Sin embargo, su implementación requiere infraestructura adecuada, acceso a herramientas tecnológicas y docentes capacitados para guiar el proceso reflexivo. Esto plantea un desafío para instituciones con recursos limitados, pero también abre oportunidades para el uso de entornos virtuales y simuladores que reduzcan la dependencia de laboratorios físicos.

Las simulaciones profesionales constituyen otro aporte relevante identificado en la literatura. Estas permiten recrear escenarios reales de la industria, como la gestión de proyectos ágiles, la integración continua, el control de versiones y la resolución de conflictos en equipos de desarrollo. Las simulaciones ofrecen un entorno seguro donde los estudiantes pueden experimentar, cometer errores y aprender sin las consecuencias que tendría en un contexto profesional real. No obstante, su implementación requiere herramientas tecnológicas especializadas y una planificación cuidadosa para garantizar que las actividades sean auténticas y significativas. Esto sugiere que las instituciones deben establecer alianzas con empresas tecnológicas, invertir en plataformas de simulación y promover la actualización docente en metodologías ágiles y herramientas de desarrollo colaborativo.

Un aspecto transversal que emerge de la discusión es la importancia de la tecnología como mediadora del aprendizaje activo. Las plataformas colaborativas, los repositorios de código, los entornos de desarrollo en la nube y las herramientas de comunicación sincrónica y asincrónica potencian la implementación de metodologías activas, especialmente en entornos virtuales e híbridos. Sin embargo, la literatura advierte que la tecnología por sí sola no garantiza el aprendizaje; su efectividad depende del diseño pedagógico, la capacitación docente y la disposición de los estudiantes para participar activamente en su proceso formativo.

Finalmente, la discusión revela que la implementación de metodologías activas en la formación de ingenieros de software no está exenta de desafíos. Entre los más relevantes se encuentran la resistencia al cambio por parte de docentes y estudiantes, la necesidad de formación pedagógica especializada, la carga de trabajo asociada a la planificación y evaluación de actividades complejas, y las limitaciones de infraestructura tecnológica. No obstante, estos desafíos también representan oportunidades para transformar la educación superior hacia modelos más flexibles, centrados en el estudiante y orientados a la práctica profesional.

En síntesis, la discusión evidencia que las metodologías activas constituyen un enfoque pedagógico altamente pertinente para la formación de ingenieros de software, ya que permiten desarrollar competencias técnicas, cognitivas y socioemocionales esenciales para el ejercicio profesional. Su implementación requiere un compromiso institucional, una visión pedagógica clara y una integración estratégica de tecnologías educativas, pero sus beneficios justifican ampliamente el esfuerzo.

CONCLUSIONES

El análisis realizado permite concluir que las metodologías activas constituyen un enfoque pedagógico altamente pertinente y necesario para la formación de ingenieros de software en la educación superior. La naturaleza dinámica, colaborativa y orientada a la resolución de problemas de la Ingeniería de Software exige estrategias formativas que trasciendan la transmisión tradicional de contenidos y que sitúen al estudiante como protagonista de su propio aprendizaje. En este sentido, las metodologías activas ofrecen un marco idóneo para desarrollar competencias técnicas, cognitivas y socioemocionales que responden a las demandas actuales de la industria tecnológica.

Los resultados evidencian que el Aprendizaje Basado en Proyectos se posiciona como la metodología más alineada con los procesos reales del desarrollo de software, ya que permite integrar teoría y práctica mediante la ejecución de proyectos auténticos. Esta metodología favorece la motivación, la autonomía y la adquisición de habilidades profesionales, aunque su implementación requiere una planificación rigurosa, una adecuada selección de proyectos y mecanismos de evaluación que reconozcan tanto el desempeño individual como el colectivo.

El Aprendizaje Basado en Problemas, por su parte, se revela como una estrategia eficaz para fortalecer el pensamiento crítico, la capacidad de análisis y la toma de decisiones fundamentadas. Estas competencias son esenciales en un campo donde los problemas suelen ser ambiguos, complejos y abiertos. Sin embargo, su efectividad depende del acompañamiento docente y de la disposición de los estudiantes para asumir un rol activo en su proceso formativo.

El aprendizaje colaborativo emerge como un componente transversal que potencia la interacción social, la comunicación y la responsabilidad compartida. La industria del software se caracteriza por el trabajo en equipos multidisciplinarios, por lo que esta metodología contribuye significativamente a preparar a los estudiantes para entornos laborales reales. No obstante, su implementación requiere estrategias claras para gestionar conflictos, distribuir responsabilidades y evaluar el aporte individual dentro del grupo.

El aprendizaje experiencial y las simulaciones profesionales complementan este panorama al ofrecer oportunidades para que los estudiantes vivan experiencias cercanas a la práctica profesional. Estas metodologías permiten recorrer el ciclo completo del aprendizaje, desde la experiencia concreta hasta la experimentación activa, fortaleciendo la comprensión profunda y la transferencia del conocimiento. Su implementación, sin embargo, demanda infraestructura tecnológica, acceso a herramientas especializadas y docentes capacitados para guiar procesos reflexivos.

De manera transversal, la discusión evidencia que la tecnología desempeña un papel fundamental en la implementación de metodologías activas, especialmente en entornos virtuales e híbridos. Plataformas colaborativas, repositorios de código, entornos de desarrollo en la nube y

herramientas de comunicación facilitan la interacción, la colaboración y la simulación de escenarios profesionales. No obstante, la tecnología debe entenderse como un medio y no como un fin; su efectividad depende del diseño pedagógico, la capacitación docente y la participación activa de los estudiantes.

Finalmente, se reconoce que la adopción de metodologías activas implica desafíos significativos, entre ellos la resistencia al cambio, la necesidad de formación pedagógica especializada, la carga de trabajo asociada a la planificación y evaluación de actividades complejas, y las limitaciones de infraestructura. Sin embargo, estos desafíos representan también oportunidades para transformar la educación superior hacia modelos más flexibles, inclusivos y centrados en el estudiante. La evidencia sugiere que los beneficios de las metodologías activas superan ampliamente las dificultades, y que su implementación contribuye a formar ingenieros de software más competentes, autónomos, críticos y preparados para enfrentar los retos de una industria en constante evolución.

En síntesis, las metodologías activas no solo enriquecen el proceso de enseñanza-aprendizaje, sino que también fortalecen la pertinencia y la calidad de la formación profesional en Ingeniería de Software. Su integración estratégica en los planes de estudio constituye un paso fundamental para garantizar que los futuros ingenieros cuenten con las competencias necesarias para desenvolverse con éxito en un entorno tecnológico complejo, globalizado y altamente competitivo.

REFERENCIAS

Berglund, A., Eckerdal, A., & Pears, A. (2019). *Problem solving and learning in computing education*. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), Computer science education: Perspectives on teaching and learning in school (pp. 81–102). Bloomsbury Academic.

Borrego, M., Cutler, S., Prince, M., Henderson, C., & Froyd, J. (2020). Fidelity of implementation of research-based instructional strategies in engineering science courses. *Journal of Engineering Education, 109*(3), 445–469. https://doi.org/10.1002/jee.20020

Davidovitch, L., Parush, A., & Shtub, A. (2006). Simulation-based learning in engineering education: Performance and transfer in project management learning. *Journal of Engineering Education, 95*(4), 289–299. https://doi.org/10.1002/j.2168-9830.2006.tb00904.x

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., & Abrahamsson, P. (2017). Performance alignment work: How software developers experience the continuous adaptation of team performance in agile software development. *Information and Software Technology, 88*, 72–88. https://doi.org/10.1016/j.infsof.2015.01.010 (doi.org in Bing)

Fioravanti, M. L., Romeiro, B. O., Moreno, A. M., & otros autores. (2023). Software engineering education through experiential learning to foster soft skills. *Frontiers in Education, 8*, artículo 10343452. https://doi.org/10.1109/FIE58773.2023.10343452

Kolb, D. A. (2015). *Experiential learning: Experience as the source of learning and development* (2nd ed.). Pearson Education.

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge University Press.

Prince, M. (2004). Does active learning work? A review of the research. *Journal of Engineering Education, 93*(3), 223–231. https://doi.org/10.1002/j.2168-9830.2004.tb00809.x

Richardson, I., & Delaney, Y. M. (2009). Problem-based learning in the software engineering classroom. In 2009 22nd Conference on Software Engineering Education and Training (CSEET) (pp. 197–204). IEEE. https://doi.org/10.1109/CSEET.2009.34

Tian, K., Cooper, K., & Zhang, K. (2011). Enhancing software engineering education through extended practical experiences. In 2011 10th International Conference on Computer and Information Science (ICIS) (pp. 533–538). IEEE. https://doi.org/10.1109/ICIS.2011.53

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.

Zakaria, M. I., & Maat, S. M. (2019). A systematic review of problem-based learning in education. *Creative Education, 10*(12), 2671–2688. https://doi.org/10.4236/ce.2019.1012194