RECIBIDO EL 29 DE JULIO DE 2021 - ACEPTADO EL 30 DE OCTUBRE DE 2021

DESARROLLO DE HABILIDADES TÉCNICAS EN INGENIERÍA DE SOFTWARE APLICANDO INGENIERÍA INVERSA

DEVELOPMENT OF TECHNICAL SKILLS IN SOFTWARE ENGINEERING APPLYING REVERSE ENGINEERING

Martín Emilio Monroy Ríos[1],

Gabriel Elías Chanchí Golondrino[2],

Manuel Alejandro Ospina Alarcón[3]

Universidad de Cartagena

RESUMEN

El mundo actual exige nuevas estrategias de formación profesional, que mejoren las prácticas de enseñanza-aprendizaje, teniendo en cuenta las características de las nuevas generaciones. El objetivo de este trabajo es proponer una estrategia didáctica para cursos de ingeniería de software, fundamentada en la ingeniería inversa y el aprendizaje basado en problemas (ABP). Se aplicó la metodología de investigación-acción, con la participación de 86 estudiantes y 3 docentes, usando observación participante, entrevistas y repositorios de trabajos de los estudiantes. La estrategia didáctica propuesta se estructura en un conjunto de principios, lineamientos, actividades de clase y recomendaciones. Los resultados del estudio permitieron concluir que la estrategia didáctica propuesta ayudó a estimular la motivación de los estudiantes y contribuyó al logro de los

1 Martín Emilio Monroy Ríos. Doctor en Ingeniería Telemática - Universidad del Cauca. Profesor - Facultad de Ingeniería - Universidad de Cartagena. Colombiano, C.C. 80411302. mmonroyr@unicartagena.edu.co. ORCID: https://orcid.org/0000-0003-4135-3251.

2 Gabriel Elías Chanchí Golondrino. Doctor en Ingeniería Telemática - Universidad del Cauca. Profesor - Facultad de Ingeniería - Universidad de Cartagena. Colombiano, C.C. 10.301.274. gcchanchig@unicartagena.edu.co. ORCID: https://orcid.org/0000-0002-0257-1988.

3 Manuel Alejandro Ospina Alarcón. Doctor en Ingeniería – Ciencia y Tecnología de Materiales – Universidad Nacional de Colombia -sede Medellín. Profesor - Facultad de Ingeniería - Universidad de Cartagena. Colombiano, C.C. 71.265.598. mospinaa@unicartagena.edu.co. ORCID: https://orcid.org/0000-0003-4510-0753.

resultados de aprendizaje, porque desarrollaron habilidades técnicas, cognitivas y conductuales.

PALABRAS CLAVES

actividades de clase; estrategia didáctica; ingeniería inversa; habilidades; ingeniería de software.

ABSTRACT

Today's world circumstances demand new professional training strategies that improve teaching-learning practices, taking into account the characteristics of the new generation. The aim of this work is to propose a didactic strategy for software engineering courses, based on reverse engineering and problem-based learning (PBL). With the participation of 86 students and 3 teachers, we apply the action research methodology using participant observation, interviews, and student work repositories. The didactic strategy is structured in a set of principles, guidelines, class activities, and recommendations. The results of the study allowed us to conclude that this didactic strategy helped to stimulate the student's motivation and contributed to the achievement of the learning outcomes because they developed technical, cognitive, and behavioral skills.

KEYWORDS

class activities; didactic strategy; reverse engineering; skills; software engineering.

1. INTRODUCCIÓN

La formación de los ingenieros del futuro es un desafío que exige cambios en las estrategias de enseñanza y aprendizaje (Motyl et al., 2017). Esto implica que los estudiantes deben ser agentes activos, conscientes y responsables de su aprendizaje, y que el docente debe ser quien guía al estudiante en la búsqueda de información, brindándole oportunidades para que desarrolle habilidades cognitivas, técnicas y de comportamiento (Morales Bueno, 2018), favoreciendo así a la formación de profesionales autónomos, principal fin de la educación superior. Para lograrlo no debemos olvidar que los estudiantes del nuevo milenio se caracterizan por que para ellos (Frand, 2000): 1) El computador no es tecnología, 2) Internet es mejor que la televisión, 3) la realidad ya no es real, 4) hacer es más importante que conocer, 5) el aprendizaje se parece más a Nintendo que a la lógica, 6) la multitarea es una forma de vida, 7) se prefiere mecanografiar a escribir a mano, 8) mantenerse conectado es esencial, 9) hay tolerancia cero para los retrasos y 10) el consumidor y el creador se difuminan.

En este sentido, la Organización para la Cooperación y el Desarrollo Económicos (OCDE) reportó que uno de los principales desafíos de los sistemas de educación es la mejora de las prácticas educativas en todos los niveles de formación (OECD, 2016). Esto conlleva a hacer cambios en los procesos de enseñanza aprendizaje. Para lograr este propósito, enfoques como el constructivismo propone que el docente debe estimular el aprendizaje significativo en los estudiantes, para que pueda conectar sus procesos de construcción de conocimiento con el conocimiento colectivo culturalmente organizado (Barkin, 2003), lo cual se manifiesta con la mejora de las habilidades (Kostiainen et al., 2018). Para el caso concreto de la Ingeniería de Software, el modelo de competencia (IEEE, 2014) incluye: habilidades cognitivas, habilidades y atributos de comportamiento, habilidades técnicas, conocimientos requeridos y conocimientos sobre disciplinas relacionadas.

Las habilidades cognitivas se manifiestan en la capacidad de aplicar el conocimiento y el razonamiento, mientras se realizan actividades propias de la ingeniería de software aplicando habilidades técnicas. Las habilidades cognitivas incluyen el razonamiento, habilidades analíticas, la resolución de problemas y la innovación.

Las habilidades y atributos de comportamiento se evidencian con la capacidad de aplicar de manera eficiente el conocimiento, las habilidades cognitivas y las habilidades técnicas. Estas habilidades incluyen aptitud, iniciativa, entusiasmo, ética de trabajo, confiabilidad, voluntad, habilidades de comunicación, sensibilidad cultural, habilidades de participación en equipo y habilidades de liderazgo técnico. Las habilidades técnicas se refieren al conocimiento especializado y la capacidad necesaria para realizar acciones, tareas y procesos complejos relacionados con la disciplina de la ingeniería de software. Las habilidades técnicas se agrupan en áreas de habilidades de ciclo de vida y áreas de habilidades transversales (IEEE, 2014).

Las habilidades del ciclo de vida del software corresponden a aquellas que son necesarias para cumplir actividades específicas, relacionadas con el desarrollo y mantenimiento de software, tales como: habilidades de ingeniería de requisitos, habilidades de diseño, habilidades de construcción de software, habilidades para realizar pruebas de software y habilidades para hacer mantenimiento a productos software. Las habilidades transversales son aquellas que se aplican a lo largo del ciclo de vida del software, y se clasifican en habilidades transversales al proceso y ciclo de vida de software, habilidades de ingeniería de sistemas, habilidades para garantizar la calidad de software, habilidades para garantizar la seguridad de software, habilidades para la gestión de configuración de software, habilidades para la medición de software y habilidades garantizar la interacción persona-computadora.

Por otra parte, la ingeniería inversa es el proceso de análisis de un sistema para identificar los componentes que lo constituyen y la manera como se relacionan entre sí, con el fin de crear representaciones del sistema en otra forma o en un nivel superior de abstracción (Chikofsky & Cross, 1990), lo que implica la recuperación del conocimiento implícito en el sistema (Tonella et al., 2007). La ingeniería inversa se utiliza en múltiples disciplinas y contextos (Monroy et al., 2017, 2016). En ingeniería mecánica se ha utilizado en el aula para conocer y comprender productos, fomentando el desarrollo de habilidades de innovación de los estudiantes, especialmente a la hora de mejorar y crear nuevos productos (Luna et al., 2010). En ingeniería electrónica se ha usado para desarrollar habilidades cognitivas, como comprender un producto y revisar conceptos físicos, y habilidades técnicas para proponer mejoras en el diseño, construcción y operación del producto (Ramos, 2013).

La ingeniería inversa se utiliza actualmente en el contexto de la ingeniería de software, en actividades como la producción de software, la seguridad informática, la educación y la informática forense (Monroy et al., 2017, 2016). Algunos estudios afirman que las técnicas de ingeniería inversa mejoran las habilidades cognitivas como la comprensión de un sistema, facilitando el desarrollo de habilidades técnicas como el diseño y la programación (Ali, 2005), contribuyendo a generar entusiasmo en los estudiantes y al desarrollo sistemático de la capacidad de análisis y el pensamiento lógico (Klimek et al., 2011). Mientras que otros estudios sostienen que es necesario incluir la ingeniería inversa en la formación de ingenieros de software y profesionales en el campo de la informática (Monroy et al., 2019; Ali, 2005).

Hay propuestas basadas en el enfoque socio-constructivista y otras desde una perspectiva de visión sistémica, donde se definen las competencias de los ingenieros de software que participan en el desarrollo y modificación de sistemas intensivos en software (IEEE, 2016, 2014). Estas propuestas definen habilidades cognitivas, habilidades y atributos de comportamiento y habilidades técnicas. Sin embargo, también se observa la ausencia desde

la perspectiva pedagógica de estrategias que permitan lograr el desarrollo de estas habilidades en el aula (Semerikov et al., 2020; Monroy et al., 2019; Striuk & Semerikov, 2019; Díaz-Barriga, 2014), teniendo en cuenta las características de los estudiantes del nuevo milenio y las exigencias que el medio actualmente requiere de los profesionales en el campo de la ingeniería de software.

En consecuencia, el principal aporte de este trabajo es la propuesta de una estrategia didáctica fundamentada en la ingeniería inversa y el aprendizaje basado en problemas, para el desarrollo de habilidades técnicas en el proceso de enseñanza aprendizaje de la ingeniería de software. La aplicación de la estrategia propuesta hace posible que los estudiantes logren aprendizajes significativos, representados en el desarrollo de habilidades, teniendo en cuenta sus características. Adicionalmente, sugerimos un conjunto de actividades de aprendizaje, que pueden ser utilizadas para desarrollar habilidades conductuales, cognitivas y técnicas aplicando ingeniería inversa en el aula. La estrategia propuesta y las actividades de aprendizaje sugeridas pretenden servir de referencia para profesores del área de la ingeniería de software y de áreas afines, en cuanto al desarrollo de competencias y resultados de aprendizaje en esta área.

A continuación, explicamos la metodología utilizada en el desarrollo de la investigación, posteriormente presentamos los resultados y su discusión. En la parte final exponemos las conclusiones y trabajos futuros derivadas de la presente investigación.

2. METODOLOGÍA

Se realizó una investigación cualitativa bajo el enfoque metodológico de investigación-acción, porque el objetivo principal de este trabajo es mejorar las prácticas docentes, en lugar de generar nuevos conocimientos. Además, aquí se entiende que la investigación no sólo debe usarse para obtener una mejor comprensión de los problemas que surgen en la práctica diaria, sino que en realidad se propone para alterar las cosas, como parte integral del proceso de investigación, en lugar de etiquetarlo como una ocurrencia tardía que sigue a la conclusión de la investigación (Denscombe, 2014). Para la recolección de datos utilizamos la observación participante, entrevistas y repositorios de trabajos de los estudiantes, lo que nos permitió triangular la información. En la investigación-acción el proceso cíclico comprende cinco actividades (Denscombe, 2014): 1) Práctica profesional, 2) Reflexión crítica, 3) Investigación, 4) Planificación estratégica y 5) Acción. Los resultados publicados en este artículo corresponden a dos ciclos. El primer ciclo centró la atención en la generación del diagnóstico de la situación. Se hizo durante un semestre, participaron 19 estudiantes y 3 profesores. El segundo ciclo se focalizó en el análisis de los resultados obtenidos al aplicar la estrategia didáctica propuesta. Se llevó a cabo durante dos semestres, participaron 67 estudiantes con los mismos 3 profesores.

La práctica profesional en este trabajo incluye la práctica docente en el proceso de enseñanza-aprendizaje de la ingeniería de software, en la Facultad de Ingeniería de la Universidad de Cartagena - Colombia. Esta práctica comprende los siguientes cursos del plan de estudios del Programa de Ingeniería de Sistemas: Algoritmos, Programación Básica, Estructuras de Datos, Bases de Datos, Ingeniería de Servicios de Internet, Gestión de la Calidad y Pruebas de Software, Gestión de Proyectos de Ingeniería de Software, Tópicos Avanzados de Ingeniería de Software, Programación Orientada a Objetos, Ingeniería de Software y Arquitectura de Software, de los cuales sólo se incluyeron en la investigación cursos de los tres últimos. La investigación se enfocó en analizar dos categorías de la práctica docente: La

motivación por parte de los estudiantes y los aprendizajes construidos por ellos. La motivación se entiende como un fenómeno multifacético y dinámico que brinda energía, direccionalidad y persistencia a la conducta, dirigiéndola hacia el logro de un propósito (Berridge, 2018). El aprendizaje se concibe como un proceso autorregulado que conduce al desarrollo de estructuras conceptuales a través de la reflexión y la abstracción (Van Eekelen, 2003).

En la fase de reflexión crítica realizamos un diagnóstico de la situación valorando las siguientes categorías: cumplimiento de lineamientos gubernamentales exigidos por el Ministerio de Educación Nacional; cumplimiento de lineamientos internacionales sobre el currículo, definidos en el documento que establece la base del conocimiento en el campo de la ingeniería de software (SWEBOOK Software Engineering Body of Knowledge) (IEEE, 2016); cumplimiento de los objetivos de aprendizaje y percepción de los estudiantes sobre su motivación en la realización de las actividades académicas. Como resultado del diagnóstico planteamos las siguientes preguntas: 1) ¿Cómo se está aplicando la práctica docente de la ingeniería de software en nuestra institución?, 2) ¿Qué nos gustaría cambiar? y 3) ¿Cómo podemos mejorar esta situación?

En la fase de investigación realizamos una indagación sistemática y rigurosa, aplicando las técnicas de recolección de datos mencionadas previamente, y triangulación de resultados para descartar información incorrecta. La realización de esta fase en el primer ciclo nos permitió establecer en forma más precisa el diagnóstico de la situación actual y responder a la primera pregunta planteada. Para el segundo ciclo, en esta fase se realizó el estudio del impacto que tuvo aplicar la estrategia didáctica en los procesos académicos de cada uno de los cursos que hicieron parte de la investigación. La fase de planeación se usó para diseñar la estrategia didáctica en el primer ciclo, y en el segundo para perfeccionarla. Finalmente, en la fase de acción conducimos al cambio poniendo en práctica un conjunto de mejoras obtenidas como resultado de la investigación, y que se proponen como principal aporte en este trabajo.

3. RESULTADOS Y DISCUSIÓN

En esta sección explicamos en primer lugar la estrategia didáctica que proponemos para mejorar la práctica docente en la enseñanza de la ingeniería de software, estructurada en un conjunto de principios, lineamientos, actividades de clase y recomendaciones. Posteriormente analizamos el impacto que tuvo en nuestro programa académico la aplicación de la estrategia didáctica.

Respondiendo a la primera pregunta (¿Cómo se está aplicando la práctica docente de la ingeniería de software?), evidenciamos que el programa cumple con los lineamientos gubernamentales exigidos por el Ministerio de Educación Nacional, porque cuenta con acreditación de alta calidad. De igual forma, la revisión de la propuesta curricular de las asignaturas objeto de estudio, comprobó el cumplimiento de los lineamientos internacionales sobre el currículo, definidos en el documento SWEBOOK. El cumplimiento de los objetivos de aprendizaje se evidenció con dos hechos, la mayoría de los egresados del programa se desempeñan en el campo del desarrollo de software, y los resultados de las pruebas estatales Saber Pro, para el componente de diseño de software, siempre han sido superiores al grupo de referencia, como se observa en la Figura 1.

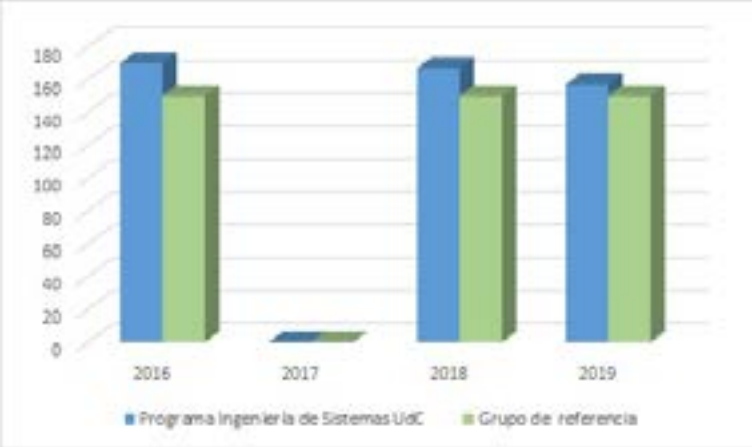

Figura 1- Resultados pruebas Saber Pro. (Datos tomados de ICFES, 2019)

Así mismo se estableció que el programa aplica el modelo pedagógico ecléctico establecido en los lineamientos institucionales, el cual está enfocado en el aprendizaje de los estudiantes, posibilitando el logro de los objetivos de aprendizaje. Además, se evidenció que el cuerpo docente utiliza estrategias pedagógicas como el aprendizaje basado en proyectos, que de igual forma, contribuyen al logro de los objetivos de aprendizaje establecidos en el plan de estudios. Tanto los profesores como los estudiantes reconocen los beneficios de las prácticas docentes en el proceso de enseñanza-aprendizaje.

Sin embargo, la percepción estudiantil también reveló que no se tienen en cuenta las formas de aprendizaje de los estudiantes, ni sus características propias de las nuevas generaciones. Por parte de los profesores identificamos que perciben dificultad al momento de despertar y mantener la motivación de los estudiantes. Además, consideran que pueden mejorar su práctica docente, para lograr mejores resultados con respecto al aprendizaje significativo mantenido a lo largo del tiempo y al desarrollo de habilidades conductuales, cognitivas y técnicas. Por último, con respecto a la primera pregunta planteada, en el diagnóstico de la situación, se identificó que el nuevo modelo de evaluación de los programas de pregrado y posgrado en Colombia, se enfoca más en los resultados de aprendizaje que en los objetivos de aprendizaje; y que la Facultad de Ingeniería de la Universidad de Cartagena, en el momento de la investigación, estaba en un proceso de acreditación intencional con ABET (Accreditation Board of Engineering and Technology).

En consecuencia, con base en el diagnóstico resultado de la reflexión crítica, y como respuesta a la segunda pregunta planteada (¿Qué nos gustaría cambiar?), determinamos que era necesario complementar dos aspectos de las prácticas docentes: 1) Incrementar la motivación de los estudiantes en función de sus características, en calidad de integrantes de una nueva generación y 2) Mejorar el desarrollo de habilidades en los estudiantes asegurando el logro de los resultados de aprendizaje y teniendo en cuenta sus formas de aprendizaje. Para lograr este propósito, dando respuesta a la tercera pregunta planteada (¿Cómo podemos mejorar esta situación?), definimos una estrategia didáctica fundamentada en la ingeniería inversa y el aprendizaje basado en problemas. Este último entendido como una técnica pedagógica que promueve el aprendizaje auto-dirigido y el desarrollo del pensamiento crítico, brindando escenarios en los que los estudiantes deben resolver problemas (Wood, 2003).

La estrategia didáctica que proponemos está estructurada en un conjunto de principios, lineamientos, actividades de clase y recomendaciones que se explican en detalle más adelante. Como se observa en la Figura 2, esta estrategia didáctica asume que los estudiantes, además de contar con conocimientos previos, que se manifiestan a través de sus atributos y habilidades de comportamiento, se caracterizan porque para ellos hacer es más importante que conocer, el aprendizaje se parece más a Nintendo que a la lógica, internet es mejor que la televisión, y el consumidor y el creador se difuminan (Frand, 2000). La estrategia didáctica aplica la técnica del aprendizaje basado en problemas en el proceso de enseñanza aprendizaje, donde el docente propone escenarios que conllevan a que los estudiantes utilicen la ingeniería inversa, "desarmando" productos software para que puedan solucionar el problema planteado, construyendo de esta forma nuevo conocimiento que se manifiesta en el desarrollo de habilidades cognitivas, y que en el ejercicio práctico de la solución del problema implica el desarrollo de habilidades técnicas, que garantizan los resultados de aprendizaje propuestos para cada asignatura.

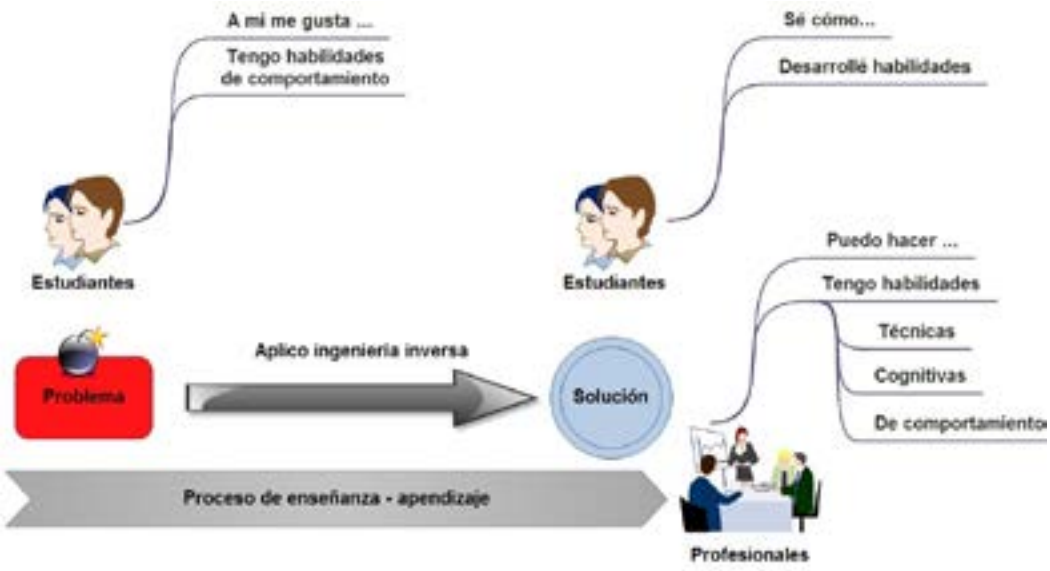

Figura 2 - Esquema de la estrategia didáctica

Esta estrategia se puede aplicar para el desarrollo de un tema específico, o para varios temas integrados. De igual forma, se sugiere que se aplique en todos y cada uno de los cursos de la línea de formación correspondiente al campo de la ingeniería de software, en combinación con otras estrategias pedagógicas. Esto permitirá que al final del proceso de formación el egresado haya desarrollado las habilidades de comportamiento, técnicas, y cognitivas que garanticen su desarrollo profesional con calidad. Para fundamentar la estrategia didáctica que proponemos asumimos los siguientes enunciados como principios: 1) El aprendizaje es una acción voluntaria, por lo tanto, si quiero motivar a los estudiantes debo tener en cuenta sus características y estilos de aprendizaje. 2) Cuando los estudiantes se enfrentan a un problema, tienen una mayor probabilidad de aprender que cuando escuchan una conferencia. 3) Cuando el estudiante desarma un producto, aprende a identificar sus partes y cómo funcionan. La ingeniería inversa se utiliza para analizar y comprender un producto. 4)

La ingeniería inversa brinda la oportunidad de aprender de los éxitos y de los errores, lo que ayuda a los estudiantes a aprender identificando buenas prácticas para repetirlas y malas prácticas para evitarlas.

Para lograr mejores resultados en las actividades de clase definimos los siguientes lineamientos:

1. Para cada actividad se deben definir los resultados de aprendizaje en términos del desarrollo de habilidades conductuales, cognitivas y técnicas. La Tabla 1 muestra algunas habilidades cognitivas, y las Tablas 2 y 3 muestran habilidades técnicas (IEEE, 2014) relacionadas con un conjunto de actividades que se pueden realizar mediante ingeniería inversa.

2. Se debe identificar cuidadosamente cada uno de los problemas abordados, teniendo en cuenta los resultados de aprendizaje establecidos, la disponibilidad del código fuente del sistema con el que trabajarán los estudiantes, y las herramientas de ingeniería inversa necesarias.

3. La formulación del problema a resolver debe obligar a los estudiantes a: identificar, prevenir y/o corregir errores en un código fuente; y/o rediseñar, modificar o mejorar un producto de software. Sugerimos aplicar las recomendaciones de Diana Wood para crear escenarios eficientes de aprendizaje basados en problemas (Wood, 2003).

4. Se debe diseñar una rúbrica de evaluación que establezca los criterios que definen las expectativas de desempeño y describa los niveles de calidad de cada criterio (Andrade, 2000).

5. Es importante tener en cuenta que las actividades de clase deben obligar a los estudiantes a realizar un proceso de ingeniería inversa sobre un código fuente disponible.

6. Aplicar el proceso de aprendizaje basado en problemas, que incluye los siguientes pasos (Wood, 2003): identificar y aclarar términos desconocidos, definir el problema, discutir de problema, generar soluciones alternativas, formular los objetivos de aprendizaje, estudio individual por parte de los estudiantes, y socialización de los resultados obtenidos.

7. Se debe guiar a los estudiantes para que no pierdan el foco en la resolución del problema. Ocasionalmente observamos que los algunos estudiantes pasaban más tiempo haciendo el proceso de ingeniería inversa, porque tenían mucha curiosidad por el producto analizado, y olvidaban el propósito principal de la actividad de clase. Por ello, es necesario que el docente oriente a los estudiantes para que no pierdan la atención de su objetivo principal.

Tabla 1 – Habilidades cognitivas. (Adaptado de IEEE, 2014)

Habilidad	*Habilidad específica*	*Actividad de clase usando ingeniería inversa*
Razonamiento	Razonamiento inductivo	Identifique código fuente duplicado. Identifique buenas prácticas de codificación. Identifique malas prácticas de codificación y corríjalas.
	Razonamiento deductivo	Identifique el uso de patrones de diseño (puede ser uno específico, varios o cualquiera). Identifique el uso de patrones arquitectónicos. Identifique errores en el código fuente.
	Uso de abstracción	Transforme manualmente código fuente en modelos de UML (Lenguaje de Modelado Unificado). Identifique los componentes de un producto software. Remplace el código duplicado.
	Razonamiento asociativo	Identifique código fuente duplicado. Identifique el uso de un patrón de diseño específico. Identifique el uso de un patrón arquitectónico específico. Identifique el uso de estándares en el código fuente.
	Síntesis	Remplace el código fuente duplicado.
Análisis	Análisis de causa	Identifique la causa de un error en el código fuente
	Identificación de riesgos	Identifique posibles riesgos en el código fuente. Identifique el uso de técnicas de codificación para minimizar las amenazas y la propagación de errores.
	Análisis de impacto	Identifique el efecto que genera un cambio en el código fuente.

Habilidad	*Habilidad específica*	*Actividad de clase usando ingeniería inversa*
Resolución de problemas	Dividir y vencer	Identifique los componentes en un sistema software. Identifique el uso de patrones arquitectónicos.
	Analogía y reutilización	Identifique componentes reutilizables. Reutilice los componentes identificados para solucionar un nuevo problema.
	Reconocimiento de patrones	Identifique el uso de un patrón específico de programación. Identifique el uso de un patrón específico de diseño. Identifique el uso de un patrón arquitectónico específico.
	Enfoque incremental	Identifique los puntos de extensión en un sistema software. Agregue una funcionalidad a un sistema software.
Innovación	Modelado	Perfeccione un sistema software.

Tabla 2: Habilidades Técnicas: Área del ciclo de vida. (Adaptado de IEEE, 2014)

Habilidades del área del ciclo de vida	*Habilidad específica*	*Actividad de clase usando ingeniería inversa*
Habilidades de diseño de software	Diseño básico de software	Identifique el uso de alguna técnica disponible en el código fuente (por ejemplo, acoplamiento, cohesión, abstracción, ocultación de información, etc.). Identifique técnicas de tolerancia al fallo y manejo de excepciones en el código fuente. Reestructure un sistema software. Identifique el uso de técnicas de diseño para el manejo de la concurrencia, de eventos, y/o de la persistencia.
	Estrategias de diseño de software	Identifique el uso de estrategias de diseño en un producto software (tales como: arriba-abajo, abajo-arriba, refinamiento paso a paso, uso de patrones y modismos, procesos iterativos e incrementales, etc.).
	Diseño arquitectónico de software	Identifique el uso de estilos arquitectónicos. Reconstruya las vistas arquitectónicas de un sistema software. Identifique las interfaces de los componentes que constituyen un sistema software. Rediseñe un sistema software.
	Análisis de la calidad del diseño de software	Identifique el uso de patrones de diseño, arquitectónicos y modismos. Identifique la aplicación de estrategias de diseño. Identifique posibles errores de diseño.
	Evaluación de la calidad del diseño de software	Identifique decisiones acertadas de diseño en un sistema software. Identifique problemas de diseño en un sistema software.
Habilidades para la construcción de software	Diseño detallado	Rediseñe un sistema software minimizando la complejidad y mejorando la calidad. Identifique el uso apropiado de patrones de diseño.
	Codificación	Identifique código duplicado y sustitúyalo. Identifique si es necesario reorganizar el código fuente y hágalo en caso afirmativo. Identifique el uso de estándares en el código fuente. Identifique el uso de técnicas de defensa para minimizar errores y la propagación de amenazas.

Habilidades del área del ciclo de vida	*Habilidad específica*	*Actividad de clase usando ingeniería inversa*
Habilidades para el mantenimiento de software	Mantenimiento de software	Identifique errores de diseño en un sistema software y corríjalos. Identifique errores de programación en un sistema software y corríjalos. Adapte un sistema software a una nueva situación. Agregue funcionalidades a un sistema software.

Tabla 3: Habilidades Técnicas: Habilidades transversales. (Adaptado de IEEE, 2014)

Habilidades transversales	*Habilidad específica*	*Actividad de clase usando ingeniería inversa*
Habilidades de ingeniería de sistemas de software	Diseño de sistemas	Identifique los componentes hardware y software que conforman el sistema. Reconstruya la vista de despliegue del sistema.
Habilidades de calidad de software	Revisión	Recopile y analice los datos apropiados resultantes de un proceso de ingeniería inversa sobre un sistema software. Identifique y realice las acciones correctivas necesarias.
	Auditoría	Recopile y analice los datos apropiados resultantes de un proceso de ingeniería inversa sobre un sistema software.

Habilidades transversales	*Habilidad específica*	*Actividad de clase usando ingeniería inversa*
Habilidades de seguridad de software	Diseño	Identifique problemas asociados con amenazas y riesgos, y rediseñe el sistema para solucionarlos. Identifique la aplicación de principios de diseño relacionados con la seguridad del sistema. Identifique el uso de patrones de diseño relacionados con la seguridad del sistema.
	Construcción	Identifique el uso de principios de codificación que garanticen la seguridad del sistema. Identifique el uso de estándares de codificación relacionados con la seguridad del sistema. Verifique que las decisiones de diseño relacionadas con la seguridad, se implementaron bien en el código fuente.
	Calidad	Identifique problemas de calidad relacionados con la seguridad de un sistema software.

La formulación del problema a resolver en cada actividad de clase fue uno de los mayores desafíos que enfrentamos. Por esta razón definimos un conjunto de actividades de clase, que se organizan como se muestra en las Tablas 1, 2 y 3. Las habilidades cognitivas de los ingenieros de software incluyen cuatro clasificaciones (IEEE, 2014): habilidades de razonamiento, analíticas, resolución de problemas e innovación. Cada una de estas clasificaciones incluye habilidades específicas. Por ejemplo, las habilidades de razonamiento incluyen: razonamiento inductivo, razonamiento deductivo, razonamiento heurístico, uso de la abstracción, habilidades de razonamiento jerárquico y asociativo. En la Tabla 1 relacionamos, no de manera exhaustiva, un conjunto de actividades de clase que aplican ingeniería inversa y que apoyan el desarrollo de estas habilidades específicas.

De manera similar, las habilidades técnicas de los ingenieros de software se agrupan como área de habilidades de ciclo de vida y área de habilidades transversales (IEEE, 2014). El área de habilidades del ciclo de vida incluye las habilidades necesarias para realizar diversas actividades laborales dentro de una fase de desarrollo o mantenimiento del software. El área de habilidades transversales se aplica a todas las áreas de habilidades del ciclo de vida e incluso en algunos casos, puede aplicarse a otras áreas de habilidades transversales. En las Tablas 2 y 3, presentamos, de nuevo no exhaustivamente, un conjunto de actividades de clase que aplican ingeniería inversa y que contribuyen al desarrollo de estas habilidades, respectivamente.

Adicionalmente, cuando comenzamos a aplicar esta estrategia didáctica, también enfrentamos desafíos como: la dificultad para encontrar software que pueda servir como objeto de estudio para las actividades de clase; el desconocimiento de una metodología para realizar el proceso de ingeniería inversa en el contexto educativo; la selección incorrecta del software como producto en estudio en las actividades de clase; la dificultad para formular

de manera adecuada los problemas abordados en las actividades de clase; entre otros. Para abordar estas preocupaciones, sugerimos las siguientes recomendaciones:

1. Construir un repositorio de productos de software que sirva como producto en estudio para las actividades de clase. Este repositorio debe ser clasificado considerando criterios como: El uso de patrones de diseño y/o patrones arquitectónicos, la aplicación de buenas prácticas, la presencia de bugs, entre otros.

2. Utilizar un enfoque metodológico al aplicar el proceso de ingeniería inversa, que sea adecuado para el contexto de la educación, como el propuesto por (Monroy et al., 2018).

3. El producto software usado en la actividad de clase debe ser seleccionado cuidadosamente, para que coincida con los resultados del aprendizaje que se pretenden lograr. La selección incorrecta del producto software utilizado impide la consecución de los resultados de aprendizaje propuestos en la actividad de clase.

4. Para obtener mejores resultados, los estudiantes deben estar familiarizado con el proceso de ingeniería inversa. Sugerimos realizar actividades previas de ingeniería inversa con los estudiantes, con el fin de conocer el proceso y las herramientas que se pueden utilizar.

5. Los problemas abordados deben ser acordes con el nivel de formación de los estudiantes, con respecto al plan de estudios del programa. Es recomendable centrar más la atención en promover el desarrollo de habilidades cognitivas en los grados inferiores y en fomentar las habilidades técnicas en los grados superiores, sin olvidar que ambas tienen una estrecha relación.

6. Tener en cuenta que involucrar la ingeniería inversa como estrategia didáctica, requiere un trabajo previo de planificación y diseño de las actividades de clase para lograr los resultados del aprendizaje.

Esta estrategia didáctica se aplicó durante dos semestres en los cursos de Programación Orientada a Objetos, Ingeniería de Software y Arquitectura de Software de la Universidad de Cartagena. Cuando analizamos los resultados, observamos que: 1) Se estimuló la curiosidad del alumno, generando interés por descubrir un producto ya desarrollado. 2) El sentimiento de satisfacción del alumno estimula el deseo de aprender a la hora de resolver el problema (Van Eekelen, 2003). 3) Cuando el estudiante aplica ingeniería inversa a un producto de software aprende a replicar buenas prácticas y corregir errores. 4) Los resultados del aprendizaje se lograron, porque los estudiantes desarrollaron habilidades conductuales, cognitivas y técnicas. Esto representó un aprendizaje significativo en los estudiantes, confirmando que la principal ventaja de aplicar la ingeniería inversa en la enseñanza es el logro de un mejor y más profundo entendimiento (Ali, 2005). 5) Debido a que la ingeniería inversa requiere un trabajo paciente y difícil, ayuda al desarrollo de habilidades para resolver problemas (Ali, 2005). 6) Los estudiantes prefirieron hacer tutoriales en video explicando lo que se hizo, en lugar de escribir informes técnicos (Frand, 2000).

Este trabajo de investigación no pretende establecer generalizaciones sobre el uso de la ingeniería inversa en un proceso de enseñanza-aprendizaje como herramienta didáctica. Sólo se limita a difundir los resultados obtenidos en esta investigación, y a presentar algunas recomendaciones. Todo esto en un contexto de la formación profesional en el campo de la ingeniería de software. Además, dejamos claro que para la evaluación de la motivación, sólo se tuvo en cuenta el contexto externo (el

aula). La razón principal es que el análisis del contexto interno implica el estudio detallado de los aspectos psicológicos y cognitivos de todos los estudiantes que participan en el proceso, lo cual está fuera del alcance de la investigación.

4. CONCLUSIONES Y TRABAJOS FUTUROS

La estrategia didáctica propuesta, fundamentada en el uso de la ingeniería inversa y el aprendizaje basado en problemas, estimuló la motivación de los estudiantes y contribuyó al logro de los resultados del aprendizaje, porque los estudiantes desarrollaron habilidades conductuales, cognitivas y técnicas, como se documentó en los resultados. Confirmamos que el desarrollo de habilidades en el aula sólo es posible cuando se aborda desde la perspectiva pedagógica (Semerikov et al., 2020; Monroy et al., 2019; Striuk & Semerikov, 2019). Asimismo, confirmamos la importancia de involucrar la ingeniería inversa en las propuestas curriculares de los programas de ingeniería (Monroy et al., 2019; Ali, 2005). La estrategia didáctica propuesta no pretende reemplazar estrategias similares; se puede utilizar junto con otras para complementar y enriquecer el proceso de enseñanza-aprendizaje dentro del área de la ingeniería de software y áreas afines.

Como trabajo futuro proponemos sistematizar el diseño de las actividades de clase, su ejecución y el seguimiento del trabajo de cada estudiante, para mantener un control más detallado del proceso de aprendizaje.

AGRADECIMIENTOS

Agradecemos a la Universidad de Cartagena por el apoyo brindado en el desarrollo de esta investigación, así como a los estudiantes y profesores que hicieron posible su realización.

REFERENCIAS BIBLIOGRÁFICAS

Ali, M. R. (2005). Why Teach Reverse Engineering? ACM SIGSOFT Software Engineering Notes, 30(4), 1–4.

Andrade, H. G. (2000). Using rubrics to promote thinking and learning. Educational Leadership, 57(5), 13–18.

Barkin, J. S. (2003). Realist constructivism. International Studies Review, 5(3), 325-342.

Berridge, K. C. (2018). Evolving concepts of emotion and motivation. Frontiers in Psychology, 9, 1647.

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. IEEE Software, 7(1), 13–17.

Denscombe, M. (2014). The good research guide for small-sacle socail research projects. In McGraw-Hill Education (Ed.), Biddles Ltd, Guildford and King's Lynn, Great …. McGraw-Hill Education.

Díaz-Barriga, A. (2014). Construcción de programas de estudio en la perspectiva del enfoque de desarrollo de competencias. Perfiles Educativos, XXXVI(143), 142–162.

Frand, J. L. (2000). The Information-Age Mindset: Changes in Students and Implications for Higher Education. EDUCAUSE Review, 35(5), 15–24.

ICFES, Informe nacional Saber Pro 2016-2019 (2019). Instituto Colombiano para el Fomento de la Educación Superior, https://www.icfes.gov.co/resultados-saber-pro

IEEE. (2014). Software Engineering Competency Model. IEEE Computer Society.

IEEE. (2016). Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0). IEEE Computer Society.

Klimek, I., Keltika, M., & Jakab, F. (2011). Reverse engineering as an education tool in computer science. In IEEE (Ed.), Emerging eLearning Technologies and Applications (ICETA), 9th International Conference on (pp. 123–126). IEEE.

Kostiainen, E., Ukskoski, T., Ruohotie-Lyhty, M., Kauppinen, M., Kainulainen, J., & Mäkinen, T. (2018). Meaningful learning in teacher education. Teaching and Teacher Education, 71, 66–77. https://doi.org/10.1016/j.tate.2017.12.009

Luna, G., Jiménez, E., García, L., & Reyes, L. (2010). The importance of the research programs of reverse engineering in engineering. International Conference on Engineering Education ICEE, 1–8.

Monroy, M., Arciniegas, J. L., & Rodríguez, J. C. (2016). Modelo Ontológico para Contextos de uso de Herramientas de Ingeniería Inversa. Información Tecnológica, 27(4), 165–174.

Monroy, M., Arciniegas, J. L., & Rodríguez, J. C. (2017). Characterization of the contexts of use of reverse engineering. Información Tecnológica, 28(4), 75–84.

Monroy, M., Rodríguez, J. C., & Puello, P. (2019). Reverse engineering as a teaching tool : A case study in the learning of object-oriented programming. 17th LACCEI International Multi-Conference for Engineering, Education, and Technology: “Industry, Innovation, And Infrastructure for Sustainable Cities and Communities,” 1–5. https://doi.org/http://dx.doi.org/10.18687/LACCEI2018.1.1.65

Monroy, M., Rodríguez, J. R., & Puello, P. (2018). A Methodological Approach for Software Architecture Recovery. Indian Journal of Science and Technology, 11(21), 1–8. https://doi.org/10.17485/ijst/2018/v11i21/124487

Morales Bueno, P. (2018). Aprendizaje basado en problemas (ABP) y habilidades de pensamiento crítico , ¿ una relación vinculante ? Problem- - based learning (PBL) and critical thinking skills - - a binding relationship ? Revista Electrónica Interuniversitaria de Formación Del Profesorado, 21(2), 91–108.

Motyl, B., Baronio, G., Uberti, S., Speranza, D., & Filippi, S. (2017). How will Change the Future Engineers’ Skills in the Industry 4.0 Framework? A Questionnaire Survey. Procedia Manufacturing, 11(June), 1501–1509. https://doi.org/10.1016/j.promfg.2017.07.282

OECD. (2016). Educación en Colombia. Aspectos destacados.

Ramos, D. A. (2013). Uso de la ingeniería inversa como metodología de enseñanza en la formación para la innovación. In W. Engineering & E. Forum (Eds.), World Engineering Education Forum. World Engineering Education Forum.

Semerikov, S., Striuk, A., Striuk, L., Striuk, M., & Shalatska, H. (2020). Sustainability in Software Engineering Education : a case of. 10036.

Striuk, A. M., & Semerikov, S. O. (2019). The Dawn of Software Engineering Education. CEUR Workshop Proceedings, 35–57.

Tonella, P., Torchiano, M., Du Bois, B., & Systa, A. (2007). Empirical studies in reverse engineering : state of the art and future trends. Empirical Software Engineering, 12(5), 551–571. https://doi.org/10.1007/s10664-007-9037-5

Van Eekelen, I. M., Boshuizen, H. P. A., & Vermunt, J. D. (2005). Self-regulation in higher education teacher learning. Higher education, 50(3), 447-471.

Wood, D. F. (2003). Problem based learning. Bmj 326.7384, 328–330.