COMPUTATION E INFORMATICA

Recibido 9 FEB. 2024 *ReCIBE, Año 12 No.2, NOV. 2023*

Aceptado 12 FEB. 2024

Métricas de productividad y gestión del trabajo basadas en psp para estudiantes de ingeniería de sistemas: caso universidad eafit

Productivity and work management metrics based on psp for systems engineering students: case study at eafit university

Paola Vallejo[1]
pvallej3@eafit.edu.co
Andrés Echeverri Jaramillo[1]
Liliana González Palacio[1]
Rafael David Rincón[1]

Universidad EAFIT, Medellín, Colombia

RESUMEN

El Proceso Personal de Software (PSP) se utiliza en la industria del software, Ingeniería de Sistemas y programas de formación relacionados. Estudios previos muestran que su inclusión en el currículo mejora la capacidad de los estudiantes para estimar tiempo y recursos, así como para gestionar y controlar eficientemente el proceso de desarrollo de software. Este estudio busca evaluar el impacto de la implementación del PSP en estudiantes de primer semestre de Ingeniería de Sistemas de la Universidad EAFIT, a través de métricas de productividad y gestión del trabajo, con el propósito de mejorar su disciplina, desempeño y la calidad del desarrollo de software. Se presenta un caso de implementación de PSP en el curso Seminario de Ingeniería de Sistemas, a partir de un conjunto de preguntas orientadoras. Cada estudiante registra métricas como cantidad de líneas de código por hora, tiempo de desarrollo por programa y número de errores. Los hallazgos indican que la aplicación del PSP y las métricas de productividad pueden mejorar significativamente la disciplina y el desempeño de los estudiantes, permitiendo un seguimiento más preciso del progreso y una mejor gestión del trabajo. Este artículo ofrece una perspectiva valiosa para los educadores y estudiantes de Ingeniería de Sistemas que deseen implementar buenas prácticas de programación y gestión del trabajo, así como mejorar la calidad del producto desarrollado.

PALABRAS CLAVE

Proceso Personal de Software (PSP), PSP en el aula, Medición de desempeño, Desarrollo de software.

ABSTRACT

The Personal Software Process (PSP) is utilized in the software industry, Systems Engineering, and related training programs. Previous studies demonstrate that its inclusion in the curriculum enhances students' ability to estimate time and resources and efficiently manage and control the software development process. This study aims to evaluate the impact of implementing PSP on first-semester Systems Engineering students at EAFIT University, using productivity and work management metrics to improve their discipline, performance, and the quality of software development. A case of PSP implementation in the Systems Engineering Seminar course is presented, guided by a set of questions. Each student records metrics such as lines of code per hour, development time per program, and the number of errors. Findings indicate that applying PSP and productivity metrics can significantly improve students' discipline and performance, enabling more precise progress monitoring and better work management. This article provides valuable insights for educators and Systems Engineering students looking to implement good programming and work management practices and enhance the quality of the developed product.

KEYWORDS

Personal Software Process (PSP), PSP in the classroom, Performance measurement, Software development.

1. INTRODUCCIÓN

El Proceso Personal de Software (PSP) es ampliamente reconocido y adoptado en la industria del software, en la Ingeniería de Sistemas y programas de formación relacionados [Shen, Hsueh, & Lee, 2011; Razzaq et al., 2020]. Investigaciones anteriores han mostrado los beneficios de incorporar el PSP en el plan de estudios, mejorando la capacidad de los estudiantes para estimar el tiempo y los recursos necesarios para la realización de tareas, así como su competencia en la gestión y el control efectivo del proceso de desarrollo de software [Shen et al., 2011]. Este estudio investiga la aplicación práctica de métricas de productividad y técnicas de gestión del trabajo PSP en estudiantes de primer semestre del programa de Ingeniería de Sistemas de la Universidad EAFIT de Colombia. Se presenta un caso de implementación de los principios de PSP en dos grupos (61 estudiantes) del curso Seminario de Ingeniería de Sistemas, utilizando un conjunto de preguntas orientadoras.

Durante el estudio, cada estudiante registra métricas esenciales, como la cantidad de líneas de código por hora, el tiempo de desarrollo por programa y el número de errores. El estudio destaca los beneficios de utilizar PSP y métricas de productividad para mejorar la disciplina y el rendimiento de los estudiantes. Además, muestra cómo el PSP puede facilitar un seguimiento más preciso del progreso y una mejora en las capacidades de gestión del trabajo. Los resultados resaltan las ventajas de incorporar el PSP y las métricas de productividad a la enseñanza de Ingeniería de Sistemas.

En resumen, este artículo analiza la aplicación del PSP y las métricas de productividad en estudiantes de primer semestre de Ingeniería de Sistemas. Estas ideas son valiosas para educadores y estudiantes interesados en adoptar las mejores prácticas de gestión del trabajo, mejorando la calidad general de los productos de software que desarrollan y contribuyendo al objetivo general de promover la excelencia en los procesos de desarrollo de software y fortalecer el crecimiento profesional de los estudiantes.
Este artículo se organiza como se indica a continuación: la siguiente sección presenta un marco de referencia que relaciona conceptos y estudios sobre el PSP. La sección 3 presenta un paso a paso general para aplicar el PSP en el curso de Seminario de Ingeniería de Sistemas. La sección 4 presenta los resultados obtenidos. Finalmente, la sección 5 enuncia conclusiones y ofrece algunas perspectivas.

2. MARCO DE REFERENCIA

2.1 Conceptos

PSP es una metodología que brinda técnicas y herramientas para la gestión individual del trabajo en el desarrollo de software, permitiendo a los profesionales medir y mejorar su productividad y calidad en el proceso [Razzaq et al., 2020]. En la educación universitaria, el PSP ha sido objeto de estudios y experiencias que han mostrado diversos beneficios y aplicaciones. Estos incluyen mejorar la calidad del trabajo, establecer prácticas disciplinadas, desarrollar habilidades de gestión del tiempo, aprender a estimar tiempos y esfuerzos, fomentar buenas prácticas de programación, practicar la autoevaluación y el autoaprendizaje, y lograr una mejora continua en su proceso de desarrollo de software [Soto & Rafael, 2020; Ramingwong & Ramingwong, 2012].

2.2 Trabajo relacionado

En esta sección se presentan ejemplos de uso del Proceso Personal de Software (PSP) en cursos de Ingeniería de Sistemas a nivel universitario.

Kusakabe y otros analizaron cómo el uso de técnicas de estimación basadas en el PSP en un curso de ingeniería de software resultó en estimaciones más precisas y realistas, lo que mejoró la planificación y gestión de los proyectos [Kusakabe et al., 2020]. Denwattana encontró que los estudiantes, al aplicar PSP, mejoran significativamente la calidad de su trabajo, su habilidad para estimar tareas y administrar el tiempo de manera efectiva. Además, los estudiantes adquirieron una mentalidad más disciplinada y sistemática en su enfoque hacia el desarrollo de software [Ramingwong & Ramingwong, 2012].
Contreras y Soto, por su parte, realizaron dos estudios sobre el impacto de usar PSP para el rendimiento de los estudiantes. Los resultados mostraron mejoras significativas en la calidad del código, la estimación precisa de tareas y la reducción de defectos en el trabajo de los estudiantes. Además, se observó una mayor conciencia sobre el manejo de métricas [Soto & Rafael, 2020; Contreras-Vas et al., 2021]. Otros estudios sobre PSP en la academia están más orientados a conocer la percepción final de los estudiantes al pedirles que usen este modelo para medir su desempeño [Contreras et al., 2020]. El resultado final permitió a los autores mejorar las estrategias de enseñanza de los conceptos asociados a PSP y orientar de otras formas el curso.
Por último, se encontró una revisión de literatura sobre diversos experimentos donde se aplica PSP en entornos universitarios [Arduino & Bollati, 2020]. Se evidencia que, en general, los autores reconocen mejoras en el proceso de codificación de los estudiantes, mediante métricas como cantidad de líneas de código, disminución de errores y mayor productividad. Al analizar las referencias consultadas, es posible identificar un patrón de mejora en el desempeño de los estudiantes, igual se ratifica en nuestro trabajo, y se proponen opciones para darle continuidad e integrar con otros cursos del programa Ingeniería de Sistemas de la Universidad EAFIT.

3. MÉTODO

El siguiente es un paso a paso general para aplicar el Proceso Personal de Software (PSP) en el curso de Seminario de Ingeniería de Sistemas (ver figura 1).

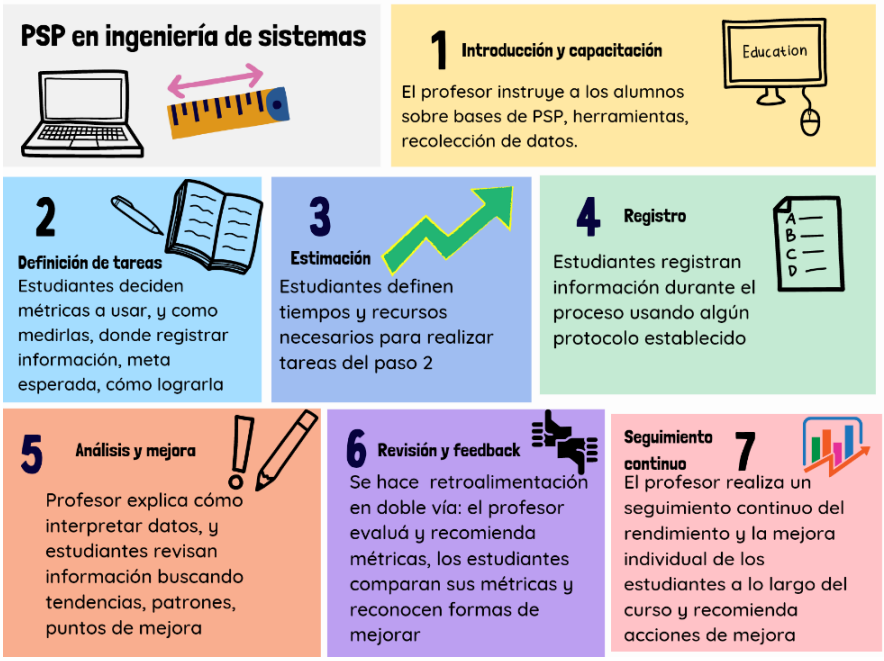

Figura 1. Aplicación del PSP en Seminario de Ingeniería de Sistemas [Borstler et al., 2002; Venkatasubramanian et al., 2001; Maletic et al., 2001]

Las preguntas orientadoras del ejercicio indicado en la figura 1 fueron: ¿Cuáles son las métricas que se utilizarán en el desarrollo de software? ¿Cuál es mi desempeño actual? ¿Cuál es la meta esperada o deseada? ¿Cómo espera alcanzar la meta? ¿Qué se concluye del ejercicio? [Borstler et al., 2002; Venkatasubramanian et al., 2001; Maletic et al., 2001].

Los cursos involucrados en el ejercicio fueron: a) Fundamentos de Programación, donde los estudiantes aplican la teoría de PSP para evaluar su desempeño en las prácticas continuas de programación; b) Seminario de Ingeniería de Sistemas, planteando un reto donde los estudiantes usaban los resultados obtenidos en sus prácticas de programación para hacer seguimiento y recopilar información asociada a las preguntas orientadoras especificadas anteriormente (la cantidad de líneas de código por hora, errores, etc). La población estuvo compuesta por estudiantes de primer semestre de Ingeniería de Sistemas de la universidad EAFIT, cursando las dos asignaturas ya enunciadas.

Si otros docentes desean aplicar un ejercicio como el planteado en este artículo, es importante adaptar y personalizar los pasos mostrados en la figura 1 según las necesidades y características del curso a intervenir.

4. RESULTADOS

Mediante este ejercicio, los estudiantes identificaron formas para medir su desempeño. La mayoría de ellos utilizaron métricas como: número de líneas de código por hora, cantidad de errores y tiempo de desarrollo.

La figura 2 ilustra el promedio de número de líneas de código por hora. Se observa que la mayoría de los estudiantes dedicaron entre 25 y 29 horas al desarrollo de sus actividades, lo que sugiere un grupo significativo de individuos comprometidos y dispuestos a invertir un tiempo considerable en ellas. Además, se observan grupos más pequeños en otros rangos de tiempo, como 20 a 24 horas y 30 a 34 horas, que también demuestran un nivel de compromiso en la actividad. En contraste, algunos rangos de tiempo, como 45 a 49 horas, tienen una representación mínima, lo que indica que pocos participantes han invertido un tiempo tan extenso. La figura 3 presenta la cantidad de horas invertidas por cada programa, es notable que la mayoría de los estudiantes dedicaron entre 0 y 1 hora al desarrollo de sus programas, lo que indica una preferencia por programas que requieren una inversión de tiempo mínima o la simplicidad de los mismos. Por otro lado, un grupo significativo de estudiantes invirtió entre 2 y 3 horas, lo que sugiere un compromiso moderado con programas más extensos. Además, un estudiante dedicó un tiempo sustancial, entre 13 y 14 horas, a un programa en particular, lo que podría indicar un alto nivel de interés o la necesidad de una capacitación más exhaustiva. La figura 4 presenta el promedio de errores inyectados al código, se destaca que la mayoría de los estudiantes tienen un promedio de errores que oscila entre 6 y 10, lo que sugiere un nivel moderado de precisión en su trabajo. Además, 6 estudiantes presentan un promedio de errores más bajo, entre 1 y 5, indicando un mejor rendimiento. Por otro lado, 5 estudiantes tienen un promedio de errores en el rango de 11 a 15, lo que podría requerir una atención adicional para mejorar la calidad de su trabajo.

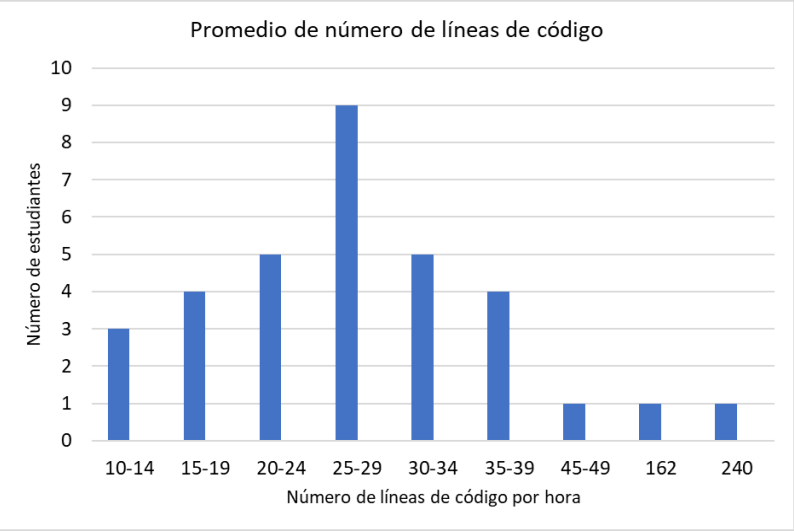

Figura 2. Promedio de número de líneas de código

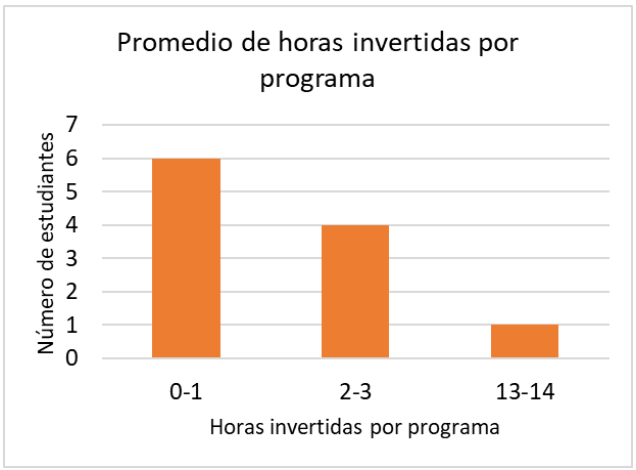

Figura 3. Número de horas invertidas

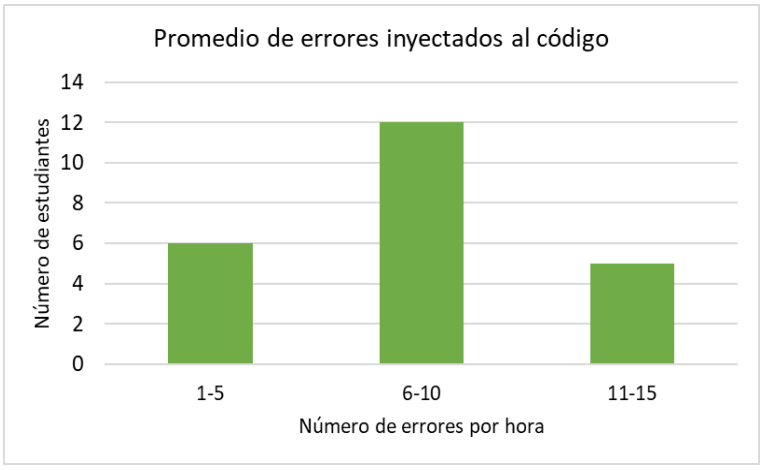

Figura 4. Cantidad de errores inyectados

Es importante mencionar que se observaron datos inexactos. Por ejemplo, en la figura 2, se registraron valores de 162 y 240 líneas de código por hora, los cuales no parecen realistas. Al investigar más a fondo, se descubrió que los estudiantes que obtuvieron estos resultados inicialmente tomaron la medida en minutos y concluyeron que podían escribir 2.6 o 4 líneas de código por minuto. Esta extrapolación a horas resultó en valores poco precisos. Se especula que, durante la medición, no tuvieron en cuenta el tiempo dedicado a analizar el problema, diseñar la estructura del código, entre otros aspectos. Esto llevó a que finalizaran rápidamente cada programa, que les tomaba entre 10 y 20 minutos, lo cual no refleja una medida precisa de su productividad.

Además, surgieron otras medidas que no fueron tan usadas por los estudiantes pero que sería valioso incorporar. Dos ejemplos que destacan son el tiempo invertido en el diseño y el tiempo invertido en hacer seudocódigo. Los estudiantes que usaron estas métricas mostraron mayor productividad al completar el código y también mostraron una visión más realista y clara de cómo mejorar sus tiempos. Estas prácticas les han aportado mucho,

especialmente considerando que son ejercicios de primer semestre, que, aunque son simples, tendrán un impacto aún mayor en proyectos de mayor escala. Este trabajo les ayuda a comprender la importancia de este tipo de prácticas y cuánto les beneficiarán. Otra medida relevante es el tiempo de interrupción según lo reportan los estudiantes. Algunos estudiantes midieron no solo cuánto tiempo dedicaron a realizar las tareas, sino también cuánto tiempo se distraían utilizando el celular, viendo vídeos en YouTube o utilizando redes sociales.
La mayoría de los estudiantes solo lograban en principio expresar sus aspiraciones en cuanto a métricas como líneas de código esperadas. Al comenzar este ejercicio lograron vislumbrar otras formas de medir su desempeño, y con un poco más de documentación lograron proyectar mejor su meta. Es lógico esto porque si hay desconocimiento de herramientas estructuradas para gestionar el trabajo, es complejo comenzar a registrar información y saber cómo emplearla para la mejora continua.

Además de esto, varios estudiantes reconocieron áreas de mejora en su desempeño en programación, incluyendo la procrastinación, la falta de planificación y la necesidad de practicar más. Los estudiantes reconocieron la importancia de implementar el diseño antes de escribir el código y utilizar los recursos disponibles. El uso de PSP se consideró valioso para mejorar sus habilidades. La práctica continua y la resolución eficiente de problemas son cruciales, y todos buscan un progreso constante en su desarrollo como programadores.

Los resultados reflejan que la implementación del PSP produce un impacto positivo en la disciplina y el desempeño de los estudiantes. Este enfoque no solo conduce a una mejora sustancial en la calidad de trabajo de los estudiantes, sino que también facilita un seguimiento más preciso del progreso individual y fomenta una gestión más eficiente y efectiva del trabajo en el contexto del desarrollo de software. Estos hallazgos subrayan la importancia de la adopción de prácticas estructuradas y el uso de métricas en la educación en Ingeniería de Sistemas, brindando beneficios tangibles tanto para los educadores como para los estudiantes.

6. CONCLUSIONES Y TRABAJO FUTURO

Este artículo presenta un estudio donde se usan las métricas de productividad y gestión del trabajo basadas de PSP con estudiantes de Ingeniería de Sistemas en la Universidad EAFIT-Colombia. Hay diversas conclusiones, que se presentarán por cada pregunta orientadora del ejercicio.

En relación con la pregunta ¿Cuáles son las métricas que se utilizarán en el desarrollo de software? Es posible concluir que la mayoría de los estudiantes usó el número de líneas de código por hora, la cantidad de errores y tiempo de desarrollo para medir su desempeño. Aparecieron otras alternativas que los estudiantes no emplearon con frecuencia, pero cuya inclusión resultaría beneficiosa. Dos ejemplos destacados incluyen el tiempo dedicado al proceso de diseño y el tiempo invertido en la elaboración de pseudocódigo. Quienes incorporaron estas métricas demostraron una mayor eficiencia en la finalización del código, además de adquirir una mejor comprensión de cómo optimizar sus tiempos.

Frente a la pregunta ¿Cuál es mi desempeño actual? La mayoría de los estudiantes dedicaron 25 a 29 horas a sus actividades, lo que muestra un compromiso considerable.

También hay grupos más pequeños en los rangos de 20 a 24 horas y 30 a 34 horas, que indican un nivel de compromiso. Por otro lado, hubo poca representación en rangos de tiempo como 45 a 49 horas. En cuanto al tiempo invertido en programas, la mayoría dedicó de 0 a 1 hora, mientras que un grupo significativo invirtió entre 2 y 3 horas. Un estudiante destacó al invertir 13 a 14 horas en un programa. En cuanto a errores, la mayoría tuvo un promedio de 6 a 10, algunos tuvieron menos errores (1 a 5), y otros más (11 a 15), lo que podría requerir atención adicional.

Frente a la pregunta ¿Cuál es la meta esperada o deseada? y ¿Cómo espera alcanzar la meta? La mayoría de los estudiantes inicialmente solo expresaban sus expectativas en términos de líneas de código. Sin embargo, al comenzar el ejercicio, descubrieron formas adicionales de medir su desempeño y con más documentación lograron definir mejor sus metas. Esto es comprensible, ya que la falta de familiaridad con herramientas estructuradas dificulta el registro y uso de información para la mejora continua. Además, varios estudiantes identificaron áreas de mejora en su programación, como la procrastinación, la falta de planificación y la necesidad de practicar más. También reconocieron la importancia de diseñar antes de codificar y aprovechar los recursos disponibles. El uso de PSP se consideró valioso para mejorar sus habilidades. Todos buscan un progreso constante en su desarrollo como programadores a través de la práctica continua y la resolución eficiente de problemas.

Por último, ante la pregunta ¿Qué se concluye del ejercicio? El primer hallazgo valioso es que la medición del tiempo dedicado a las tareas junto con el tiempo de distracción proporciona una visión más completa sobre cómo los estudiantes administran su tiempo y cómo esto afecta su productividad. Esta perspectiva holística indica que los estudiantes están reconociendo la importancia de considerar tanto el tiempo de trabajo como el tiempo de distracción para obtener una imagen precisa de su eficiencia.

Los resultados de este estudio pueden ser de gran utilidad para identificar patrones y tendencias en el comportamiento de los estudiantes con respecto a las distracciones digitales. También dan un panorama más exacto del tiempo que le dedican a sus trabajos académicos. Es probable que los denominados “distractores” incidan a favor, si se saben canalizar.

La información recopilada servirá como base para el desarrollo de estrategias destinadas a minimizar o encaminar las interrupciones y mejorar la gestión del tiempo de los estudiantes, lo cual a su vez puede tener un impacto positivo en su eficacia y rendimiento académico, y, por ende, el índice de deserción. Además, los hallazgos revelan que las distracciones digitales, como el uso de dispositivos móviles, ver vídeos en YouTube y utilizar redes sociales, representan una fuente significativa de interrupción para los estudiantes. También se puede ratificar lo expresado en estudios previos, en cuanto a mejoras significativas en la calidad del código, la estimación precisa de tareas y la reducción de defectos en el trabajo de los futuros ingenieros. Se observó también una mayor conciencia sobre el manejo de métricas [Soto & Rafael, 2020; Contreras-Vas et al., 2021].

Como trabajo futuro se espera realizar un estudio longitudinal para evaluar los efectos a mediano plazo de la aplicación de PSP en la educación de Ingeniería de Sistemas. Esto implica monitorear a una cohorte de estudiantes durante varios semestres (en cursos como Ingeniería de Software y Proyecto Integrador) para evaluar su mejora en la estimación del

tiempo de finalización de tareas, la gestión del proceso de desarrollo de software y la mejora del rendimiento general. También se puede dirigir hacia el trabajo de (proceso de software en equipo) TSP. Además, nos interesa colaborar con profesionales de la industria del software para evaluar la transferibilidad de las habilidades y prácticas de PSP desde entornos académicos hasta entornos de desarrollo de software del mundo real. Sería interesante explorar la alineación entre las habilidades adquiridas a través de PSP en el contexto académico y las expectativas y requisitos de los profesionales de la industria.

REFERENCIAS

Shen, W.-H., Hsueh, N.-L., & Lee, W.-M. (2011). Assessing PSP effect in training disciplined software development: A Plan–Track–Review model. Information and Software Technology, 53(2), 137-148.

Razzaq, A., Ahmad, S., Khalil, A., & Ahmad, S. (2020). Systematic Mapping Study: Augmenting Personal Software Process Analysis For Extreme Programming Teams. Authorea Preprints.

Ramingwong, S., & Ramingwong, L. (2012). Implementing a personal software process (PSPSM) course: a case study. Journal of Software Engineering and Applications, 5 (8), 639-644.

Soto, B. P., & Rafael, G. R. (2020). Habilidades de Personal Software Process (PSP) para la industria del software en Latinoamérica. Industrial data, 23(1), 229-244.

Kusakabe, S., Araki, S., Katamine, K., & Umeda, M. (2020). Analyzing Motivation in Personal Software Process Education Course with a Qualitative Approach. In 2020 9th International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 298-303). IEEE.

Contreras-Vas, J. Á., Arias-Masa, J., Hidalgo-Izquierdo, V., & Martín-Espada, R. (2021). Personal Software Process: A Study from an Academic Perspective. In Computer Supported Qualitative Research: New Trends in Qualitative Research (WCQR2021) 5 (pp. 143-154). Springer.

Contreras, J. Á., Arias, J., Hidalgo, V., & Martín, R. (2020). Análisis cualitativo de las respuestas de los alumnos sobre aspectos del curso del Proceso de Software Personal. New Trends in Qualitative Research, 4, 304-316.

Arduino, G. A., & Bollati, V. A. (2020). Proceso Personal de Software aplicado a la formación universitaria: una revisión sistemática de literatura. In 2020 IEEE Congreso Bienal de Argentina (ARGENCON) (pp. 1-7). IEEE.

Borstler, J., Carrington, D., Hislop, G. W., Lisack, S., Olson, K., & Williams, L. (2002). Teaching PSP: Challenges and lessons learned. Ieee Software, 19(5), 42-48.

Venkatasubramanian, K., Roy, S. B. T., & Dasari, M. V. (2001). Teaching and using PSP in a software engineering course: an experience report. In Software Engineering Education and Training Annual Conference.

Maletic, J. I., Howald, A., & Marcus, A. (2001, February). Incorporating PSP into a traditional software engineering course: an experience report. In Proceedings 14th Conference on Software Engineering Education and Training ('In search of a software engineering profession') (pp. 89-97). IEEE.

NOTAS BIOGRÁFICAS

Andrés Echeverri Jaramillo es Estudiante de ingeniería de sistemas en la universidad de EAFIT, actualmente está como practicante en CodeBulls S.A.S.

Paola Vallejo es Doctora en Ciencias de la Computación. Actualmente, es docente e investigadora en la Universidad EAFIT en Medellín, Colombia. Imparte asignaturas relacionadas con Ingeniería de Software. Su interés se enfoca en arquitecturas de software, diseño de software y código limpio.

Liliana Gonzalez Palacio es Doctora en ingeniería. Actualmente, es docente e investigadora en la Universidad EAFIT en Medellín, Colombia. Imparte asignaturas relacionadas con Ingeniería de Software. Su interés se enfoca en ingeniería de requisitos, educación gamificada en ingeniería, tecnología para población con discapacidad.

Rafael David Rincón Bermúdez es Magister en Sistemas de Calidad del Tec de Monterrey, México, Magister en Matemáticas Aplicadas de la U Eafit de Medellín, Colombia. Certificado como Data Scientist en Big Data por ARCITURA. Consultor en temas de Analytics, Definición, Mejora y Automatización de Procesos de Negocio, Automatización Robótica de Procesos con RPA, Análisis de Información, Implementación de Sistemas de Calidad, Modelos y Estándares Internacionales de Calidad de Software, como CMMI (Capability Maturity Model Integration), Competisoft, ITIL, entre otros. Imparte asignaturas relacionadas con Calidad de Software, Estadística y Transformación Digital. Su interés se enfoca en Automatización con RPA, Inteligencia Artificial, Analítica de Datos, EcoSistemas Digitales.