Encuentro Internacional de Educación en Ingeniería ACOFI
Una formación de calidad en ingeniería para el futuro
Centro de Convenciones Cartagena de Indias
15 al 18 de Septiembre de 2015

IMPLEMENTACIÓN DE UNA ESTRATEGIA DE ENSEÑANZA PARA LA ADOPCIÓN DE PRÁCTICAS DE PSP EN UN CURSO DE FUNDAMENTOS DE ALGORITMIA DEL PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN DE LA UNIVERSIDAD DEL QUINDÍO

Jorge Orlando Herrera Morales, Sergio Augusto Cardona Torres

Universidad del Quindío
Armenia, Quindío

Resumen

Para la sostenibilidad y el progreso de la industria del software en Colombia, se requiere de profesionales con capacidades de aplicar buenas prácticas y estándares internacionales de calidad del software, que soporten la construcción de solucionesinformáticas de acuerdo a los requerimientos de las organizaciones. La incorporación de buenas prácticas para el desarrollo de software mejora la capacidad y la productividad en las organizaciones de Tecnologías de las Información. Sin embargo, esto puede significar altas inversiones en tiempo y capacitación, para el recurso humano de estas empresas.

Desde los programas de ingeniería de sistemas y afines en Colombia, se tiene la responsabilidad de formar profesionales con competencias para la autogestión y autorregulación de su proceso personal de software. En la actualidad son varias las Universidades que en sus currículos incorporan modelos de procesos orientados a la calidad. Los estudiantes durante su formación profesional aplican buenas prácticas de desarrollo de software: gestión del proyecto, estimación de tamaño del producto, costos y tiempos, manejo de estándares, remoción y prevención de defectos. La gestión del trabajo personal del desarrollo de software contribuye a que el ingeniero administre un proceso definido, medido y controlado con criterios de calidad.

El Personal Software Process (PSP) es un proceso de desarrollo de software definido para cada ingeniero. Dicho proceso orienta y brinda soporte al Ingeniero para la construcción de productos de calidad. El PSP es también un complemento de formación, que propende por una cultura de calidad en el desarrollo de software

El presente trabajo presenta los resultados preliminares de la implementación de una estrategia de enseñanzaorientada a la adopción de algunas prácticas de PSP en un curso de fundamentos de algoritmia del programa de Ingeniería de sistemas y computación de la Universidad del Quindío – Colombia. La estrategia de enseñanza tuvo como propósito la introducción de técnicas individuales de desarrollo de software orientadas al desarrollo de habilidades en aspectos de planeación, estimación de tiempo y gestión de defectos de software.

Palabras clave: proceso personal de software; planeación; administración del tiempo; gestión de defectos; calidad de software; estrategia de enseñanza

Abstract

For the sustainability and progress of the software industry in Colombia, it is required to have professionals with skills to apply good practices and international quality standards of software that support the construction of computing solutions, according to the increasingly complex requirements of the organizations. The inclusion of good practices for software development improves the capacity and productivity in organizations of information technology. However, this could translate into high investments of time and training for the human resources of these organizations.

From the systems and computer engineering programs and related ones in Colombia, it is our responsibility of training professionals with skills for the self-management and self-regulation of their personal software process. Currently, there are several universities that incorporate quality-oriented model processes in their curricula. Students, during their professional training, apply good practices in software development: project management, appraisal of the product size, cost and time, management standards, and removal and prevention of defects. The management of personal work on software development helps the engineer to administer a defined, measured and controlled process with quality criteria.

The Personal Software Process (PSP) is a software development process defined for each engineer. This process guides and supports the engineer to build quality products. Besides, The PSP is also a training complement, which aims for a culture of quality in software development.

This paper presents the preliminary results of the implementation of an education strategy oriented towards the adoption of some PSP practices into an algorithms fundamentals course of the systems and computer engineering program at the University of Quindío - Colombia. The teaching strategy had as a purpose the introduction of individual techniques of software development aimed at the development of skills in aspects of planning, time estimation and management of software defects.

***Keywords**: personal software process; planning; time management; defect management; software quality; teaching strategy*

1. Introducción

La agenda interna para la productividad y competitividad del Gobierno Nacional, se fundamenta en actividades de desarrollo económico diferenciadas, que reconozcan y atiendan las particularidades de cada región(Departamento Nacional de Planeación, 2007). Uno de los segmentos emergentes en las actividades económicas del país es la industria del software, en la cual se han identificado las oportunidades de incursionar en el sector de productos de productos y servicios, a nivel nacional e internacional. Ante esto, desde lo público se han desarrollado estrategias para la consolidación de la industria del software. Una de estas estrategias está orientada a la formación del recurso humano mediante la capacitación en estándares de calidad internacionales, entre las que se destacan el PSP, el cual es un modelo orientado a incrementar la productividad individual en el proceso de desarrollo de software, mediante buenas practicas.

En la actualidad los sistemas informáticos se constituyen en una de las bases estratégicas para las organizaciones, razón por la cual cada vez más las aplicaciones de software deben responder a requerimientos complejos y de carácter critico en las organizaciones. Es por ello que la industria del software requiere de recurso humano con competencias para incorporar buenas prácticas de desarrollo de software basadas en estándares internacionales de calidad, para dar respuesta a las necesidades de las organizaciones.

El programa de Ingeniería de Ingeniería de Sistemas de la Universidad del Quindío, tiene como propósito implementar estrategias de formación orientadas al desarrollo de habilidades que potencien prácticas individuales de software, buscando que el estudiante pueda gestionar un proceso definido, medido y controlado, con criterios de calidad.

En este artículo se presentan los resultados preliminares de una investigación exploratoria en la cual se aplicó una estrategia de aprendizaje en la cual se incorporaron algunas prácticas de PSP en un curso de fundamentos de algoritmia. El propósito era que los estudiantes aplicaran técnicas individuales relacionadas con la gestión del tiempo y defectos, y la planeación. Los resultados mostraron que los estudiantes apropiaron de forma significativa las prácticas de gestión del tiempo y gestión de defectos.

2. Antecedentes

El PSP fue creado por Watts Humphrey(Humphrey, 1995) con el propósito que los ingenieros de software mediante un enfoque disciplinado construyan programas de software. El PSP está diseñado para ayudar a los ingenieros a organizar y planificar su trabajo, el seguimiento de su rendimiento, la gestión de defectos, y analizar y mejorar su proceso de desarrollo personal (Towhidnejad & Hilburn, 1997). El diseño de PSP se basa en los siguientes principios de planeación y de calidad (Humphrey W. , 1995):

- Los ingenieros deben ejecutar los trabajos basados en sus propios datos.
- Los ingenieros deben utilizar procesos bien definidos y medidos.
- Los ingenieros deben sentirse comprometidos con la calidad de sus productos.
- Los ingenieros deben medir el tiempo invierten en cada proceso, los defectos que inyectan y remueven de cada proyecto, y medir el tamaño de los productos.

esde que fue creado el PSP, se han realizado diferentes investigaciones relacionadas con el impacto de PSP sobre la formación de los estudiantes (Towhidnejad & Hilburn, 1997). Se han realizado experimentos en cursos de programación a nivel de pregrado (Abrahamsson & Kautz, 2002), y a nivel de posgrado en cursos avanzados de programación (Prechelt & Unger, 2001). El PSP también ha sido utilizado para experimentar en cursos de Ingeniería de Software (Honig, 2008), (Venkatasubramanian, Roy, & Dasari, 2001). Así mismo, se tiene el reporte de lecciones aprendidas de la implementación de PSP en el sector académico con el apoyo de la industria de software (El Eman, Shostak, & Madhavji, 1996), (Rincón, 2010) . También se tienen experiencias académicas relacionadas con PSP a nivel industrial (Hayes, 1998).

Para el análisis de trabajos relacionados con la experiencia de PSP en la academia, se retoman los tres factores principales que influyen en la enseñanza de PSP (Börstler et al., 2002): entorno de trabajo, cobertura yherramientas. El entorno de trabajo se refiere a la población objetivo. La cobertura contempla el alcance de las prácticas de PSP usadas, las cuales pueden ser completas (Full PSP) o parciales (PSP Lite). Las herramientas son el medio usado para el registro de la información del proceso de desarrollo. Se adiciona un factor relacionado con la aplicación o no de una estrategia de aprendizaje. En la tabla 1 se presentan trabajos relacionados con la aplicación de PSP en diferentes universidades.

Universidad	Entorno de trabajo	Nivel de cobertura	Herramientas de soporte	Estrategia de enseñanza
Lund University – Suecia	Estudiantes de Pregrado y Posgrado	Full PSP	Hojas de Cálculo	No se hace referencia
Zagreb University – Croacia	Estudiantes de Pregrado	PSP-Lite	PSP-Tool	No se hace referencia
PurdueUniversity - USA	Estudiantes de Pregrado	PSP-Lite	Hojas de Cálculo	No se hace referencia
Embry-Riddle Aeronautical University – USA	Estudiantes de Pregrado	Full PSP	Formularios de PSP	Si se hace referencia
Universidad Carlos III– España	Estudiantes de Pregrado	PSP-Lite	PSP Student Workbook	Si se hace referencia

Tabla 1. Factores que influyen en la enseñanza de PSP

Cada experiencia reportada presenta restricciones, particularidades y metodologías muy particulares de acuerdo a su contexto. Lo que evidencia el interés por la investigación educativa en el campo de las buenas prácticas en desarrollo de software.

3. Metodología

La metodología de investigación está basada en un diseño cuasi experimental descriptivo, con diseño intergrupos con grupo de control y experimental, a los cuales se les aplica medida de pretest y postest. La población objeto de estudio se constituyó de dos grupos de fundamentos de algoritmia del programa de ingeniería de Sistemas y computación de la universidad del Quindío. El propósito de la investigación fue determinar si la implementación de una estrategia de enseñanza orientada a adoptaralgunas prácticas de PSP en un curso de fundamentos de algoritmia aporta al desarrollo de prácticas individuales de desarrollo de software en los estudiantes del grupo experimental.

3.1 Pretest

Para determinar la homogeneidad de los grupos de control y experimental se aplicó un instrumento depretest, con el propósito de identificar el nivel de adopción de algunas prácticas individuales de desarrollo por parte de los estudiantes de ambos cursos. El instrumento contenía 12 preguntas en una escala de Likert de 1 a 5. La cantidad de estudiantes participantes fueron 13 del grupo experimental y 19 del grupo de control.
Las categorías en las cuales se agruparon las preguntas fueron: gestión del tiempo, manejo de defectos y planeación del proyecto. Para cada una de las categorías se definieron las preguntas que se presentan en la tabla 2.

Categorías	Preguntas
Gestión del tiempo	1. Registra el tiempo que invierte en el desarrollo de un programa. 2. Registra el tiempo de las interrupciones en el trabajo de desarrollo. 3. Estima el tiempo que debe tardar el desarrollo de un programa 4. Calcula el tiempo total que tardó en realizar un programa.
Gestión de defectos	5. Registra los defectos que ha introducido al escribir un programa. 6. Entiende los defectos que ha introducido al escribir un programa. 7. Clasifica los defectos cuando se encuentra desarrollando software. 8. Cuenta la cantidad de defectos que encuentra al programar.
Planeación del proyecto	9. Planifica las actividades con las que va a desarrollar un programa. 10. Conoce y aplica las fases del proceso de software cuando va a realizar un programa. 11. Reúne información de los proyectos de software (tiempo, tamaño en líneas de código y la cantidad de errores corregidos) para la planeación de proyectos futuros. 12. Documenta los problemas encontrados en el proyecto para corregirlos en futuros trabajos.

Tabla 2. Preguntas aplicadas para pretest y postest

La homogeneidad se determinó mediante un Análisis de Varianza ANOVA cuya variable de respuesta es la calificación de la pregunta y el factor es el grupo con los niveles de control y experimental. Para cada pregunta del instrumento se aplicaron supuestos de aleatoriedad, homogeneidad de varianzas y la distribución normal de residuos. Como ambos supuestos no se cumplieron para todas las preguntas, se aplicó la prueba No paramétrica de Kruskay y Wallis. En la tabla 3 se presentan los resultados estadísticos.

Pregunta	p_valor	Homogeneidad
p1	0,168267	Si
p2	0,820911	Si
p3	0,0862041	Si
p4	0,615115	Si
p5	0,238638	Si
p6	0,441341	Si

Pregunta	p_valor	Homogeneidad
p7	0,238638	Si
p8	0,661241	Si
p9	0,352272	Si
p10	0,0738654	Si
p11	0,241817	Si
p12	0,472331	Si

Tabla 3. Resultados de homogeneidad para pretest

La prueba no paramétrica determinó la homogeneidad de ambos grupos, a partir de lo cual se procedió a la implementación de la estrategia de enseñanza.

3.2 Estrategia de enseñanza

Una estrategia de enseñanza contiene los recursos o procedimientos utilizados por el agente de enseñanza para promover aprendizajes significativos(West, Farmer, & Wolff, 1991). La estrategia de enseñanza se llevó a cabo con 13 estudiantes del grupo experimental durante 6 semanas del semestre académico. Paralelamente al desarrollo del contenido temático, se fueron incorporando los conceptos fundamentales de los niveles PSP0 y PSP 0.1. El grupo se dividió en 3 equipos de trabajo de 5, 4 y 4 estudiantes. Los conceptos relacionados con cada uno de los temas se fueron estudiando por equipos de trabajo, luego un representante de cada equipo realizaba una exposición de lo estudiado por el equipo y se recibía la retroalimentación del profesor. De forma progresiva se incorporaron los conceptos fundamentales de los niveles PSP0 y PSP 0.1

Se propusieron 3 ejercicios de programación, los cuales fueron resueltos directamente en el laboratorio del curso, bajo el seguimiento del profesor. Para los entregables, los estudiantes usaron el log de Registro de tiempos y el log de Registro de defectos para los niveles PSP0, PSP0.1. El formato de manejo de estándares de codificación se usó para los niveles PSP0.1. Para el proyecto final del curso se exigieron las plantillas mencionadas. El registro de cada una de las actividades se realizó en las plantillas diseñadas para tal propósito y la retroalimentación de los resultados se hacía en la siguiente clase, resaltando la importancia de las actividades planteadas.

Para las prácticas del nivel PSP0, se diseñó una guía de cátedra con los fundamentos teóricos necesarios para el aprendizaje y aplicación de las siguientes prácticas:

- Conceptos fundamentales de PSP
- Registro de tiempo para la realización del proyecto.
- Registro de los defectos y de los tipos de defectos, mediante el estándar de PSP.
- Resumen del plan de proyecto.

Para cada una de los temas de PSP vistos, se definió un plan de evaluación basado en criterios, en donde se tuvieron en cuenta aspectos como:

- Observación de las actitudes los estudiantes.
- Seguimiento al desarrollo de las prácticas que realizan los estudiantes en el laboratorio y durante su trabajo independiente.

- Realización de evaluaciones de forma individual.

4. Resultados

Con el objetivo de constatar si la intervención con las prácticas de PSP en el grupo experimental fue exitosa, se verificó mediante una prueba no paramétrica, si en el postest se conservaba la propiedad de homogeneidad con el grupo de control en cada una de las preguntas del instrumento. El profesor aplicó el postest la última semana del semestre.

4.1 Postest

Los resultados del postest, muestran que la propiedad de homogeneidad de los grupos se conserva para las preguntas 9,10, 11; de lo cual se puede afirmar que la estrategia de enseñanza no tuvo una incidencia significativa en las prácticas relacionadas con la planeación del proyecto. Para las categorías de gestión de tiempo y gestión de defectos, se encontró que la propiedad de homogeneidad entre los grupos no se conserva y por lo tanto se puede afirmar que para estas la estrategia de enseñanza fue exitosa.

4.2 Resultados grupo experimental

Los resultados de aplicar la estrategia de enseñanza en el grupo experimental para la gestión de tiempo muestran que un 61,5% de los estudiantes registra el tiempo que invierte cuando programa. El 46,1% registra el tiempo de sus interrupciones al momento de programar. El 53,8% hace estimaciones de tiempo para construir un programa. El 61,5% calculó el tiempo total invertido para resolver los problemas planteados.

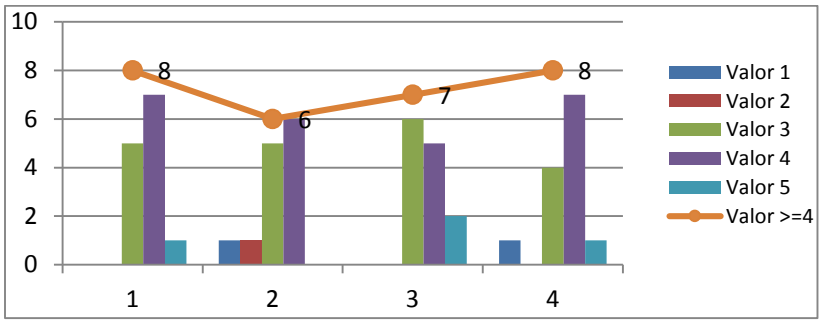

Figura 1. Preguntas aplicadas para pretest y postest (gestión de tiempo)

Con relación a la gestión defectos el 69,2% de los estudiantes registra y entiende los defectos que ha introducido al momento de escribir un programa. Solo un 46,1% de los estudiantes clasifica los defectos de acuerdo a una taxonomía y el 6,6% de ellos cuenta los defectos que encuentra al momento de programar.

Para la categoría de planeación se identificó que sólo el 38,4% de los estudiantes planifica las actividades para desarrollar un programa. Así mismo, después de la intervención el 46,1% de los estudiantes afirma conocer y aplicas las fases del proceso

de desarrollo de software. Para la planeación de proyectos futuros, el 33,3% de los estudiantes reúne información relacionada con el manejo del tiempo y la administración de los errores. Finalmente, se encontró que el 53,8% de los estudiantes documenta los problemas identificados en el proyecto para corregirlos en futuros proyectos.

5. Conclusiones

La implementación de la estrategia de enseñanza mostró que la apropiación de buenas prácticas a nivel individual, está relacionada con la motivación y el interés de los estudiantes, para que estos identifiquen la importancia de conocer y apropiar un proceso de desarrollo de software desde los primeros semestres de su formación profesional. Las actividades de la estrategia permitieron identificar los errores y las oportunidades de mejora, para que durante el curso se les incentivara a que su trabajo estuviera en correspondencia con los criterios de calidad esperados.

Los estudiantes del grupo experimental manifestaron al inicio del semestre que PSP implica tiempo adicional en su proceso de desarrollo, percibiéndolo como un llenado de formatos sin valor agregado. Esta percepción está en correspondencia con resultados de investigaciones previas. A pesar de ello, en la medida en que los estudiantes fueron realizando las actividades planteadas, las mismas se incorporaron incrementalmente, llegando a ser realizada de forma natural por un número importante de los estudiantes del grupo experimental. El registro de los defectos resultó difícil para los estudiantes debido a que muchos de ellos no identificaban el tipo de defecto y se encontró que estos siempre eran ubicados en las mismas categorías.

Con base en los resultados de la intervención al grupo experimental mediante la estrategia de enseñanza fue satisfactoria en lo que refiere a la gestión del tiempo y de los defectos. El desarrollo de este trabajo mostró una serie de desafíos, pues desde lo pedagógico se puede afirmar que el éxito de estas experiencias académicas está asociado con la madurez de los estudiantes, para que reconozcan el valor de una disciplina aplicada a un proceso de programación (cuestión que todavía no han experimentado en su formación).

6. Referencias

- Abrahamsson, P., & Kautz, K. (2002). Personal Software Process : Classroom Experiences from Finland. *Lecture Notes in Computer Science*, *2349*, 175–185.
- Börstler, 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.
- Departamento Nacional de Planeación. (2007). Agenda Interna para la Productividad y la Competitividad. Bogotá.
- El Eman, K., Shostak, B., & Madhavji, N. (1996). Implementing Concepts from the Personal Software Process in an Industrial Setting Implementing Concepts from

the Personal Software Process in an Industrial Setting. In *Proceedings of the Fourth International Conference on the Software Process* (pp. 117–131). Brighton.

- Hayes, W. (1998). Using a Personal Software Process to improve performance. *Proceedings Fifth International Software Metrics Symposium*, 61–71. doi:10.1109/METRIC.1998.731227
- Honig, W. L. (2008). Teaching Successful "Real-World" Software Engineering to the "Net" Generation: Process and Quality Win! In *21st Conference on Software Engineering Education and Training* (pp. 25–32). Charleston: Ieee. doi:10.1109/CSEET.2008.38
- Humphrey, W. (1995). *A discipline for software engineering*. Addison-Wesley.
- Prechelt, L., & Unger, B. (2001). An Experiment Measuring the Efects of Personal Software Process (PSP) Training. In *IEEE Transactions on Software Engineering* (pp. 465 – 472).
- Rincón, R. (2010). *Análisis y capitalización de las experiencias y lecciones aprendidas de la implementación de PSP (Personal Software Process) y TSP (Team Software Process) desde el sector académico a las empresas de software mexicanas. Informe Final Sabático.* Medellín.
- Towhidnejad, M., & Hilburn, T. (1997). Integrating the Personal Software Process (PSP) across the Undergraduate Curriculum. In *Frontiers in Education Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change* (pp. 162–168). Pittsburgh.
- 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*. Chennai.
- West, C., Farmer, J., & Wolff, P. (1991). *Instructional Design: Implications from Cognitive Science*. Prentice Hall College.

Sobre los autores

- **Jorge Orlando Herrera Morales** Ingeniero de Sistemas Universidad Autónoma. Aspirante a Magister en Ingeniería Universidad EAFIT. Profesor Asistente del programa de Ingeniería de Sistemas y Computación Universidad del Quindío, Armenia (Colombia). joherrera@uniquindio.edu.co
- **Sergio Augusto Cardona Torres** Ingeniero de Sistemas Universidad del Valle. Magister en Ingeniería EAFIT. Profesor Asociado del programa Ingeniería de Sistemas y Computación de la Universidad del Quindío, Armenia (Colombia). sergio_cardona@uniquindio.edu.co