ESTRATEGIA DE ENSEÑANZA PARA LA APROPIACIÓN DE PRÁCTICAS PSP EN EL CURSO DE PROGRAMACIÓN AVANZADA DEL PROGRAMA DE INGENIERÍA DE SOFTWARE DE LA ESCUELA DE ADMINISTRACIÓN Y MERCADOTECNIA DEL QUINDÍO

ÓSCAR MAURICIO GRANADA MUÑOZ

UNIVERSIDAD AUTÓNOMA DE MANIZALES

FACULTAD DE INGENIERÍA

MAESTRÍA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE

MANIZALES

2020

ESTRATEGIA DE ENSEÑANZA PARA LA APROPIACIÓN DE PRÁCTICAS PSP EN EL CURSO DE PROGRAMACIÓN AVANZADA DEL PROGRAMA DE INGENIERÍA DE SOFTWARE DE LA ESCUELA DE ADMINISTRACIÓN Y MERCADOTECNIA DEL QUINDÍO

Autor

ÓSCAR MAURICIO GRANADA MUÑOZ

Proyecto de grado para optar al título de Magister en Gestión y Desarrollo de Proyectos de Software

Tutor

SERGIO AUGUSTO CARDONA TORRES

UNIVERSIDAD AUTÓNOMA DE MANIZALES

FACULTAD DE INGENIERÍA

MAESTRÍA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE

MANIZALES

2020

RESUMEN

El presente trabajo tuvo como finalidad validar la efectividad del uso de una estrategia de enseñanza en la apropiación de buenas prácticas de desarrollo de software PSP0 y PSP0.1, por parte de estudiantes del curso de Programación Avanzada I del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM.

El proceso inició aplicando una encuesta a todos los estudiantes del programa, que permitió conocer las prácticas de desarrollo que realizaban. El análisis de los resultados facilitó la identificación de dos grupos homogéneos con los cuales trabajar, un grupo experimental y un grupo control.

Se utilizó la metodología de proyectos formativos para realizar la intervención del grupo experimental, permitiendo estructurar el proceso de formación y el sistema de evaluación a través de rúbricas.

Se propusieron dos proyectos relacionados con la temática del curso, durante toda la intervención se realizó la retroalimentación de las evaluaciones realizadas y finalmente, se aplicó nuevamente la encuesta de buenas prácticas de desarrollo y un examen de conocimiento, para determinar los resultados de la intervención.

Al terminar el proceso, los estudiantes que participaron en el grupo experimental aventajan a los estudiantes del grupo control a nivel conceptual de PSP y de la cantidad de buenas prácticas que han incorporado en su proceso de programación; el grupo experimental empezó a usar un proceso definido para construir software incorporando algunas prácticas de PSP0 y PSP0.1, mientras que el grupo control no tuvo cambios significativos durante todo el proceso.

Palabras Claves: PSP0; PSP0.1; Ingeniería de software; proyectos formativos; estrategia de enseñanza

ABSTRACT

The purpose of this work was to validate the effectiveness of the use of a teaching strategy in the appropriation of good software development practices PSP0 and PSP0.1, by students of the Advanced Programming I course in the Software Engineering program of the Escuela de Administración y Mercadotecnia del Quindío EAM.

The process began with the application of a survey to all students of the program, which allowed to know the development practices that they use. The analysis of the results helped to identify two homogeneous groups to work with, an experimental group and a control group.

The methodology of formative projects was used to execute the intervention of the experimental group, this allowed to structure the process and the evaluation system through rubrics.

Two projects associated with the course were worked on, feedback was made on the evaluations, the survey of good development practices was applied again and a knowledge test was executed to determine the results of the intervention.

At the end of the process, the students who participated in the experimental group have an advantage over the students in the control group, at the conceptual level of PSP and the amount of good practices they have incorporated into their programming process; the experimental group began to use a defined process to build software incorporating some practices of PSP0 and PSP0.1, while the control group didn't have significant changes throughout the process.

Keywords: PSP0; PSP0.1; Software Engineering; Formative Projects; Teaching Strategies

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

1 PRESENTACIÓN

El desarrollo de software es una actividad que requiere de recurso humano que reúna cada vez más competencias y habilidades con el fin de satisfacer con las necesidades y requerimientos de las aplicaciones de software de la actualidad. Estas habilidades no sólo se componen de aspectos técnicos como lógica para la solución de problemas o la sintaxis de los lenguajes de programación; sino que también es necesario poseer habilidades blandas, que permitan hacer una administración del trabajo por parte del mismo programador. Las prácticas propuestas por Watts Humphrey para el PSP (Personal Software Process) o Proceso Personal de Software en español (W. S. Humphrey, 2001), pueden contribuir en la consecución de ese objetivo.

PSP promueve una mejora en el trabajo de los programadores que aplican estas prácticas en sus procesos de construcción de software, permitiéndoles estimar y planificar sus actividades, controlar su rendimiento según su planeación y mejorar la calidad de sus programas (Soto & Reyes, 2010). El desarrollo de estas competencias en etapas tempranas de la formación de los programadores, permite crear una cultura de disciplina personal y laboral a la hora de construir aplicaciones de software, facilitar su incorporación al medio laboral y ejecutar un trabajo más predecible y menos riesgoso.

La academia tiene un papel muy importante en la formación de los programadores, a quienes debe instruir en estándares y tecnología de vanguardia, que les permita expandir su campo de acción laboral (Soto & Reyes, 2010); la instrucción en PSP se convierte en un valor agregado que las personas que desarrollan software aportan en sus proyectos y en la industria del desarrollo de aplicaciones que se encuentra ávida de productos y servicios con calidad.

El objetivo de la investigación fue proponer una estrategia de enseñanza, construida a partir de la metodología de proyectos formativos, que sea base para que estudiantes de un curso de Programación Avanzada del programa de Ingeniería de Software, en adelante ISW, de la Escuela de Administración y Mercadotecnia del Quindío (EAM), puedan apropiar un

subconjunto de prácticas PSP, orientadas al mejoramiento de su proceso personal de desarrollo de software.

2 ANTECEDENTES

Desde 1995, año en el que Watts Humphrey propuso las prácticas PSP (W. S. Humphrey, 2001), se han realizado diversas investigaciones, estudios y trabajos en torno a este tema en diferentes escenarios, tanto en la academia como en la industria.

La importancia de un desarrollo de software con calidad, motiva a las universidades y sus investigadores a buscar las mejores formas de instruir a los estudiantes en su formación, con el fin que adquieran las competencias que requieren para su futuro desempeño profesional.

En la tabla 1, que se encuentra a continuación se relaciona una descripción de algunas de las investigaciones realizadas y que sirven como referencia para este trabajo.

Tabla 1 Investigaciones relacionadas presentadas en la academia

Autor	Trabajo	Universidad	País	Nivel PSP	Estrategia enseñanza	Nivel estudiantes	Curso oficial PSP	Objetivo
(Manrique & Anaya, 2012)	Estudio empírico de aplicación de PSP para el desarrollo transversal de competencias de gestión, en estudiantes de un programa de Tecnología en Sistemas	Corporación Universitaria Adventista	Colombia	PSP0	No refiere	Estudiantes de Tecnología en sistemas.	Guiones PSP0 adaptados para etapas: Compilación y pruebas unitarias para estudiantes de primeros semestres, para los demás fueron los guiones del curso oficial. Registro de tiempos y defectos con PSP Student Workbook del SEI. Formatos PSP simplificados para primeros semestres. Métricas básicas PSP.	Contrastar resultados de aplicar PSP0 simultáneamente en cursos con estudiantes que están empezando el proceso de formación en lógica y programación y estudiantes que ya han superado esa etapa inicial de formación aplicando la metodología GQM.
(S. A. Cardona, 2011)	Diseño de una estrategia de aprendizaje para implementar prácticas de PSP y TSP en cursos básicos de programación. Caso programa de ingeniería de sistemas y computación Universidad del Quindío	Universidad del Quindío	Colombia	Subconjunto de prácticas PSP (todos los niveles) y TSP	Estrategia de enseñanza construida a partir de la investigación realizada.	Estudiantes de pregrado	Se seleccionan un subconjunto de métricas. Se buscó identificar las prácticas relacionadas con el manejo del tiempo, la gestión de los defectos, las estimaciones de tamaño, la planeación y el trabajo en equipo.	Generar una estrategia de aprendizaje para el mejoramiento del proceso de desarrollo de software tanto individual como grupal, mediante apropiación de prácticas de PSP y TSP en cursos básicos de Programación y Algoritmia.
(Rincón, 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	5 Universidades de diferentes ciudades de México	México	Todos los niveles PSP y TSP.	No refiere	Estudiantes de pregrado, estudiantes de posgrado y trabajadores de la industria del software	Refiere iniciativas y acuerdos realizados en conjunto entre las universidades y la industria mexicanas y el SEI (Software Engineering Institute), quienes acceden a los cursos oficiales de PSP y TSP.	Describe como a través del gobierno mexicano, se apoya la generación de software, haciendo énfasis en políticas que capacitan el recurso humano en adopción de modelos de calidad (como PSP y TSP) a través de iniciativas como Mexicofirst y acuerdos con el SEI.

(Venkatasubramanian et al., 2001)	Teaching and Using PSP in a Software Engineering course: An Experience Report	Birla Institute of Technology and Science	India	Diversos elementos de PSP. No refieren niveles	No refiere	Estudiantes de pregrado	No refieren si utilizaron el curso oficial o fue adaptado.	Ayudar a los estudiantes a desarrollar buenos hábitos de desarrollo de software de manera temprana y motivarlos para que el desarrollo de software sea visto como una disciplina sistemática antes que como una actividad de ensayo y error
(Dymenstein, Etchamendi, & Maidana, 2011)	Towards a student oriented approach to teaching PSP discipline	Universidad Politécnica de Madrid (UPM) - Universidad ORT Uruguay (ORT)	España y Uruguay	Todos los niveles PSP	No refiere	Estudiantes de pregrado	Curso de PSP adaptado. Se introdujeron trabajos diferentes a los del curso oficial.-	Presenta un curso de formación de PSP adaptado que está dirigido a mejorar la experiencia de los estudiantes. El objetivo principal es enseñar los valores fundamentales de disciplina, el profesionalismo y la mejora continua.
(Bermón-angarita et al., 2009)	Experiencias Docentes en la Aplicación del Proceso Software Personal en Primero de Grado de Ingeniería Informática	Universidad Carlos III de Madrid	España	PSP0, PSP0.1, PSP1 y PSP1.1	Estrategia de aprendizaje	Estudiantes de pregrado	No refieren adaptaciones al curso.	Expone los resultados de la formación dada a estudiantes de primer año de Ingeniería informática, de las prácticas sugeridas por PSP, a través de conceptos teóricos y prácticas de evaluación continua con el fin que adquieran disciplina en su labor. Destaca la importancia de registrar métricas para incorporarlas como mejoras.
(Soto & Reyes, 2010)	Introduciendo PSP (Proceso Personal De Software) En El Aula	No refiere	Colombia	NA	NA	NA	NA	Plantea estrategias para introducir de manera gradual e incremental, las prácticas PSP, que dan disciplina en el trabajo individual del programador de software. No se trata de una experimentación en la enseñanza de las prácticas PSP, sino de exponer las ventajas de la preparación de los estudiantes de cursos de algoritmia y programación en dichas prácticas.

(Börstler, Hislop, & Lisack, 2002)	Teaching PSP: Challenges and lessons learned	Umea University (Sweden) University of Queensland (Australia) Drexel University (USA) Purdue University (USA) Utah Valley State College (USA) North Carolina State University (USA)	Suecia Australia USA	PSP1 en Umea University, PSP-Lite y PSP Completo en Utah University y Montana Tech, PSP-Lite en Purdue University y PSP Completo en Drexel University	No refiere	Estudiantes de pregrado	Adaptaciones PSP y cursos oficiales.	El artículo describe las estrategias que siguieron cada una de las universidades que participaron con su experiencia para enseñar PSP en algunos de sus cursos. Concluyen describiendo los principales desafíos encontrados y las lecciones que aprendieron al desarrollar los cursos.
(Towhidnejad & Hilburn, 1997)	Integrating the Personal Software Process (PSP) across the Undergraduate Curriculum	Embry-Riddle Aeronautical University	USA	Todos los niveles PSP	Estrategia de enseñanza	Estudiantes de pregrado	Se simplificó material de estimación y diseño.	El artículo describe algunas de las actividades realizadas en la Universidad Aeronáutica de Embry-Riddle para incorporar las prácticas de PSP en los estudiantes del programa de las ciencias de la computación. Basan su estudio en la brecha existente entre las habilidades y competencias desarrolladas en los estudiantes en las diferentes universidades que los forman y lo que realmente requiere la industria. Enfatizan a los estudiantes la importancia de la calidad en su trabajo.
(Runeson, 2003)	Using Students as Experiment Subjects – An Analysis on Graduate and Freshmen Student Data	Lund University	Suecia	PSP0 a PSP2	No refiere	Estudiantes de pregrado, estudiantes de posgrado y trabajadores de la industria del software	No refieren adaptaciones al curso.	El artículo describe como se experimenta con los estudiantes de ingeniería de software en el contexto de PSP. Buscan comparar estudiantes de primer año, estudiantes graduados e información de trabajadores de la industria.
(Lisack, 2000)	The Personal Software Process in the Classroom: Student Reactions (An Experience Report)	Purdue University	USA	No refieren niveles específicos de PSP	No refiere	Estudiantes de pregrado	No refieren adaptaciones al curso.	Describe la experiencia al enseñar un subconjunto de prácticas PSP en estudiantes de cursos de primer y segundo semestre de programación. Se hicieron entrevistas al terminar los cursos para determinar la actitud de los estudiantes ante PSP. El artículo explora los resultados de estas encuestas.

Fuente: elaboración propia

A continuación, se relacionan y detallan los ejercicios académicos referenciados en la tabla anterior, con sus conclusiones más importantes:

- **Corporación Universitaria Adventista (Colombia)**

Se buscó contrastar los resultados de aplicar PSP0 simultáneamente en cursos con estudiantes que están empezando el proceso de formación en lógica y programación y estudiantes que ya han superado esa etapa inicial de formación aplicando la metodología GQM Goal-Question-Metrics (Solingen & Berghout, 1999).

Con los estudiantes de mayores semestres se obtuvieron mejores resultados en la detección de defectos y la estimación de tiempo. Por su parte, con estudiantes de primeros semestres se generaron mejores resultados en la precisión en la atomización de tareas y su interés posterior de usar la metodología.

En general, en la población intervenida, se mejoraron sus competencias en planeación, control y gestión de la calidad de su proceso de desarrollo de software. Por otro lado, se encontraron inconvenientes con la tipificación de defectos, probablemente por un pobre entendimiento de las categorías.

- **Universidad del Quindío (Colombia)**

Su objetivo fue generar una estrategia de aprendizaje para el mejoramiento del proceso de desarrollo de software tanto individual como grupal, mediante apropiación de prácticas de PSP y TSP en cursos básicos de Programación y Algoritmia.

Al finalizar el estudio, se concluyó que existe la necesidad que los estudiantes asuman roles y responsabilidades relacionadas con la gestión, estimaciones, manejo de estándares, prevención y eliminación de defectos, simulando ambientes como los de la industria, incorporando prácticas que los habitúen al día a día normal de un ingeniero.

Se enfatizó en mostrar la importancia de abordar el desarrollo de software como un proceso complejo con un conjunto de actividades de ingeniería y que no sólo se depende de las competencias para codificar programas.

En cuanto a la asimilación de los contenidos, se destacó la apropiación conceptual y las prácticas relacionadas con administración y gestión del tiempo y de los defectos. Se pudo evidenciar que no se obtuvieron buenos resultados al estimar el tamaño, el trabajo individual y la planeación del proyecto y del trabajo en equipo. De igual forma es importante identificar como un riesgo la posibilidad que no se le dé la suficiente importancia a las ventajas que tienen la recopilación de datos y el análisis de la información obtenida.

- **Birla Institute of Technology and Science (India)**

Su objetivo fue ayudar a los estudiantes a desarrollar buenos hábitos de desarrollo de software de manera temprana, y motivarlos para que vieran el desarrollo de software como una disciplina sistemática en vez de verla como una actividad de ensayo y error.

Los estudiantes concluyeron que era importante realizar el registro de datos, no obstante no gustarles hacerlo. Destacaron los registros de tiempo y defectos. Algunos confesaron que no les gustó PSP. Es importante buscar la motivación de los estudiantes y que se enfoquen en los principales elementos del proceso. Mejoraron indicadores como la productividad medida en la cantidad de líneas de código que construían, la reducción de los tiempos de desarrollo y la densidad de defectos por KLOC (KiloLines Of Code), es decir cada 1000 líneas de código.

- **Universidad politécnica de Madrid (España) y Universidad ORT (Uruguay)**

Presentó una experiencia con un curso de formación de PSP adaptado que estaba dirigido a mejorar la experiencia de los estudiantes. El objetivo principal era enseñar los valores fundamentales de disciplina, el profesionalismo y la mejora continua.

Como una medida del éxito de la experiencia de adaptación, se demostró que fue posible desarrollar en los estudiantes, habilidades de medición del proceso. En concreto, se muestra

que la adaptación tiene similares resultados en productividad y estimación del esfuerzo en comparación con cursos tradicionales de PSP.

En contraste, un aspecto que no fue exitoso en el trabajo fue el relacionado con la utilización de los defectos como una métrica de la calidad de los productos generados. Como solución se revisó el punto con los estudiantes y se propusieron algunas mejoras para cursos futuros.

- **Universidad Carlos III de Madrid**

Se expusieron los resultados de la formación dada a estudiantes de primer año de Ingeniería informática, de las prácticas sugeridas por PSP, a través de conceptos teóricos y prácticas de evaluación continua con el fin que adquirieran disciplina en su labor. Destacaron la importancia de registrar métricas para incorporarlas como mejoras.

Los estudiantes realizaron un proceso con disciplina en su gestión del tiempo y ejecutaron tareas básicas de ingeniería de software desde el inicio de su formación.

Las calificaciones demostraron que el registro de tiempo y defectos fue una labor entendida. La estimación, planificación y seguimiento tuvieron notas medias dada su complejidad y también debido a los errores presentados en la generación de los programas.

- **Umea University (Sweden), University of Queensland (Australia), Drexel University (USA), Purdue University (USA), Utah Valley State College (USA), North Carolina State University (USA)**

Se describen en este trabajo, las estrategias que siguieron en cada una de las universidades que participaron, exponiendo su experiencia para enseñar PSP en algunos de sus cursos de desarrollo de software. Concluyen describiendo los principales desafíos encontrados y las lecciones que aprendieron al desarrollar los cursos.

Destacan la importancia de enseñar PSP a los estudiantes de sus cursos.

Relacionaron los diferentes tipos de estudiantes que pueden encontrarse en un curso de programación y el texto base que los profesores utilizaron para enseñarlo según estos tipos.

Concluyen que existen 3 factores fundamentales para revisar a la hora de enseñar PSP: el tipo de estudiante, el cubrimiento de los niveles y la herramienta de soporte.

- **Embry-Riddle Aeronautical University**

Se describieron algunas de las actividades realizadas en la Universidad Aeronáutica de Embry-Riddle para incorporar las prácticas de PSP en los estudiantes del programa de las ciencias de la computación. Basaron su estudio en la brecha existente entre las habilidades y competencias desarrolladas en los estudiantes en las diferentes universidades que los forman y lo que realmente requiere la industria. Enfatizan a los estudiantes la importancia de la calidad en su trabajo.

Concluyen afirmando que consideran necesario simplificar y automatizar las formas, los logs y los registros. De manera adicional se establece que se requiere continuar con el ajuste y modificación de las técnicas de enseñanza para adaptar e integrar los conceptos del trabajo realizado.

- **Lund University**

El artículo describe como se experimenta con los estudiantes de ingeniería de software en el contexto de la enseñanza del PSP. Buscan comparar estudiantes de primer año con estudiantes graduados, y adicionalmente con profesionales ya incorporados en la industria.

Se observan mejoras en los tres grupos de experimentación en los diferentes niveles de PSP, en aspectos como la precisión en la estimación y la inyección de defectos.

- **Purdue University**

Describe la experiencia al enseñar un subconjunto de prácticas PSP en estudiantes de cursos de primer y segundo semestre de programación. Se hicieron entrevistas al terminar los cursos para determinar la actitud de los estudiantes ante PSP. El artículo explora los resultados de estas encuestas.

Se concluyó que la población con la cual se realizó el experimento (estudiantes de primer y segundo semestre) no estaba preparada aún para apreciar los beneficios de PSP. Los estudiantes tuvieron inconvenientes con el tiempo que les tomaba realizar los registros. De manera adicional, no se llenaron adecuadamente los formularios y terminaron con una mala actitud frente al proceso.

Los beneficios solo fueron vistos por estudiantes con una formación un poco más avanzada, concluyendo que sería mejor enseñar PSP en semestres más adelantados o en un nivel de posgrado.

Tomando como base las experiencias anteriormente citadas, para el presente estudio se tuvieron presentes los siguientes elementos:

- Los estudiantes que participaron del experimento, eran de un programa de pregrado, tal como la mayoría de las experiencias académicas tomadas como antecedente.
- Se buscó que los cursos del trabajo estuviesen integrados por estudiantes de semestres intermedios, evitando que fueran estudiantes que aún no supieran programar y cuyo esfuerzo estuviese focalizado a obtener dicha competencia, y también que no sean estudiantes que probablemente tengan ya un proceso definido de construcción de software, el cual sea difícil de modificar.
- Las prácticas más comunes que se incorporaron en los trabajos revisados fueron las de niveles PSP0 y PSP0.1, por ello serán estas mismas el foco de esta investigación.
- Se utilizarán los mismos formatos PSP oficiales, pero reescritos buscando que sean más claros y fáciles de entender.

3 ÁREA PROBLEMÁTICA Y PREGUNTA DE INVESTIGACIÓN

La dinámica actual de las organizaciones requiere de soluciones de software que puedan soportar la complejidad de su operación y garanticen estar alineadas con sus objetivos estratégicos, por esta razón las empresas de desarrollo de software deben construir productos basados en procesos formales y disciplinados a partir de estándares internacionales de calidad. Es importante que los países aprovechen el potencial que pueden tener en el sector de la construcción de software, fortaleciéndolo en las capacidades de los equipos de trabajos y las personas. Conscientes de la importancia del recurso humano en la industria del software, se propone el proceso personal de desarrollo de software PSP, el cual pretende contribuir a la aplicación de buenas prácticas al momento de desarrollar software, por parte de los programadores.

El recurso humano en las empresas de TI y específicamente el programador de software, es considerado un recurso crítico e indispensable para alcanzar el éxito de los proyectos de desarrollo. Para ello es necesario que desde las instituciones de educación en sus diferentes niveles de formación, se promuevan prácticas de desarrollo de software de acuerdo a estándares internacionales. Sin embargo, en el ámbito académico son pocas las iniciativas relacionadas con la adopción de modelos o metodologías que mejoren sus competencias en estimación, planeación, calidad en su trabajo, cumplimiento de cronograma y reducción de defectos en sus productos (Rincón, 2010). La ausencia de estas competencias en la formación de los programadores, puede generar la poca obtención de capacidades que les permita hacer un trabajo más predecible y con mayor posibilidad de cumplimiento en el alcance y tiempo de sus asignaciones.

Es común que en los programas académicos relacionados con la informática, se centren en la adquisición de habilidades y capacidades técnicas para la construcción de software (Manrique & Anaya, 2012). Se empiezan a desarrollar programas sin aplicar principios básicos de análisis, solución de problemas y de gestión de sus actividades, y sin realizar una reflexión acerca de la calidad de sus productos. Es por ello que el concepto de calidad no se encuentra debidamente caracterizado y por lo tanto no se posee una cultura de aplicación de buenas prácticas en el proceso de desarrollo de software. La formación de competencias para desarrollar software con calidad es impartida finalizando los programas de pregrado o

en cursos especializados de formación para egresados. Para ese momento, el programador ya posee un proceso de construcción de software definido y puede presentar dificultades en la asimilación e incorporación de nuevas actividades en su proceso. Es importante que esas competencias sean adquiridas por los estudiantes desde el inicio de su formación profesional y en lo posible, en paralelo con la adquisición de las habilidades técnicas (Soto & Reyes, 2010).

Teniendo en cuenta que particularmente PSP implica un método riguroso para la recopilación y el análisis de información, estas acciones no son un proceso sencillo y se corre el riesgo que el estudiante no lo perciba como un elemento que le agregue valor a su formación, viéndolo como un método que no le aporta a su proceso de desarrollo de software personal. En muchas ocasiones las actividades propuestas por PSP pueden ser entendidas como un llenado de formatos, que resultan poco motivadores e interesantes para los estudiantes (S. A. Cardona, 2011).

Son diversas las experiencias académicas relacionadas con la apropiación de prácticas de PSP, cuya meta en común es que estudiantes en etapa de formación adopten este modelo internacional de calidad en la construcción de software, y en las cuales se han registrado inconvenientes como el descrito por (Rincón, 2010) en su investigación en la calidad del entorno mexicano en el desarrollo de software, al concluir que se requiere de gran cantidad de tiempo por parte de los docentes que imparten los cursos relacionados con las prácticas PSP, para dar retroalimentación inmediata de los trabajos y ejercicios de los estudiantes, hacerles un acompañamiento permanente e impartir los temas y conceptos del proceso.

En (Venkatasubramanian, Roy, & Dasari, 2001), se realizó un trabajo orientado a motivar a los estudiantes de un curso de ingeniería de software para que vieran el desarrollo de software como una disciplina sistemática antes que como una actividad que se genera a partir de ensayo y error, encontraron dificultades al tratar de mantener a los estudiantes enfocados en las ideas generales del proceso, generar en ellos la suficiente auto-disciplina, obtener una herramienta de soporte adecuada y en resaltar los diferentes tipos de registro que se realizan en PSP. Por su parte, (Manrique & Anaya, 2012) en la introducción de PSP0 en cursos de la Tecnología de Sistemas de la Corporación Universitaria Adventista en Medellín, Colombia, manifiesta dentro de sus resultados, encontrarse con dificultades de

los docentes para realizar el monitoreo de los estudiantes, dificultad para dar una retroalimentación inmediata y relevante e inconvenientes al ajustar las asignaciones prácticas alineadas con las habilidades reales de los estudiantes. En (S. A. Cardona, 2011), al realizar una prueba piloto de su estrategia de enseñanza de PSP, propuesta para un curso de programación de Ingeniería de Sistemas de la Universidad del Quindío, en Armenia, Colombia, se afirma que los estudiantes no evidenciaron mejoras en la estimación del tamaño del producto, la planeación de actividades, el trabajo en equipo y en la aplicación de estándares de codificación.

Con base en el referencial teórico analizado y las problemáticas allí descritas, se hizo necesario realizar el diseño de una estrategia de enseñanza que permitiera a los estudiantes apropiar las competencias y habilidades, y les encaminara a emplear buenas prácticas en su proceso personal de desarrollo de software, con la calidad como una característica implícita en sus labores como programadores. De manera adicional, en las experiencias académicas anteriormente mencionadas, se evidencia un objetivo común, orientar a los estudiantes en las buenas prácticas de desarrollo que recomienda PSP, sin embargo, carecen de una retroalimentación y métodos de medición orientadas a conocer el nivel de obtención de los logros y la asimilación de las prácticas planteadas por parte de los estudiantes.

¿Cuál sería la estrategia de enseñanza que facilite la asimilación de un subconjunto de prácticas de programación de PSP0 y PSP0.1, en los estudiantes del curso de Programación Avanzada del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío?

4 JUSTIFICACIÓN

Para el año 2013, la industria del software en Colombia tuvo un aporte cercano al 1.7% del PIB, con ventas superiores a los 5 billones de pesos y la generación de más de 50.000 empleos (Hernández, 2013). Estos indicadores son una clara muestra de la importancia que actualmente tiene el sector del desarrollo de software en nuestro país. El Gobierno Nacional Colombiano, ha definido políticas y planes de formación, orientados a favorecer el crecimiento, maduración y consolidación de esta disciplina en el país. Este apoyo se está realizando a través del Ministerio de las Tecnologías de la Información y las Comunicaciones (MINTIC), que está promoviendo iniciativas, convocatorias y programas que buscan fortalecer el sector del software, tales como la formación de recurso humano en TI a través del programa Talento Digital (Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia, 2012), formación en prácticas de calidad TSP/PSP a través de la iniciativa de Fortalecimiento de la Industria TI – FITI (Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia, 2015b), formación en competencias específicas y transversales (Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia, 2015a), convocatoria para el fortalecimiento de la competitividad de la industria TI (Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia, 2013), entre otras.

En el departamento del Quindío, a través del Plan Regional de Competitividad del Quindío – PRCQ (Gobernación del Quindío, 2013), se identifica a la industria del software como uno de los segmentos fundamentales de competitividad, que favorece la articulación de todos los sectores productivos del departamento y que estimula el crecimiento económico local.

Para ser competitivos nacional e internacionalmente, las empresas de la industria del software deben contar con talento humano que posea habilidades y conocimientos en buenas prácticas de desarrollo. Estas habilidades y competencias requeridas, pueden fomentarse a través de la aplicación de un proceso disciplinado, el cual les permita interiorizar la importancia de medir sus resultados, les facilite la gestión de su tiempo, promueva en ellos la planeación de sus actividades, les permita estimar el esfuerzo

requerido para cumplir con sus objetivos, conocer sus errores, los motive a realizar planes de mejora, hacer seguimiento a sus tareas y ejecutar su labor de manera predecible.

Es responsabilidad de la academia, diseñar planes de estudio que permitan formar y potenciar en los estudiantes, cualidades y destrezas que les lleve a interiorizar un proceso personal de construcción de software, a partir de las buenas prácticas sugeridas por PSP. Esta definición de su proceso de desarrollo, debe complementarse con sus habilidades técnicas para el cumplimiento exitoso de sus labores de desarrollo.

Con el desarrollo de este trabajo se pretendió apropiar en los estudiantes una serie de prácticas orientadas a fomentar competencias y habilidades acordes con estándares internacionales. Se debió tener en cuenta que los estudiantes poseían características de adaptación rápida a los cambios, muy visuales en sus procesos de aprendizaje, con apoyo constante de herramientas tecnológicas en su proceso de apropiación y que aprenden haciendo interacción constante y permanente con otros. Se hizo entonces necesario desarrollar estrategias de aprendizaje que potencializaran todas estas características (S. A. Cardona, 2011).

A partir del presente trabajo también se generó un mecanismo que permitió analizar la información generada en la práctica del proceso propuesto, con el objetivo de realizar una retroalimentación a los estudiantes, donde se identificaron inconvenientes con la asimilación y uso de las prácticas, así como también las deficiencias en sus competencias técnicas al momento de construir software; y que promovió por parte de los docentes y encargados de diseñar los programas de carrera, la generación de los planes de mejora para aumentar la calidad de sus procesos de formación en desarrollo de software.

5 REFERENTE TEÓRICO

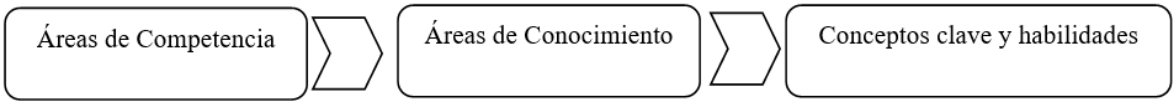
5.1 PSP

Las prácticas PSP (Personal Software Process) fueron propuestas por Watts Humphrey, con el objetivo de mostrar a los estudiantes de ingeniería de software, cuál es la manera de producir programas de alta calidad de acuerdo con una planificación y unos costos. Humphrey estableció una guía para utilizar las prácticas personales disciplinadas de desarrollo de software que muchos ingenieros experimentados utilizan para hacer un trabajo competente (W. S. Humphrey, 2001).

Con el fin de exponer y proveer un material con toda la información relacionada con PSP, la Universidad Carnegie Mellon y el SEI (Software Engineering Institute)(SEI, 2015), construyeron un reporte que tiene como objetivo servir como una guía a los profesionales del software que se encuentren interesados en usar métodos disciplinados para su labor como programadores; este reporte es el PSPBOK (Personal Software Process Body Of Knowledge)(Pomeroy-huff et al., 2009), y contiene la información que se describe a continuación.

Como se puede ver en la figura 1, existen 3 niveles de abstracción en los cuales el PSPBOK descompone conceptos y habilidades relacionados con PSP. Los conceptos son usados para describir aspectos intelectuales del contenido de PSP, es decir, información, terminología, y componentes filosóficos de la tecnología. El término habilidad hace referencia a la capacidad de una persona para interpretar y aplicar los conceptos en la realización de una tarea. Los conceptos clave y habilidades constituyen un área de conocimiento, suponiendo que una persona que entiende el concepto, está en capacidad de desarrollar las habilidades relacionadas con éste. Por último, las relaciones entre áreas de conocimiento constituyen las áreas de competencias.

Figura 1 Estructura PSP



Fuente: elaboración propia

El PSPBOK describe 7 Áreas de competencias, es decir, el conjunto de competencias que debe dominar una persona que adapte su proceso de desarrollo a las buenas prácticas sugeridas por PSP.

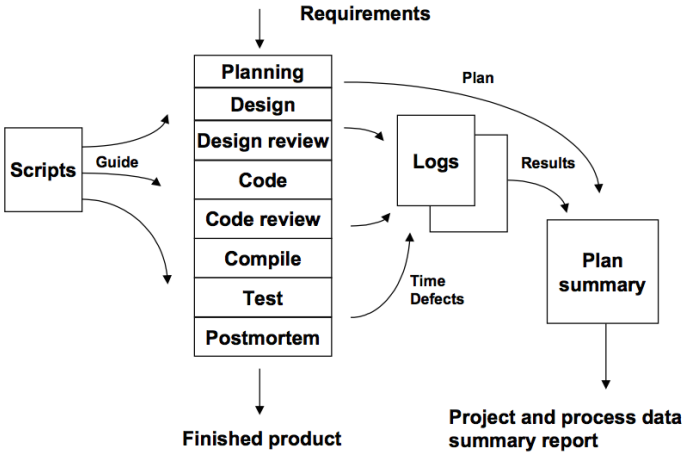
1. Conocimiento fundamental
2. Conceptos básicos de PSP
3. Medición y estimación del tamaño
4. Realización y seguimiento de planes de proyecto
5. Planeación y seguimiento de calidad de software
6. Diseño de software
7. Extensiones y personalizaciones del proceso

Por su parte (W. Humphrey & Over, 2000), describe la estructura del proceso PSP, que guía a los programadores en las actividades que deben realizarse durante todo el ciclo personal de desarrollo de software, y exhibe las etapas que se sugieren realizar, los elementos que componen PSP y la interacción entre ellos.

Típicamente, el proceso inicia al conocer los requerimientos a desarrollar, que son el insumo para realizar la planeación del trabajo. Hay un guión de planeación que guía el trabajo y un resumen de plan para registrar los datos de la planeación. Los programadores van haciendo su trabajo siguiendo el guión correspondiente, y registran los datos de sus tiempos empleados y de los defectos en los logs existentes para esta información. En la última fase del proceso, conocida como Post Mortem, los programadores registran en el formulario de resumen del plan los datos totales de los logs de tiempo y defectos, al igual que la medida del tamaño del programa. Finalmente se tendrá el producto desarrollado y el formulario de resumen del plan.

La figura 2 muestra el proceso.

Figura 2 Elementos PSP



Fuente: (W. S. Humphrey, 2001)

Existen 7 versiones o niveles del proceso PSP, etiquetados de la siguiente manera:

PSP0: Inicia con el proceso actual y agrega medidas básicas.

PSP0.1: Se adiciona el estándar de codificación, una propuesta de mejora del proceso y medida de tamaño.

PSP1: Suma la estimación del tamaño y el reporte de pruebas.

PSP1.1: Añade al proceso la planeación de tareas y de calendario.

PSP2: Adhiere al trabajo las revisiones de diseño y de código.

PSP2.1: Agrega las plantillas de diseño al proceso.

PSP3: Permite generar un proceso de desarrollo cíclico.

Estos niveles se pueden implementar de manera incremental, pues contienen similares conjuntos de elementos que complementan los elementos del nivel anterior. Estos elementos son registros, formularios, guiones y estándares. Los guiones del proceso definen los pasos de cada parte del proceso, los registros y los formularios proveen

plantillas para registrar y almacenar la información, y los estándares guían a los programadores para que sepan cómo hacer su trabajo.

Incluidas en las buenas prácticas propuestas por PSP, existe un método que permite realizar la estimación del tamaño del software que se está construyendo, y es conocido como PROBE (PROxy Based Estimating). Este método usa objetos como proxies (medida sustituta que relaciona el tamaño del producto a una función planeada, provee información en fase de planeación para estimar el tamaño de un producto (Pomeroy-huff et al., 2009)) que componen el desarrollo como su base para hacer las estimaciones del tamaño del software. Los programadores clasifican los objetos según su tamaño probable y la cantidad de ellos que deben construir, usando los datos históricos relacionados. El método usa la regresión lineal para determinar el tamaño final que tendrá el producto.

Teniendo en cuenta la estructura PSP que se ha descrito y las relaciones existentes entre los niveles, los elementos y las fases pre-establecidas, la apropiación de PSP por parte de los estudiantes de ingeniería de software y afines, no ha sido del todo fácil, no obstante la difusión de su conocimiento, la búsqueda de su comprensión y la motivación para su implementación, tanto a nivel académico como industrial. Es decir, aunque se expongan con claridad las ventajas que tiene la utilización de prácticas PSP en el desarrollo de software, su apropiación por parte de los programadores, sean estudiantes o profesionales, es compleja

La academia a nivel mundial, ha realizado diversos estudios relacionados con PSP. Se han dirigido investigaciones que concluyen con la descripción de las experiencias de su implementación en diversos contextos de aplicación, otras que exponen los efectos que genera su implementación en cursos de pregrado y postgrado, se ha buscado evaluar el impacto posterior en la academia, otros en la industria, otros con el uso de PSP en diversos cursos de múltiples instituciones, para experimentos de ingeniería de software, entre otros (S. A. Cardona, 2011).

5.2 ESTRATEGIA DE ENSEÑANZA

Las estrategias de enseñanza son procedimientos que se aplican de modo controlado, dentro de un plan diseñado deliberadamente con el fin de conseguir una meta fijada (Pozo, 2001), es decir, éstas son siempre conscientes e intencionales, dirigidas a un objetivo relacionado con el aprendizaje. Dichas estrategias les permite a los alumnos aprender de manera autónoma y permanente, así como enfrentar con éxito diversas situaciones de aprendizaje que se les presenten en el transcurso de su vida (Valtierra et al., 2007).

En la búsqueda de una pedagogía con estrategia, es indispensable para el docente pensar no sólo en los conceptos y temas que integran su programa, sino también en revisar la forma en la cual los expondrá y cómo los estudiantes los trabajarán en clase. Se puede afirmar entonces que las estrategias de enseñanza tienen dos dimensiones: la reflexiva, en la que el docente diseña su planificación y que involucra desde el pensamiento del docente, el análisis del contenido disciplinar, la consideración de las variables situacionales en las que lo enseña y el diseño de las alternativas de acción, hasta la toma de decisiones acerca de la propuesta de actividades que considera mejor en cada caso; y la dimensión de la acción, que involucra la puesta en marcha de las decisiones tomadas.

Para acompañar el proceso de aprendizaje, es necesario, desde la enseñanza, crear un ciclo constante de reflexión-acción-revisión o de modificación acerca del uso de las estrategias de enseñanza. El docente aprende sobre la enseñanza cuando planifica, toma decisiones, cuando pone en práctica su diseño y reflexiona sobre sus prácticas para reconstruir así sus próximas intervenciones (Anijovich & Mora, 2009).

Finalmente y teniendo definida la estrategia, previo a ejecutarla, es necesario definir y diseñar el tipo, la cantidad, la calidad y la secuencia de actividades que se ofrecerán a los estudiantes (Anijovich & Mora, 2009).

Es importante resaltar que existen diversas formas de enfocar las estrategias de aprendizaje, por ejemplo para (Diaz-Barriga & Rojas, 2002), éstas se clasifican en estrategias de recirculación, que se basa en el repaso y que suponen un aprendizaje superficial, estrategias de elaboración, que integran y relacionan la nueva información con los conocimientos previos pertinentes y las estrategias de organización, que pretenden hacer una

reorganización constructiva de la información que ha de aprenderse. Esta estrategia permite organizar, agrupar o clasificar la información con la intención de lograr una representación correcta de ella.

Por su parte (Universidad Pedagógica Experimental Libertador, 2006), expone una clasificación donde ayudan al alumno a elaborar y organizar los contenidos para que se produzca más fácil su aprendizaje, es decir, estrategias de adquisición del conocimiento, tales como:

- Estrategias de ensayo, que se integran por las estrategias de codificación como repetir, ensayar, practicar, enumerar. Mnemotécnicas, estrategias de organización. Agrupación, clasificación, categorización,
- Estrategias de Elaboración verbal: Parafrasear, identificar ideas principales, anticipar, predecir, elaborar hipótesis, hacer inferencias, activar el conocimiento, pensar en analogías, extraer conclusiones, hacer notas, responder preguntas, resumir.
- Estrategias de Elaboración imaginaria: Formarse imágenes mentales, imaginaria.
- Estrategias de organización. Elaborar esquemas, mapeo, tormenta de ideas.

También proponen una quinta estrategia, que está destinada a controlar la actividad mental del alumno para dirigir el aprendizaje, y que llaman estrategia de control de la comprensión. Implica permanecer consciente de lo que se está tratando de lograr, seguir la pista de las estrategias que se usan y del éxito logrado con ellas y adaptar la conducta en concordancia. Entre las estrategias de control de la comprensión están: la planificación, la regulación y la evaluación.

- Estrategia de planificación: los alumnos dirigen y controlan su conducta. Son anteriores a que los alumnos realicen ninguna acción. Se llevan a cabo actividades como: establecer el objetivo y la meta de aprendizaje, seleccionar los conocimientos previos que son necesarios para llevarla a cabo, descomponer la tarea en pasos sucesivos, programar un calendario de ejecución, prever el tiempo que se necesita para realizar esa tarea, los recursos que se necesitan, el esfuerzo necesario y

seleccionar la estrategia a seguir.

- Estrategias de regulación, dirección y supervisión: Se utilizan durante la ejecución de la tarea. Indican la capacidad que el alumno tiene para seguir el plan trazado y comprobar su eficacia. Se realizan actividades como: formularles preguntas, seguir el plan trazado, ajustar el tiempo y el esfuerzo requerido por la tarea, modificar y buscar estrategias alternativas en el caso de que las seleccionadas anteriormente no sean eficaces.
- Estrategias de evaluación. Son las encargadas de verificar el proceso de aprendizaje. Se llevan a cabo durante y al final del proceso. Se realizan actividades como: revisar los pasos dados, valorar si se han conseguido o no los objetivos propuestos, evaluar la calidad de los resultados finales y decidir cuándo concluir el proceso emprendido, cuándo hacer pausas, la duración de las pausas, etcétera.

Por último, están las estrategias que apoyan al aprendizaje para que éste se produzca en las mejores condiciones posibles, denominadas como estrategias de apoyo o afectivas. Éstas no se dirigen directamente al aprendizaje de los contenidos, sino que buscan mejorar la eficacia del aprendizaje, mejorando las condiciones en las cuales se produce. Entre ellas, establecer y mantener la motivación, enfocar la atención, mantener la concentración, manejar la ansiedad, manejar el tiempo de manera efectiva, entre otras.

Los docentes deben estructurar y adoptar estrategias de enseñanza con base en las necesidades y objetivos trazados, condiciones de sus cursos, es decir el ambiente que rodea a sus estudiantes y finalmente validar si obtuvo lo que se buscaba.

Algunas estrategias de enseñanza que facilitan el aprendizaje de los estudiantes y que complementan el proceso, según el momento que se esté desarrollando, según (Diaz-Barriga & Rojas, 2002), se encuentran en la tabla 2:

Tabla 2 Estrategias de enseñanza

Estrategia	Habilidades/Proceso cognitivo	Definición	Efectos en el estudiante
Objetivos	Activación de conocimientos previos. Generación de expectativas apropiadas. Orientar y mantener la atención.	Constructo que permite conocer el tipo y aspectos relevantes de la actividad que se desea desarrollar en el proceso de enseñanza-aprendizaje.	Conoce la finalidad y alcance del material y cómo manejarlo. Permite generar expectativas apropiadas respecto de lo que se va a aprender.
Resumen	Promover una organización más adecuada de la información que se aprenderá. Potenciar el enlace entre conocimientos previos y la información que se recibirá..	Síntesis y abstracción de una información relevante de aquello que se ha recibido sea de manera oral o escrita.	Consigna conceptos claves, principios, términos y argumento central de aquello que es objeto de trabajo durante el proceso de enseñanza-aprendizaje.
Organizador previo	Promover una organización más adecuada de la información que se aprenderá.	Información introductoria que se verá posteriormente. Relaciona conocimientos	Enlazarán sus conocimientos anteriores y les facilitará la asimilación del proceso enseñanza-aprendizaje. Hace más accesible el

		previos con los nuevos adquiridos.	contenido. Elabora una visión global y contextual. Optimiza el uso de los dos hemisferios del cerebro.
Ilustraciones	Activación de los conocimientos previos. Generación de expectativas apropiadas. Orientar y mantener la atención.	Representación visual de una teoría o de parte de una sesión de aprendizaje.	Facilita la codificación visual de la información. Dirige y mantiene en el alumno su atención propiciando su interés y motivación Mayor retención en la memoria.
Analogías	Promover una organización más adecuada de la información que se aprenderá. Potenciar enlace de conocimientos previos y la información que se aprenderá. Compara, contrasta e	Proposición que indica que es semejante a otro.	Comprende información abstracta y la traslada a otros contextos. Tener actitud de espíritu analítico para establecer semejanzas y diferencias desde la analogía. Establece relaciones, siendo necesario comparar, contrastar e inferir.

	infiere.		
Preguntas intercaladas	Activación de conocimientos previos. Generación de expectativas apropiadas Orientar y mantener la atención..	Preguntas que constituyen un engranaje armónico y pertinente en el proceso de enseñanza-aprendizaje. Sirven para ayudar a mantener la atención y favorecen la práctica de retención y la obtención de información relevante.	Consolida su aprendizaje. Resuelve dudas.
Pistas tipográficas y discursivas	Activación de los conocimientos previos. Generación de expectativas apropiadas. Orientar y mantener la atención.	Señales particulares o específicas por aprender que se hacen en un texto o en situaciones concretas del proceso enseñanza-aprendizaje.	Mantiene su atención e interés respecto a lo que se está desarrollando en el proceso enseñanza-aprendizaje.
Mapas conceptuales	Promover una organización más	Son partes de los llamados recursos	Les permite jerarquizar adecuadamente los

y redes semánticas	adecuada de la información que se aprenderá. Potenciar enlace de conocimientos previos y la información que se aprenderá.	esquemáticos y sirven para representar gráficamente esquemas de conocimiento.	contenidos, desarrollar la capacidad de análisis y de relación. Realiza una codificación visual y semántica de conceptos, proposiciones y explicaciones.
Uso de estructuras textuales	Promover una organización más adecuada de la información que se aprenderá. Potenciar enlace de conocimientos previos y la información que se aprenderá.	Síntesis teóricas o partes de un discurso oral o escrito que contribuyen a consolidar el aprendizaje o a modo de nemotecnia para recordar lo que se cree pertinente de acuerdo a lo deseado.	Tiene la actitud de evocación y relación con los conocimientos efectuados. Es algo parecido a una ayuda de memoria Facilita el recuerdo y la comprensión de lo más importante de un texto.

Fuente: (Diaz-Barriga & Rojas, 2002)

Existen también estrategias que permiten promover la organización de la información como lo son los mapas cognitivos, y que según (Pimienta, 2012), se clasifican como se muestra en la tabla 3:

Tabla 3 Estrategias para organización de información

Nombre	Descripción
Mapas conceptuales	Representación gráfica de conceptos y sus relaciones. Los conceptos guardan entre sí un orden jerárquico y están unidos por líneas identificadas por palabras (de enlace), que establecen la relación que hay entre ellas.
Mapas mentales	Forma gráfica de expresar los pensamientos en función de los conocimientos que se han almacenado en el cerebro.
Mapas semánticos	Es una estructuración categórica de información, representada gráficamente, que no tiene una jerarquía definida.

Fuente: elaboración propia

Relaciona también otros mapas cognitivos, tales como: tipo sol, de telaraña, de aspectos comunes, de ciclos, de secuencia, de cajas, de calamar y de algoritmo.

Los aspectos conceptuales de las estrategias de enseñanza reseñadas, tienen pertinencia en el presente trabajo, teniendo en cuenta que se desean usar como la forma en la cual se buscará la apropiación de un subconjunto de buenas prácticas de PSP por parte de los estudiantes del curso de programación avanzada.

5.3 PROYECTOS FORMATIVOS

La metodología de Proyectos Formativos se encuentra basada en problemas reales, y fueron concebidos como procesos planeados que orientan la formación de competencias a partir de un nodo problematizador al cual se articulan a través del análisis para la resolución del problema (Tobón, 2005).

Los Proyectos Formativos retoman los elementos del método de proyectos que (Kilpatrick, 1918) conceptualizó y sistematizó, como un procedimiento dinámico de organizar la

enseñanza mediante actividades con verdadero sentido vital para los estudiantes, y según él, tienen las siguientes características:

1. El objetivo central de un proyecto no es la información verbal memorizada, sino la aplicación del raciocinio y la búsqueda de soluciones a las realidades.
2. La información no se aprende y transmite por sí misma, sino que es buscada con el fin de poder actuar y solucionar la situación detectada en la realidad.
3. El aprendizaje se lleva a cabo en el entorno real e involucra la vida de los estudiantes.
4. La enseñanza se fundamenta en problemas, por lo cual éstos están antes que los principios, leyes y teorías.

En la metodología de Proyectos formativos existe un elemento articulador que guía todo el proceso metodológico, este elemento es llamado Ruta Formativa. Ésta permite estructurar y organizar sistémicamente los diversos componentes que hacen parte de la metodología (Tobón, 2005).

La definición de la ruta formativa permitió articular los elementos que la componen: Presentación del proyecto, las competencias a desarrollar, la estructura formal del programa, las actividades del proyecto y la estructura de la evaluación.

(S. Cardona, Vélez, & Tobón, 2014) describe los elementos que se integran a través de la ruta formativa y las fases correspondientes a esta metodología, de la siguiente forma:

Elementos del proyecto formativo:

- Estructura formal: contiene información relacionada con el proyecto formativo: título, créditos académicos, horas de trabajo independiente del estudiante, horas de trabajo dirigido por el profesor del curso.
- Competencias: Descripción de las actuaciones integrales que deben desarrollar los estudiantes a partir de las actividades definidas dentro de los proyectos formativos.
- Actividades del Proyecto: Las actividades deben estar concatenadas entre sí, tener una secuencia de proyecto y considerar los diferentes saberes de la o las

competencias a través de los criterios. Se programan actividades con el docente y de aprendizaje autónomo de los estudiantes.

- Estructuración de la evaluación: De acuerdo a los criterios establecidos, se planifican las matrices de evaluación (rúbricas), que deben reflejar el mapa de aprendizaje a través del cual los estudiantes van a lograr tales criterios.

Fases del proyecto formativo:

- Direccionamiento: Se acuerda la ruta del proyecto formativo con los estudiantes y se establecen los criterios necesarios a tener en cuenta en el proyecto.
- Planeación: Los estudiantes planifican con el docente un proyecto acorde con el propósito general del curso.
- Actuación – ejecución: Los estudiantes ejecutan el proyecto diseñado con la mediación del docente, buscando el logro de las competencias planteadas.
- Socialización: los estudiantes presentan los resultados alcanzados. El profesor realiza la valoración del proyecto formativo de acuerdo a los criterios definidos.

El proyecto formativo consta también de actividades, que son los momentos de trabajo entre profesor y estudiantes. Las actividades son las acciones guiadas o autónomas que llevan al aprendizaje y al logro de competencias en el proceso formativo, con base en criterios y evidencias.

La metodología contempla la valoración, que según (Tobón, 2010), propone resaltar el carácter apreciativo de la evaluación y enfatizar que es ante todo un proceso de reconocimiento de lo que las personas aprenden y ponen en acción-actuación en un contexto social, asumiéndose el error como una oportunidad de mejora y de crecimiento personal.

En la metodología de Proyectos Formativos se propone evaluar en tres momentos diferentes: al inicio del proceso (diagnóstico), durante el proceso (retroalimentación para conocer cómo va el proceso) y al finalizar, y de igual forma, valora de 3 maneras interdependientes: autoevaluación (valoración por parte del mismo estudiante de sus conocimientos, aptitudes y competencias), coevaluación (apreciación de los conocimientos, aptitudes y competencias de un estudiante realizada por otro compañero de clase, se realiza

entre pares o iguales) y la heteroevaluación (valoración de las competencias hechas por un tercero, en este caso por el docente).

6 OBJETIVOS

6.1 OBJETIVO GENERAL

Diseñar, implementar y validar una estrategia de enseñanza para la apropiación de un subconjunto de prácticas de los niveles PSP0 y PSP0.1 en el curso de Programación Avanzada del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM.

6.2 OBJETIVOS ESPECÍFICOS

1. Realizar un diagnóstico sobre el nivel de apropiación y uso de buenas prácticas de programación de los estudiantes del programa de Ingeniería de Software de la EAM.
2. Definir los lineamientos pedagógicos, metodológicos y tecnológicos que permitan la implementación de una estrategia de enseñanza orientada a la apropiación de prácticas de PSP en el curso de Programación Avanzada del programa de Ingeniería de Software de la EAM.
3. Diseñar los instrumentos de evaluación que permitan hacer seguimiento y retroalimentación a los estudiantes, para identificar el nivel de apropiación de prácticas de PSP.
4. Evaluar el impacto de la estrategia de enseñanza en la apropiación de buenas prácticas de programación de los estudiantes, mediante la realización de una prueba piloto en el curso de Programación Avanzada del programa de Ingeniería de Software de la EAM.

7 METODOLOGÍA

El diseño de la investigación fue de carácter cuasi experimental, dado que se estudiaron relaciones causa-efecto, pero no en condiciones de un control riguroso de todos los factores que pueden afectar el experimento (Martínez Rodríguez, 2011), y se soportó en un diseño intergrupos con el grupo control y experimental, con medida pretest – postest. Estuvo enmarcado en un enfoque cuantitativo que posibilitó predecir las variables que fueron objeto de estudio. El diseño permitió validar la estrategia de enseñanza propuesta para incorporar buenas prácticas de desarrollo PSP; es de tipo descriptivo, porque se basó en la observación a los diferentes entregables realizados por los estudiantes y la descripción de su proceso personal de desarrollo de software evidenciado en ellos; y correlacional porque se buscó encontrar cómo la formación en PSP apoyada en una estrategia de enseñanza influye en la apropiación de buenas prácticas de programación por partes de los estudiantes.

Se definió como hipótesis de la investigación, el siguiente enunciado: Los estudiantes del curso de Programación Avanzada I del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM, que incorporan prácticas PSP a través de una estrategia de enseñanza, apropian mayor cantidad de buenas prácticas para su proceso personal de desarrollo de software que los estudiantes del curso de Programación Avanzada II, pertenecientes al mismo programa de estudios e institución, los cuales no participan en la intervención.

En el proceso se realizó la definición formal y conceptual de las variables, al igual que la identificación de los indicadores/valores que permiten hacer la medición de éstas.

Para el desarrollo de esta investigación se definieron dos tipos de variables: independientes y dependientes. Las variables dependientes son las que determinan el objeto de estudio, las variables independientes son aquellas que producen un efecto sobre las variables dependientes.

En la tabla 4, se relacionan los tipos de variables que obedecen al propósito de la investigación, mostrando las variables independientes y dependientes con su correspondiente indicador de medida.

Tabla 4 Operacionalización de variables

Tipos de variable	Operacionalización
Dependiente	
Nivel de apropiación de prácticas de PSP	Ordinal, medido en el nivel de apropiación. Los diferentes niveles de dominio que pueden obtenerse como resultado del proceso de evaluación de cada estudiante son: Receptivo, Resolutivo, Autónomo o Estratégico.
Independiente	
Estrategia de enseñanza	- Con estrategia de enseñanza (**e_1**) - Sin estrategia de enseñanza (**e_2**)

Fuente: elaboración propia

La investigación tuvo su inicio en la aplicación de un instrumento de pretest, que permitió conocer el nivel de uso de las buenas prácticas de PSP en las actividades de desarrollo de software de los estudiantes. Posteriormente, se realizó el diseño de la estrategia de enseñanza que orientó el aprendizaje de los estudiantes y la evaluación de las competencias PSP adquiridas. La estrategia de enseñanza se construyó a partir de una metodología de proyectos formativos para el desarrollo de competencias a través de fases. La estrategia diseñada se puso en práctica con el grupo experimental, el cual, de manera adicional, recibió la retroalimentación respectiva, conforme a sus avances en las actividades, mientras que el grupo control estuvo orientado bajo la estrategia tradicional de enseñanza. El mismo instrumento que sirvió para la obtención de información inicial, se usó para realizar el postest con ambos grupos, permitiendo conocer el impacto que la estrategia de enseñanza tuvo en la formación del grupo experimental y realizando la comparación con los avances que en ese sentido pudo tener el grupo control.

En la tabla 5 se presenta el diseño intergrupos, el cual corresponde a la tipología de Campbell & Stanley (1966).

Tabla 5 Diseño de los grupos para la experimentación

Grupo de Sujetos	**Asignación de sujetos**	**Medida de sujetos (pretest)**	**Intervención**	**Medida de Sujetos (postest)**
G_1 (Experimental)	Aleatoria	O_1	X	O_2
G_2 (Control)	Aleatoria	O_1	--	O_2

Fuente: elaboración propia

La población para la realización del estudio está compuesta por los estudiantes inscritos en los cursos de Programación Avanzada I de Quinto semestre y Programación Avanzada II con estudiantes de Sexto semestre del pregrado del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío.

Parámetros del experimento:

- Institución: Ambos grupos (control y experimental) pertenecen a la misma institución.
- Programa: Ambos grupos (control y experimental) pertenecen al mismo programa de pregrado.
- Ciclo propedéutico: Ambos grupos (control y experimental) pertenecen al ciclo de tecnología en la cual los clasifica la Escuela de Administración y Mercadotecnia del Quindío EAM.

Factores del experimento:

- Estrategia de aprendizaje: Se estructuró a partir de proyectos formativos, que permiten a los estudiantes adquirir unas competencias seleccionadas.

  La estrategia será medida a través de la valoración del conocimiento de los estudiantes y tendrá como resultado un nivel de apropiación de las buenas prácticas.

Los niveles cuentan con una equivalencia numérica basada en las notas obtenidas por los estudiantes en una escala de 0.0 a 5.0, como se muestra en la tabla 6:

Tabla 6 Escala de valoración para la apropiación de las prácticas PSP

NIVEL	**VALOR CUANTITATIVO**
Receptivo	0.0 - 2.9
Resolutivo	3.0 – 3.7
Autónomo	3.8 – 4.4
Estratégico	4.5 – 5.0

Fuente: elaboración propia

Las evidencias se valoran por niveles de dominio (Cardona, 2014):

- Receptivo: El estudiante enfrenta el problema propuesto con algunas nociones y de manera operativa y mecánica.
- Resolutivo: El estudiante afronta el problema propuesto entendiendo procedimientos elementales.
- Autónomo: El estudiante intenta resolver el problema propuesto con autonomía.
- Estratégico: El estudiante obtiene resultados que impactan el desempeño ante el problema propuesto.

Los instrumentos de recolección de la información se incorporaron como se muestra en la tabla 7:

Tabla 7 Instrumentos de recolección de la información

Instrumento	Objetivo	Descripción
Encuesta de uso de buenas prácticas de desarrollo de software (Instrumento de Pretest)	Obtener información de diagnóstico.	Encuesta con preguntas agrupadas (planeación, gestión de tiempo, defectos, y proceso), que permite determinar las diferentes buenas prácticas en desarrollo de software que los estudiantes tienen al momento de iniciar la intervención.
Formatos PSP: • Registro de tiempo • Registro de defectos • Resumen de plan de proyecto • PIP • Reporte de pruebas	Realizar el registro de las métricas y datos del proceso personal de software	Formatos basados en PSP, que permiten registrar las métricas del proceso con el cual se construye software, de tal manera que posibilita el conocimiento de aspectos cualitativos de su proceso personal de desarrollo de software, tanto en ejercicios independientes como en su proyecto de final de la intervención.
Encuesta de uso de buenas prácticas de desarrollo de software (Instrumento de Postest)	Obtener información para validación de resultados.	Encuesta con preguntas agrupadas (planeación, gestión de tiempo, defectos, y proceso), que permite determinar las diferentes buenas prácticas en desarrollo de software que los estudiantes tienen al momento de finalizar la intervención.
Test de nivel de conocimiento de PSP	Obtener información sobre el conocimiento de los estudiantes sobre PSP	Test de conocimiento que se realizó al final del curso.

Fuente: elaboración propia

7.1 DEFINICIÓN DE LA ESTRATEGIA DE ENSEÑANZA

En esta fase se definen los elementos conceptuales y del estado del arte que soportan el desarrollo del trabajo y se identifican los elementos relacionados con conceptos PSP, métricas usadas, guías, formatos, estándares, niveles de cobertura de PSP, herramientas de soporte, herramientas de retroalimentación, ejercicios a realizar y resultados obtenidos a través de herramientas de evaluación como lo son las rúbricas.

La estrategia de enseñanza estuvo basada en la estructuración de actividades articuladas orientadas a identificar, interpretar, argumentar o resolver uno o varios problemas del contexto, que permitieron favorecer la formación integral y el aprendizaje de competencias a los estudiantes del curso, integrando el saber ser con el saber hacer y el saber conocer, a través de la metodología de Proyectos Formativos.

7.1.1 Generalidades del Programa de Ingeniería de Software

Esta investigación se desarrolló en la Escuela de Administración y Mercadotecnia del Quindío EAM, institución educativa fundada en el año 1971, que nació con una vocación Técnica Profesional, adopta en 2005 una reforma estatutaria que la conduce a la redefinición por ciclos propedéuticos y un año más tarde a través de resolución No. 3544 del Ministerio de Educación Nacional, es ratificada como Institución de Educación Superior que la habilita para ofrecer programas de formación universitaria que operan bajo esa modalidad (EAM, 2013).

El programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM, el primero con esta denominación en el país, inicia su historia en el año 2006, a partir de la redefinición institucional de su programa "Técnica en Análisis y Programación de Computadores", obteniendo los registros calificados para ofertar el programa por ciclos propedéuticos, a saber:

- Nivel 1: Técnica Profesional en Desarrollo de Software
- Nivel 2: Tecnología en Desarrollo de Software
- Nivel 3: Ingeniería de Software

Desde el año 2007, el programa "Técnica en Análisis y Programación de Computadores" fue retirado de la oferta académica y con sus estudiantes activos se inició un proceso de

transición que les permitía continuar con su formación, migrando a los niveles Técnico y Tecnológico, según correspondiera. En el año 2008 se tuvo los primeros egresados como Tecnólogos en Desarrollo de Software y en 2009 los primeros Ingenieros de Software, no sólo del programa, sino los primeros del país con dicho título. Hasta 2013, los egresados ascendían a 506 profesionales en sus 3 niveles (EAM, 2013).

Las fichas SNIES de los 3 ciclos del programa, se muestran en las tablas 8, 9 y 10.

Tabla 8 Ficha SNIES Programa Nivel Técnico Profesional

Institución	Escuela de Administración y Mercadotecnia del Quindío –EAM, SNIES 4709
Denominación	TÉCNICA PROFESIONAL EN DESARROLLO DE SOFTWARE POR CICLOS PROPEDÉUTICOS
SNIES	52411
Título a expedir	TÉCNICO PROFESIONAL EN DESARROLLO DE SOFTWARE
Créditos	67
Ubicación	ARMENIA –QUINDÍO
Nivel	FORMACIÓN-TÉCNICA PROFESIONAL
Metodología	PRESENCIAL POR CICLOS PROPEDÉUTICOS
Área de Conocimiento	INGENIERÍA, ARQUITECTURA, URBANISMO Y AFINES
Núcleo básico de Conocimiento	INGENIERÍA DE SISTEMAS, TELEMÁTICA Y AFINES
Tipo de Programa	PRINCIPAL – QUINDÍO – ARMENIA

Fuente: (EAM, 2013)

Tabla 9 Ficha SNIES Programa Nivel Tecnológico

Institución	Escuela de Administración y Mercadotecnia del Quindío –EAM, SNIES 4709
Denominación	TECNOLOGÍA EN DISEÑO DE SOFTWARE POR CICLOS PROPEDEUTICOS
SNIES	102908
Título a expedir	TECNÓLOGO EN DISEÑO DE SOFTWARE
Créditos	108
Ubicación	ARMENIA –QUINDÍO
Nivel	TECNOLOGÍA
Metodología	PRESENCIAL POR CICLOS PROPEDÉUTICOS
Área de Conocimiento	INGENIERÍA, ARQUITECTURA, URBANISMO Y AFINES
Núcleo básico de Conocimiento	INGENIERÍA DE SISTEMAS, TELEMÁTICA Y AFINES
Tipo de Programa	PRINCIPAL – QUINDÍO – ARMENIA

Fuente: (EAM, 2013)

Tabla 10 Ficha SNIES Programa Nivel Universitario

Institución	Escuela de Administración y Mercadotecnia del Quindío –EAM, SNIES 4709
Denominación	INGENIERÍA DE SOFTWARE POR CICLOS PROPEDEUTICOS
SNIES	52410
Título a expedir	INGENIERO DE SOFTWARE
Créditos	161
Ubicación	ARMENIA –QUINDÍO
Nivel	PREGRADO-NIVEL UNIVERSITARIO
Metodología	PRESENCIAL POR CICLOS PROPEDÉUTICOS
Área de Conocimiento	INGENIERÍA, ARQUITECTURA, URBANISMO Y AFINES
Núcleo básico de Conocimiento	INGENIERÍA DE SISTEMAS, TELEMÁTICA Y AFINES
Tipo de Programa	PRINCIPAL – QUINDÍO – ARMENIA

Fuente: (EAM, 2013)

El programa está dirigido a preparar profesionales que satisfagan las necesidades actuales de la industria, teniendo en cuenta los 3 niveles de formación en que está constituido.

El perfil profesional del Técnico en Desarrollo de Software genera competencias para implementar, instalar y mantener aplicaciones de software de baja complejidad, desarrollando en ellos competencias que le permitan la interpretación e implementación de artefactos de diseño de software, a partir de estándares y representaciones de lenguajes de modelado, en lenguajes orientados a objetos, en entorno web, con persistencia en bases de datos. Su perfil lo habilita para desempeñar labores operativas relacionadas con el proceso de desarrollo de software (EAM, 2013).

El perfil del Tecnólogo en Desarrollo de Software lo forma para desempeñarse diseñando y desarrollando aplicaciones de software de acuerdo a especificaciones del proceso de ingeniería, desarrollando competencias como el diseño e implementación de aplicaciones de software de mediana complejidad, uso de UML para modelado de software, realizar diseño detallado de aplicaciones a partir de la interpretación de diseños arquitectónicos y optimización de aplicaciones. Su perfil le permite estar en capacidad de velar por la calidad del desarrollo de productos haciendo uso de sus habilidades y conocimiento técnico (EAM, 2013).

Por su parte, el perfil del Profesional en Ingeniería de Software le permite desarrollar, gestionar, controlar, evaluar, dirigir y llevar a cabo proyectos de software. Sus competencias lo habilitan para diseñar y construir aplicaciones de alta complejidad, realizar diseños arquitectónicos, usar metodologías de desarrollo ágil para la construcción de software, solucionar problemas a través del modelado de sistemas y aplicar principios de calidad de software en sus etapas de construcción (EAM, 2013).

Cada ciclo propedéutico se encuentra conformado por sus correspondientes semestres, como se muestra en la tabla 11:

Tabla 11 Niveles por semestre

Nivel Técnico Profesional (60 a 70 créditos)				**Nivel Tecnológico (95 a 110 créditos)**		**Nivel Universitario (150 a 165 créditos)**		
I	II	III	IV	V	VI	VII	VIII	IX

Fuente: (EAM, 2013)

En la tabla 12 se presenta en el plan de estudios del programa, diferentes cursos relacionados con programación de computadores que pueden servir como espacios académicos para la incorporación de las prácticas PSP:

Tabla 12 Plan de estudios (Espacios académicos Línea Programación)

Nombre espacio académico	**Semestre**	**Ciclo propedéutico**	**Número de créditos**	**Intensidad semanal**
Lógica de Programación	I	Técnico	4	6
Lenguaje de programación	II	Técnico	4	6
Programación Web	III	Técnico	4	4
Estructuras de datos	III	Técnico	4	6
Desarrollo de software	IV	Técnico	4	4

Programación Avanzada I	V	Tecnólogo	4	4
Programación Avanzada II	VI	Tecnólogo	4	4
Ingeniería de software I	VI	Tecnólogo	3	4
Ingeniería de software II	VII	Universitario	3	4
Ingeniería de software III	VIII	Universitario	3	4
Desarrollo en equipo	VIII	Universitario	3	4
Arquitecturas de software	IX	Universitario	3	4

Fuente: (EAM, 2013)

Teniendo en cuenta los espacios académicos relacionados en la tabla 11, se evidencia una importante formación en aspectos técnicos de la programación en cada uno de los ciclos propedéuticos.

7.2 DISEÑO DE LA ESTRATEGIA DE ENSEÑANZA

Con el fin de articular adecuadamente los elementos necesarios que permitan la apropiación de las prácticas de desarrollo PSP, se realizó el diseño de cada uno de los componentes académicos que facilitaron la intervención sobre el grupo experimental. El resultado es la estrategia de enseñanza, que incluye los instrumentos requeridos para aplicar en el curso.

A continuación se describe los aspectos generales de los cursos de Programación Avanzada I y II, los cuales hacen parte del contexto en el cual se desarrolló la estrategia:

7.2.1 Descripción del Grupo Experimental: Programación Avanzada I

La tabla 13 relaciona la información del curso registrada en el documento Carta Descriptiva del programa Tecnología Profesional en Desarrollo de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM:

Tabla 13 Identificación de la asignatura Programación Avanzada I

NOMBRE	Programación Avanzada I (C)
CÓDIGO	23414
SEMESTRE	V
ÁREA	Ingeniería Aplicada
NÚMERO DE CRÉDITOS	4
INTENSIDAD SEMANAL	4
HABILITABLE	No
HOMOLOGABLE	Si
PRERREQUISITOS	23413 (Diseño de Software)

Fuente: (EAM, 2013)

El objetivo general de este curso de programación es aplicar patrones de diseño para crear aplicaciones robustas que usen concurrencia, distribución geográfica y persistencia avanzada de objetos. Tomando como base las unidades temáticas propias del curso, se propuso incorporar las temáticas PSP relacionadas en la tabla 14.

Tabla 14 Integración de prácticas PSP en las unidades temáticas del curso

Unidad	**Temática Programación Avanzada: Objetivos**	**Temática Programación Avanzada: Contenido**	**Temática PSP**	**Tiempo estimado**
Unidad 1: CONCURRENCIA	• Aplicar hilos para crear aplicaciones concurrentes. • Usar el patrón Observador-Observable para la comunicación de la vista con la lógica de negocio.	• Concepto de concurrencia. • Clase Thread e interface Runnable. • Excepciones en la concurrencia. • Sincronización y bloqueos de procesos. • Señalización entre hilos.	• Introducción a la calidad de software • Proceso de desarrollo de software • Métricas en el software	14 horas
Unidad 2: MODELO VISTA CONTROLADOR USANDO EL PATRÓN OBSERVADOR	• Reconocer las ventajas de usar un patrón de diseño para resolver un problema. • Usar el patrón observador para desacoplar el modelo de la interfaz gráfica de usuario.	• Patrones de diseño de software. • Patrón observador. Conceptos y motivación. • *Observer* y *Observable*.	• Registro de tiempos (plantilla) • Gestión del tiempo (interrupciones) • Registro de defectos (plantilla y estándar) Estándar tipos de defectos	8 horas

Unidad 3: **PROGRAMACIÓN DISTRIBUIDA CON *SOCKETS***	• Crear aplicaciones distribuidas a través de *socket*. • Usar flujos de E/S para enviar y recibir datos por la red. • Crear protocolos de mensajes para la comunicación de aplicaciones distribuidas. • Usar la concurrencia en aplicaciones distribuidas.	• Conceptos de comunicación TCP/IP • *Socket* y *ServerSocket*. • Flujos de E/S para *String* y objetos. • Protocolos de aplicación. • Concurrencia aplicada a aplicaciones distribuidas.	• Guiones PSP0 • Planeación en el proceso de desarrollo de software • Estándar de codificación	12 horas
Unidad 4: **PERSISTENCIA CON JPA**	• Mapear una base de datos relacional a objetos. • Usar las anotaciones de JPA para crear entidades que representen una base de datos. • Crear una unidad de persistencia para configurar la conexión a la base de datos. • Crear consultas usando el JPQL.	• Mapeo objeto-relacional • Entidades y anotaciones. • Unidad de persistencia y EntityManager. • JPQL.	• Pruebas unitarias - Registro de sus resultados • Fase de Post-Mortem – Formato de resumen de plan de proyecto • Propuesta de mejora de proceso	14 horas

Fuente: (EAM, 2013)

7.2.2 Descripción del Grupo Control: Programación Avanzada II

El curso en el cual se encontraba inscrito del grupo control también correspondía al ciclo propedéutico Tecnológico, y se realizó en el Semestre VI del programa. Fue dirigido mediante una estrategia de aprendizaje tradicional impartida por un docente de la EAM, quien lo ha dirigido durante varios semestres.

A continuación, se relaciona la información del curso registrada en el documento Carta Descriptiva del programa Tecnología Profesional en Desarrollo de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM:

Tabla 15 Identificación de la asignatura Programación Avanzada II

NOMBRE	Programación Avanzada II ©
CÓDIGO	23415
SEMESTRE	VI
ÁREA	Ingeniería Aplicada
NÚMERO DE CRÉDITOS	4
INTENSIDAD SEMANAL	4
HABILITABLE	NO
HOMOLOGABLE	SI
PRERREQUISITOS	23414 (Programación Avanzada I)

Fuente: (EAM, 2013)

El objetivo general del curso es usar tecnologías que permitan crear aplicaciones empresariales basadas en WEB robustas, seguras y escalables de manera fácil y ágil usando herramientas de programación para tal fin. En la tabla 16, se muestran las unidades temáticas de la materia.

Tabla 16 Unidades temáticas – Programación Avanzada II

No. Unidad	**Nombre Unidad**	**Descripción**
1	**CONCEPTOS BASICOS**	Se trabaja la estructura general de las aplicaciones empresariales de Java (JEE), los servidores de aplicaciones y la configuracion del IDE para desarrollo.
2	**JAVA SERVER FACES**	Esta unidad comprende la revisión de la configuración de proyectos con JSF, etiquetas básicas, convertidores, validadores y la configuración en la navegación entre páginas.
3	**AJAX EN JSF**	Configuración de proyectos usando AJAX, solicitudes al servidor y el uso de etiquetas <a4j>
4	**FRAMEWORKS DE PRESENTACIÓN - RICHFACES**	Configuración y uso de etiquetas del framework Richfaces.
5	**LÓGICA DE NEGOCIO EN ENTERPRISE JAVA BEAN**	En esta unidad se expone el trabajo con proyectos empresariales y la importancia de los EJB en la lógica de negocio de una aplicación.
6	**ACCESO AL EJB DESDE OTROS CONTEXTOS**	Se revisan protocolos y tecnologías para conectarse a EJB y uso del serviceLocator.

Fuente: (EAM, 2013)

Para el grupo control, las temáticas correspondientes al curso se impartieron basadas en la bibliografía teórica recomendada por la EAM. Se realizaron actividades tanto guiadas por el docente en el salón de clases, como trabajo autónomo por parte de los estudiantes, de manera individual en unos casos y organizados en equipos para otros.

7.2.3 Actividades De Aprendizaje Del Grupo Experimental

Basados en las unidades temáticas y el contenido del curso de Programación Avanzada I, se realizó el diseño de diversas actividades académicas encaminadas a formar a los estudiantes en un subconjunto de prácticas PSP, haciendo uso de los elementos ilustrados en la tabla 17 que se encuentra a continuación:

Tabla 17 Actividades de aprendizaje

Unidad	Actividades	Productos esperados	Evaluación	Recursos
Unidad 1: Concurrencia	• Exposición por parte del docente acerca de calidad en el software y métricas en el software. • Debates en torno al proceso personal de desarrollo de software PSP. • Presentación de proyectos de software sencillos por parte de los estudiantes. • Lectura de artículos acerca de la calidad en el software por parte de los estudiantes.	• Cuestionario diligenciado Taller resuelto	• Realización de un cuestionario acerca de la calidad en el software y de un taller de métricas por parte de los estudiantes • Generación de un taller con métricas en el software.	• Artículos relacionados con la calidad en el software. • Proyectos de desarrollo de software para resolver problemas de diferente índole. • Formatos PSP sugeridos para obtener las métricas presentadas. • Un cuestionario acerca de la calidad en el software. • Un taller de métricas de software.
Unidad 2: Modelo Vista Controlador usando el patrón Observador	• Exposición por parte del docente acerca de etapas básicas en la construcción de software e incorporación de 2 métricas PSP. • Se brindan instrucciones de diligenciamiento de los formatos PSP de	Aplicación con desarrollo de hilos y en arquitectura MVC.	Taller con proyecto de hilos en java y arquitectura MVC.	• Formatos PSP • Taller con proyecto a desarrollar

	acuerdo con las métricas explicadas.			
Unidad 3: Programación distribuida con *Sockets*	• Exposición de guías PSP. • Simulación de la construcción del software siguiendo la guía de PSP0. • Entrega y explicación de estándar de codificación.	• Aplicación con proceso PSP0 utilizando *sockets*.	• Taller con proyecto realizando el proceso de PSP0	• Guías PSP • Estándar de Codificación PSP
Unidad 4: Persistencia con JPA	• Realización de un desarrollo de software con la entrega de requerimientos técnicos y de calidad. • Exposición por parte del profesor de la fase de post-mortem. • Se socializó el formulación de Resumen de Plan de Proyecto y formato PIP.	• Aplicación con *JPA*. • Registros PSP en los formatos correspondientes.	• Taller con proyecto con JPA y los registros explicados durante toda la intervención.	• Guías PSP • Taller con proyecto a desarrollar

Fuente: elaboración propia

La ejecución en el curso se realizó a través del docente asignado por la institución para el curso de Programación Avanzada I. Antes de cada encuentro se sostuvo una reunión de hora y media con el profesor, con el fin de presentar los instrumentos que se proponían para cada sesión y resolver cualquier duda que se pudiera tener.

Se realizaron sesiones de trabajo en las clases correspondientes al curso de Programación Avanzada I. En cada una de ellas se explicaron conceptos relacionados con la calidad en el software, procesos de desarrollo de software y las prácticas propuestas por PSP. Por cada sesión se diseñaron instrumentos para apoyar la intervención, tal como se muestra en la tabla 18: una guía de clase, material teórico, instrumentos de recolección de información del proceso y una rúbrica de validación y retroalimentación.

Tabla 18 Instrumentos de apoyo a la intervención

Instrumento	Propósito	Componentes	Dirigido a
Guía instruccional del curso	Dirigir al profesor en el desarrollo de la clase, lo contextualiza en el tema PSP, establece las actividades a realizar y relaciona los recursos de aprendizaje de la clase correspondiente.	• Objetivos PSP • Metodología • Actividades guiadas por el profesor y autónomas del estudiante • Material entregado al profesor	Docente
Instrumento de registro de información	Permitir el registro de los datos propios de PSP, basado en la práctica vista en clase	• Estructura del formato PSP oficial	Estudiantes
Elemento de valoración de competencia (Rúbrica)	Dar a conocer al profesor y estudiantes hacia los criterios que se evalúan con cada actividad que se realiza.	• Rúbrica	Docente y estudiantes

Fuente: elaboración propia

Estos instrumentos se fueron articulando en cada sesión. La guía instruccional le permitió al docente tener la información de las diferentes actividades, herramientas e instrumentos de que disponía para buscar que los estudiantes pudieran adquirir los conocimientos teóricos y entenderlos en el contexto de los problemas reales. Los instrumentos de registro de información tuvieron como base los formatos propuestos por PSP para las diferentes

prácticas. Su diligenciamiento fue explicado a los estudiantes como parte del proceso, destacando la importancia de realizarlo y de interpreta los datos registrados con el fin de mejorar su proceso personal de desarrollo de software. Las rúbricas permitieron hacer las valoraciones del conocimiento y aptitudes de los estudiantes, posterior a la intervención realizada.

La metodología de Proyectos formativos contempla 4 fases, las cuales se llevaron a cabo con el grupo experimental:

- Direccionamiento: Se realizó una breve introducción a los estudiantes acerca de la intervención y se establecieron los elementos de la ruta formativa y los parámetros a seguir durante el tiempo que durarían las sesiones.
- Planeación: Se establecieron dos proyectos: Hilos-MVC y *Sockets*-JPA, que se tenían planeados como parte de la temática del curso, para que fueran los proyectos formativos sobre los cuales se ejecutarían las prácticas PSP vistas en las sesiones. El primero con asesoría del profesor y el segundo con trabajo independiente del estudiante.
- Actuación: Se entregaron las especificaciones de los proyectos de Hilos-MVC y *Sockets*-JPA para que los estudiantes los desarrollaran ejecutando las prácticas PSP vistas en las sesiones de la intervención.
- Socialización: Los estudiantes presentaron los proyectos funcionando y las evidencias de la ejecución de las prácticas PSP solicitadas.

En la parte final de la intervención, los estudiantes realizaron dos proyectos relacionados con las temáticas del curso. Estos proyectos permitieron la incorporación paulatina de las diferentes buenas prácticas vistas en clase. En el primero de ellos, se solicitó a los estudiantes construir un videojuego donde un personaje principal debía evitar obstáculos en su recorrido de un punto inicial hasta la meta. La construcción debió realizarse tomando como base los conceptos técnicos recibidos por los estudiantes acerca de concurrencia y la arquitectura MVC. El segundo proyecto pedía construir una aplicación que permitiera intercambiar mensajes a través de *sockets* y utilizando el framework de persistencia JPA. De manera adicional, se generon los elementos de evaluación a partir de las entregas de los

requerimientos funcionales (aplicación funcionando) y los requerimientos no funcionales (formatos de buenas prácticas debidamente diligenciados).

7.3 VALIDACIÓN DE LA ESTRATEGIA DE ENSEÑANZA

Con el fin de validar la apropiación de prácticas de desarrollo a través de la estrategia de enseñanza en el curso de Programación Avanzada del programa de Ingeniería de Software de la EAM, se generaron diversos espacios que permitieron determinar el impacto de la estrategia en el proceso de desarrollo por parte de los estudiantes.

Se llevó a cabo un proceso de evaluación del curso, que se realizó al iniciar, durante y al finalizar la intervención.

- Al iniciar la intervención: se realizó un taller donde se verificaron los conceptos de los estudiantes con respecto a la calidad en el desarrollo de software. Para ello se realizó una mesa redonda donde los estudiantes expusieron sus conceptos de manera verbal.
  Posteriormente se realizó la lectura grupal de un ensayo relacionado con la calidad en el software y por último un taller que los estudiantes debían resolver en clase.

- Durante la intervención: se realizaron diversos tipos de actividades relacionadas con las clases desarrolladas y en particular la generación de un proyecto con el cual se validó de manera integral varias prácticas PSP vistas previamente. Cada actividad tenía asociada una rúbrica para evaluar el nivel de dominio que cada estudiante obtenía.

  Para determinar esos niveles de dominio, se realizó la evaluación desde tres procesos interdependientes:

  - Autoevaluación: A cada estudiante le fueron devueltos sus entregables y la rúbrica que corresponde a cada uno. Ellos valoraron su nivel de dominio revisando cada elemento entregado.

- Coevaluación: De manera aleatoria, se entregó a cada estudiante el paquete de entregables de uno de sus compañeros y la rúbrica correspondiente a cada uno, para que la valorara según sus conocimientos.
- Evaluación: El investigador realizó la evaluación del paquete de entregables de cada estudiante y lo registró en la rúbrica correspondiente.

- Al finalizar la intervención: Se preguntó tanto a los estudiantes del grupo control como los del grupo experimental, por el uso de las prácticas que se ilustraron durante la ejecución de la investigación. Los resultados fueron analizados comparando ambos grupos.

  De forma complementaria, se realizó un examen de conocimientos (ver anexo 7 del presente documento), qué permitió determinar la apropiación de los conceptos vistos, mediante los mismos 5 categorías en los cuales se agruparon las prácticas de PSP de esta investigación. Se generaron 4 preguntas de proceso, 4 preguntas de tiempo, 3 de preguntas de tamaño, 3 preguntas de defectos y 2 preguntas de planeación. La prueba se aplicó en ambos grupos y sus resultados también fueron analizados para comparar los conocimientos existentes posterior a la intervención.

  Como parte de la retroalimentación de los estudiantes hacia el proceso de evaluación realizado, se presentó un instrumento donde pudieron registrar la percepción que tuvieron acerca de los tres procesos de evaluación desarrollados. El instrumento constó de 6 preguntas para responder marcando un valor entre 1 y 5 dependiendo de su grado de conformidad con respecto a las preguntas realizadas, donde se marcaba 1 si estaban completamente en desacuerdo, 2 sí se estaba en desacuerdo, 3 ni de acuerdo ni en desacuerdo, 4 cuando se estaba de acuerdo y 5 sí se estaba en total acuerdo con lo planteado en cada cuestionamiento.

8 RESULTADOS

8.1 DIAGNÓSTICO SOBRE PRÁCTICAS PSP DE LOS ESTUDIANTES DE ISW

8.1.1 Instrumento De Recolección Para Diagnóstico De Empleo De Prácticas De Desarrollo

Con el fin de realizar un diagnóstico acerca de las prácticas de desarrollo de software ejecutadas por los estudiantes del programa de Ingeniería de Software de la EAM durante sus trabajos académicos de programación, se diseñó un instrumento para aplicarles al inicio del semestre académico y determinar la ejecución de prácticas antes de la intervención.

A través del instrumento se obtuvo información que permitió determinar el nivel de uso de diferentes prácticas correspondientes a las categorías tiempo, defectos, planeación, tamaño y proceso; que los estudiantes del programa realizaban en sus desarrollos de software.

El instrumento contó con 31 preguntas en total, agrupadas por categorías, como lo muestra la tabla 19. Se realizó un proceso de validación con 6 ingenieros de sistemas, uno de ellos certificado PSP Developer, todos desarrolladores de software con más de 2 años de experiencia, que se desempeñan en una fábrica de software de la ciudad de Armenia. Las observaciones obtenidas de la validación fueron tomadas en cuenta para realizar mejoras en el instrumento.

En el instrumento se solicitó a los estudiantes que respondieran preguntas relacionadas con sus prácticas PSP. Para ello se estableció una escala de Likert, de uno a cinco, en donde: (1) Nunca lo realiza. (2) En muy pocas ocasiones. (3) A veces. (4) Casi siempre. (5) Siempre.

Tabla 19 Instrumento de recolección de información diagnóstica

Categoría	Cantidad de preguntas
Proceso	7
Tiempo	7
Tamaño	5
Defectos	8
Planeación	4

Fuente: elaboración propia

El detalle del instrumento se encuentra en el anexo 3 de este documento.

8.1.2 Población estudiantil del diagnóstico

El instrumento de recolección de información cuyos datos obtenidos apoyan el diagnóstico, fue aplicado al inicio del semestre académico y se realizó con la totalidad de los estudiantes del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío EAM. El instrumento no fue aplicado a los estudiantes de primer semestre dado que apenas inician su formación en el desarrollo de software.

La cantidad total de personas que participaron del diligenciamiento del instrumento fueron 85, quienes representaron la muestra para el diagnóstico. En la tabla 20 se puede visualizar la información asociada con la distribución por semestres a los cuales se les aplicó el instrumento:

Tabla 20 Distribución por semestres en la aplicación de instrumento de recolección de información de diagnóstico

Ciclo	Semestre	Cantidad de estudiantes	Porcentaje
Técnico	II	20	23,53%
Técnico	III	12	14,12%
Técnico	IV	11	12,95%
Tecnólogo	V	12	14,11%
Tecnólogo	VI	9	10,59%
Universitario	VII	10	11,76%
Universitario	VIII	3	3,53%
Universitario	IX	8	9,41%

Fuente: elaboración propia

8.1.3 Selección de grupos control y experimental

La investigación de tipo cuasi-experimental, se realizó con un diseño intergrupos con grupo control y experimental. Para determinar el grupo control y experimental, se consideraron aspectos como:

- Coincidencia de temáticas, espacios académicos, actividades y momentos de interacción.
- Con base en las experiencias previas relacionadas en este informe en el apartado 3.2, se determinó que los cursos deberían ser de semestres del ciclo tecnológico, cuyos estudiantes ya tuvieran competencias de lógica y programación, y pudiesen concentrarse en apropiar adecuadamente las buenas prácticas PSP. También se tuvo en cuenta que los estudiantes de estos semestres pueden tener la motivación

adicional de poner en práctica sus nuevas competencias en los cursos de programación de los semestres futuros.

- Se tuvo en cuenta que ambos cursos deberían pertenecer a semestres que estuviesen en el mismo ciclo propedéutico.

Al analizar los cursos del programa, fue seleccionado como ciclo propedéutico el nivel Tecnológico y los cursos de Programación Avanzada I (Semestre V) y Programación Avanzada II (Semestre VI) como los grupos experimental y control, respectivamente.

La tabla 21 muestra información de género y edad de los estudiantes del curso de Programación Avanzada I:

Tabla 21 Características de los estudiantes del grupo experimental

Total estudiantes	**Proporción Género Masculino**	**Proporción Género Femenino**	**Edad promedio del curso**
12	0,917	0,083	21,16 años

Fuente: elaboración propia

La tabla 22 muestra información de género y edad de los estudiantes del curso de Programación Avanzada I:

Tabla 22 Características de los estudiantes del grupo experimental

Total estudiantes	**Proporción Género Masculino**	**Proporción Género Femenino**	**Edad promedio del curso**
9	1,0	0	22,5 años

Fuente: elaboración propia

8.1.4 Diagnóstico por ciclos propedéuticos

En el caso del diagnóstico por ciclos propedéuticos, se analizaron 3 grupos de información representados por cada uno de los niveles: Técnico, Tecnológico y Universitario.

Para la selección de la técnica estadística que permitiera realizar el análisis de los datos, se descartaron las pruebas de tipo paramétrico, dado que las variables que se manejaron en el diagnóstico son de tipo ordinal, por lo cual no es posible calcularles una media, pero si la moda y la mediana. Por esto, se optó por elegir una técnica no paramétrica, como la prueba de Kruskal-Wallis, que permite comparar medianas.

La prueba de Kruskal-Wallis, es una técnica estadística No Paramétrica que permite comparar las medianas de dos o más poblaciones, permitiendo determinar la homogeneidad o no de los datos (Wackerly, Mendenhall, & Scheaffer, 2010).

El análisis Kruskal-Wallis establece una prueba de hipótesis para cada pregunta analizada, comparando las medianas por grupos de información. La prueba maneja dos hipótesis, la primera (Ho) establece que no hay diferencia entre las medianas de las muestras de información analizadas. Por otro lado, la segunda hipótesis (Ha) establece que si existe diferencia significativa entre las medianas de los grupos analizados.

La generación del p-valor a partir del análisis, determina la posibilidad de rechazar o no rechazar la hipótesis nula. Un p-valor por debajo de 0,05 indica que sí hay diferencia entre las medianas de los datos analizados, mientras que un p-valor superior a 0,05, indica que no se puede rechazar la igualdad de las medianas de los grupos analizados.

Tabla 23 Análisis estadístico Kruskal-Wallis

Hipótesis planteada	p-valor	Decisión
H_a. Existe diferencia entre las medianas de los grupos comparados H_0. No hay diferencia entre las medianas de	$P < 0.05$	Se rechaza la H_0
	$P > 0.05$	No se puede rechazar H_0

los grupos comparados		

Fuente: elaboración propia

Se describe a continuación el análisis estadístico de los datos, mostrando el valor para el estadístico de prueba, que es la función de las mediciones muestrales en las que la decisión estadística estará basada y el P-valor, que es el nivel de significancia alcanzado (Wackerly, Mendenhall, & Scheaffer, 2010). Los datos se muestran agrupados por las categorías PSP propuestas para la presente investigación: Proceso, Tiempo, Tamaño, Defectos y Planeación, respectivamente.

- **Categoría Proceso:**

Las preguntas asociadas a la categoría de Proceso, fueron analizadas y los resultados muestran que en las preguntas 1, 2, 3, 4, 6 y 7, se presentan diferencias estadísticamente significativas entre las medianas de los niveles técnico, tecnológico y universitario. Por el contrario, la pregunta 5 muestra que no hay diferencias significativas en las medianas obtenidas.

La tabla 24 muestra el análisis para las medianas de las respuestas de la categoría Proceso para los 3 ciclos propedéuticos.

Tabla 24 Resultados – Categoría Proceso PSP – Ciclos propedéuticos Técnico, Tecnólogo y Universitario

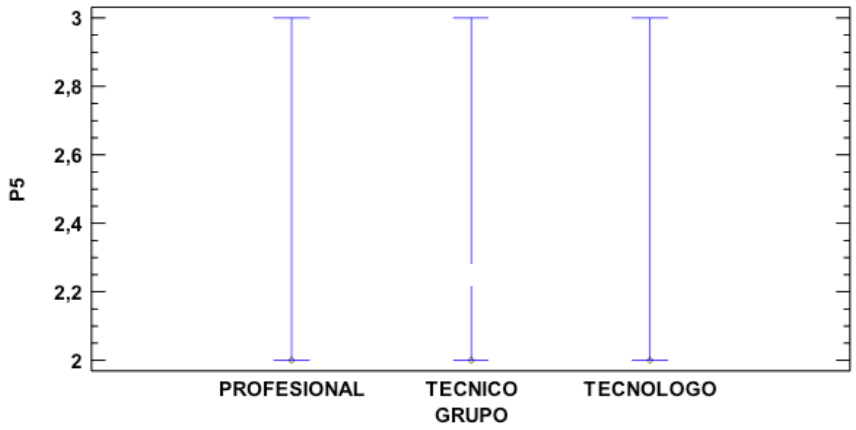
No. Pregunta	**Estadístico de prueba**	**P-valor**
1	38,6291	4,09075E-9
2	25,4615	0,00000295874
3	17,6238	0,000148949
4	37,2332	8,22061E-9
5	1,9522	**0,376778**

6	39,6044	2,51193E-9
7	33,9418	4,26217E-8

Fuente: elaboración propia

Teniendo en cuenta el P-valor asociado a la pregunta 5: ¿Realiza un diseño (diagramas y modelos) de lo que va a construir antes de iniciar su codificación?, se encontró que no existe diferencia estadísticamente significativa entre las medianas relacionadas con esa práctica, por tanto, los tres grupos son homogéneos como lo muestra la figura 3:

Figura 3 Gráfico comparativo - Pregunta 5 - Instrumento de recolección de información – Ciclos propedéuticos Técnico, Tecnólogo y Universitario



Fuente: elaboración propia

Exceptuando los resultados de la pregunta 5 y tomando en cuenta los resultados de las otras 6 preguntas se pudo deducir que el ciclo profesional apropió un mayor número de prácticas relacionadas con el proceso que el ciclo tecnológico. De igual manera se puede decir que más prácticas fueron apropiadas por el ciclo tecnológico comparándolo con los estudiantes del ciclo técnico.

- **Categoría Tiempo:**

El análisis realizado a las preguntas de la categoría de Tiempo, muestra diferencias significativas en todas las preguntas, entre las medianas de los 3 ciclos propedéuticos: técnicos, tecnólogos y universitarios, demostrando heterogeneidad entre ellos.

La tabla 25 muestra el análisis para las medianas de las respuestas de la categoría Tiempo, para los 3 ciclos propedéuticos.

Tabla 25 Análisis de homogeneidad Categoría Tiempo – Ciclos propedéuticos Técnico, Tecnólogo y Universitario

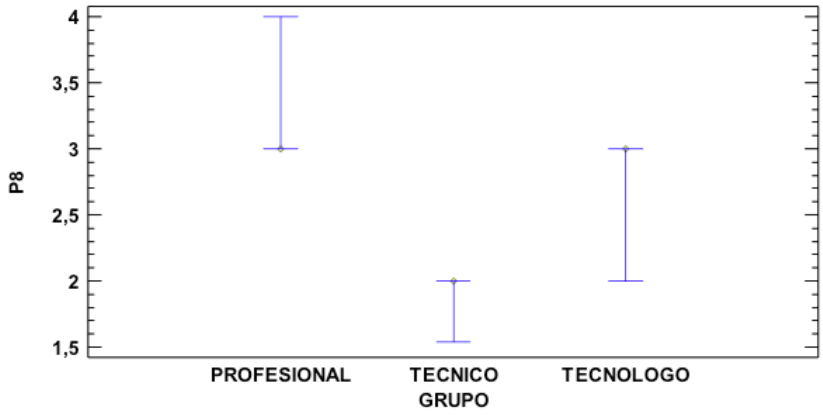
No. Pregunta	**Estadístico de prueba**	**P-valor**
8	30,6628	2,19616E-7
9	27,8801	8,8291E-7
10	45,4004	1,38493E-10
11	43,3397	3,88058E-10
12	57,3822	0
13	46,4127	8,34857E-11
14	57,4142	0

Fuente: elaboración propia

A continuación, se establece como ejemplo una de las preguntas de la categoría, donde se muestra la heterogeneidad presentada.

Se puede visualizar en la figura 4 que corresponde a la pregunta 8: ¿Registra el tiempo que destina para cada actividad que realiza al programar?, que existen diferencias significativas entre las medianas de las respuestas asociadas a la práctica correspondiente, entre los 3 ciclos propedéuticos.

Figura 4 Gráfico comparativo- Pregunta 8 - Instrumento de recolección de información – Ciclos propedéuticos Técnico, Tecnólogo y Universitario



Fuente: elaboración propia

Teniendo como base la figura correspondiente a la pregunta 8, se infiere que el ciclo profesional tiene la percepción de haber apropiado más prácticas relacionadas con el Tiempo que los estudiantes del ciclo tecnológico, y éste a su vez tiene una percepción que apropió más prácticas de Tiempo que el ciclo técnico.

- **Categoría Tamaño:**

Tomando como base las medianas correspondientes a los ciclos propedéuticos para todas las preguntas analizadas con la prueba de Kruskal-Wallis de la categoría de Tamaño, se puede visualizar que se presentan diferencias significativas entre ellas.

La tabla 26 muestra el análisis hecho a las medianas de las respuestas de la categoría Tamaño para los 3 ciclos propedéuticos.

Tabla 26 Análisis de homogeneidad Categoría Tamaño– Ciclos propedéuticos Técnico, Tecnólogo y Universitario

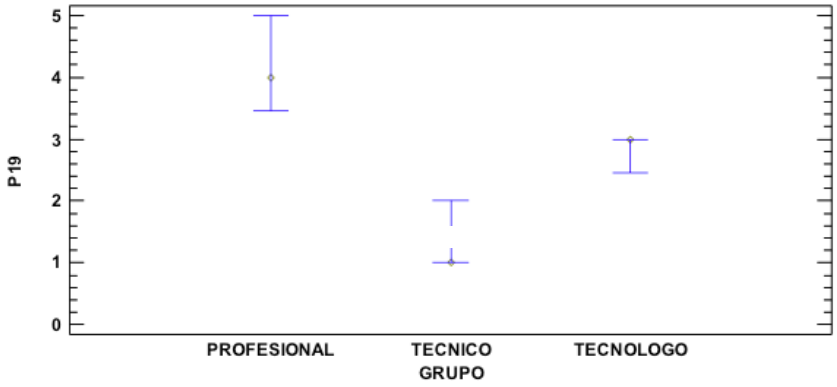
No. Pregunta	Estadístico de prueba	P-valor
15	55,5156	0
16	55,5858	0

17	47,8343	4,10122E-11
18	53,118	2,92122E-12
19	58,6105	0

Fuente: elaboración propia

Con base en el diagrama de medianas correspondiente a la pregunta 19: ¿Cuenta las líneas de código por hora que genera cuando está programando?, es posible afirmar que los estudiantes del ciclo técnico realizan menos el conteo de líneas de código los estudiantes del ciclo tecnológico y del ciclo profesional. Así se visualiza en la figura 5.

Figura 5 Gráfico comparativo - Pregunta 19 - Instrumento de recolección de información – Ciclos propedéuticos Técnico, Tecnólogo y Universitario



Fuente: elaboración propia

Teniendo como base los resultados obtenidos, es posible suponer que los estudiantes de ciclo técnico son quienes menos utilizan las prácticas relacionadas con el tamaño al momento de programar, en comparación con los estudiantes del ciclo tecnológico y éstos a su vez, apropiaron menos prácticas que los estudiantes del ciclo profesional.

- **Categoría Defectos:**

Las preguntas correspondientes a esta categoría muestran en su análisis que hay diferencias entre las medianas para los ciclos profesional, tecnólogo y técnico, tal como se ha analizado en otras categorías, es decir son heterogéneos.

La tabla 27 muestra el análisis para las medianas de las respuestas de la categoría Defectos para los 3 ciclos propedéuticos.

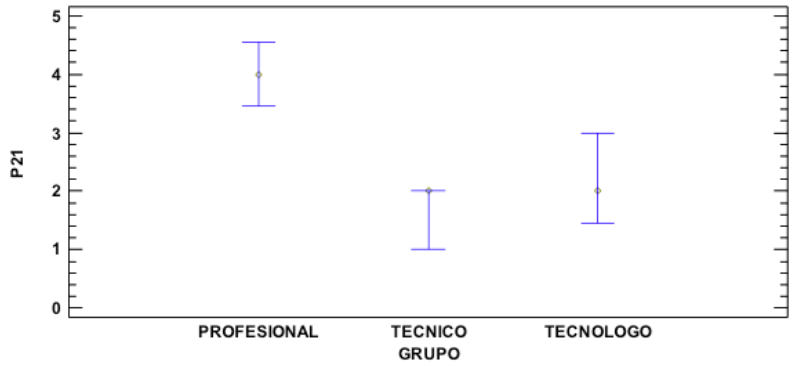
Tabla 27 Análisis de homogeneidad Categoría Defectos – Ciclos propedéuticos Técnico, Tecnólogo y Universitario

No. Pregunta	**Estadístico de prueba**	**P-valor**
20	49,1711	2,10201E-11
21	51,0447	8,23719E-12
22	44,3286	2,36678E-10
23	48,2628	3,31025E-11
24	49,1129	2,16402E-11
25	49,1834	2,0891E-11
26	48,8515	2,46626E-11
27	57,0962	0

Fuente: elaboración propia

El análisis estadístico realizado para la pregunta 21: ¿Establece algún tipo de clasificación cuando comete errores al construir software?, muestra que existe diferencia significativa entre las medianas de cada uno de los ciclos propedéuticos. Los resultados se pueden visualizar en la figura 6.

Figura 6 Gráfico comparativo - Pregunta 21 - Instrumento de recolección de información – Ciclos propedéuticos Técnico, Tecnólogo y Universitario



Fuente: elaboración propia

En conclusión, las prácticas relacionadas con defectos son más ejecutadas por el ciclo profesional en comparación con el ciclo tecnológico, y éstos las ejecutan más que los del ciclo técnico.

- **Categoría Planeación:**

Para todas las preguntas asociadas a la categoría de Planeación, se encuentran diferencias significativas entre las medianas de los ciclos analizados.

La tabla 28 muestra el análisis para las medianas de las respuestas de la categoría Planeación para los 3 ciclos propedéuticos.

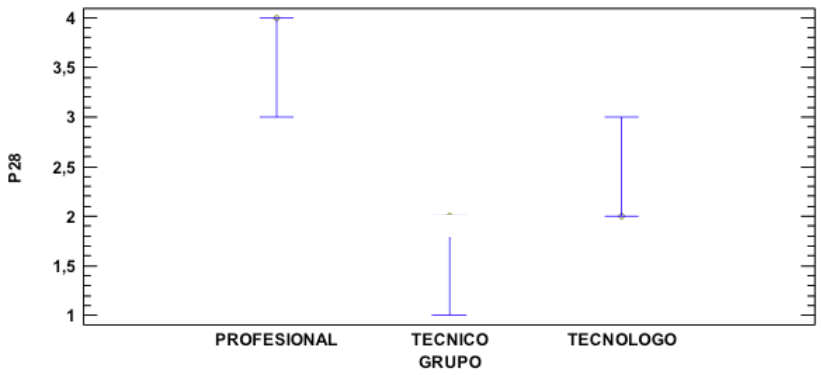
Tabla 28 Análisis de homogeneidad Categoría Planeación – Ciclos propedéuticos Técnico, Tecnólogo y Universitario

No. Pregunta	Estadístico de prueba	P-valor
28	55,4503	0
29	49,4701	1,8101E-11
30	36,9438	9,50051E-9
31	47,9139	3,94127E-11

Fuente: elaboración propia

El siguiente análisis está relacionado con la pregunta 28: ¿Establece las actividades (levantamiento de requerimientos, análisis, diseño, codificación, pruebas u otros) que va a realizar cuando programa? Sus resultados muestran diferencias significativas entre las medianas de los ciclos propedéuticos analizados. La figura 7 da muestra de dicho análisis con evidente heterogeneidad de los resultados.

Figura 7 Gráfico comparativo – Pregunta 28 - Instrumento de recolección de información – Ciclos propedéuticos Técnico, Tecnólogo y Universitario



Fuente: elaboración propia

eniendo en cuenta la gráfica de medianas se puede inferir que las prácticas de planeación son más realizadas por los estudiantes del ciclo profesional en comparación con los estudiantes del ciclo tecnológico y a su vez, los estudiantes del ciclo tecnológico realizan más las prácticas de planeación que los estudiantes del ciclo técnico.

El conjunto completo de las gráficas de homogeneidad entre los grupos, se encuentran en el Anexo 1 del presente documento.

8.1.5 Diagnóstico de los grupos experimental y control

La selección de los dos grupos estuvo basada en diversos aspectos, con el fin de validar la similitud inicial entre ellos y permitir la comparación de sus prácticas de desarrollo posterior a la intervención. Ambos grupos pertenecen a la misma institución educativa, para esta investigación se trabajó con la Escuela de Administración y Mercadotecnia del Quindío EAM; también hacen parte del mismo programa académico, Ingeniería de Software; y de igual forma hacen parte del mismo ciclo propedéutico, el ciclo tecnológico.

De manera adicional se realizó un análisis estadístico que permitiera validar la homogeneidad de los grupos experimental y control.

Para la interpretación de la información, también se agrupó por las categorías PSP propuestas para la investigación, es decir: Proceso, Tiempo, Tamaño, Defectos y Planeación, respectivamente.

A continuación, se expone el análisis de cada categoría:

- **Categoría Proceso:**

Las preguntas asociadas a la categoría de Proceso, fueron analizadas y permiten afirmar que se presenta homogeneidad en las medianas de las respuestas asociadas con las prácticas correspondientes.

La tabla 29 muestra el análisis estadístico para las medianas de las respuestas de la categoría Proceso para los grupos experimental y control.

Tabla 29 Análisis de homogeneidad Categoría Proceso – Grupos experimental y control

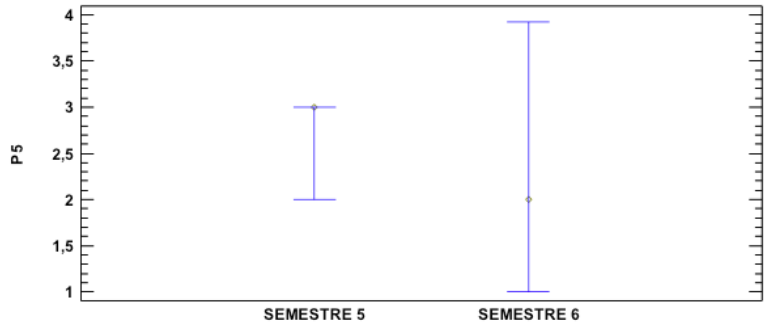
No. Pregunta	Estadístico de prueba	P-valor
1	0,912371	0,339484
2	0,0324074	0,857136
3	0,849772	0,356615
4	3,20689	0,0733249
5	0,840841	0,359155
6	1,09279	0,295851
7	0,447443	0,503551

Fuente: elaboración propia

El P-valor obtenido para la pregunta 5: ¿Realiza un diseño (diagramas y modelos) de lo que va a construir antes de iniciar su codificación?, permite determinar que no existe una

diferencia significativa entre las medianas de los grupos experimental y control. La figura 8 muestra gráficamente los resultados de las medianas para la pregunta mencionada.

Figura 8 Gráfico comparativo - Pregunta 5 - Instrumento de recolección de información – Grupos experimental y control



Fuente: elaboración propia

- **Categoría Tiempo:**

El análisis realizado a las medianas de las preguntas asociadas a la categoría de Tiempo, muestran a través del P-valor, que hay homogeneidad en sus resultados.

La tabla 30 muestra el análisis hecho a las medianas de las respuestas de la categoría Tiempo para los grupos experimental y control.

Tabla 30 Análisis de homogeneidad Categoría Tiempo – Grupos experimental y control

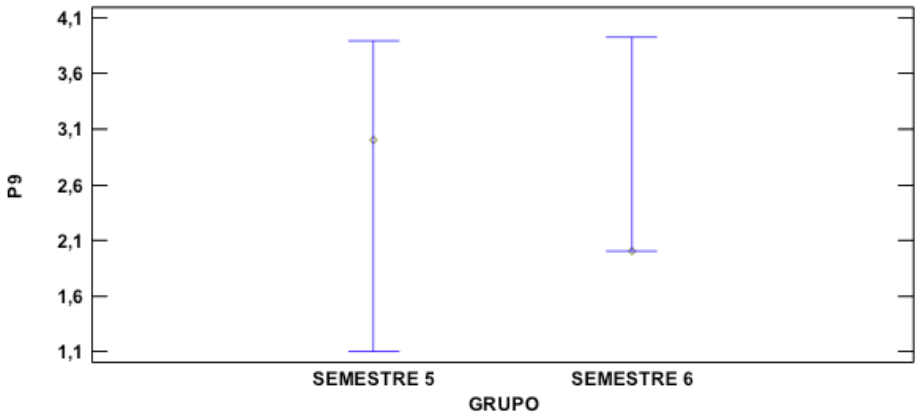
No. Pregunta	Estadístico de prueba	P-valor
8	0,073347	0,786524
9	0,051207	0,820976
10	0,373707	0,54099
11	0,82232	0,364501
12	0,486111	0,485666

13	0,112092	0,737775
14	0,140292	0,707386

Fuente: elaboración propia

Se muestra en la pregunta 9: ¿Identifica las actividades que le toman más tiempo al programar? que existe homogeneidad en las medianas de las respuestas de los grupos, asociadas a la práctica donde los estudiantes identifican las actividades que les toman más tiempo al programar, tanto para grupo control como para el grupo experimental, como se muestra en la figura 9.

Figura 9 Gráfico comparativo - Pregunta 9 - Instrumento de recolección de información – Grupos experimental y control



Fuente: elaboración propia

- **Categoría Tamaño:**

Las preguntas asociadas a la categoría de tamaño, fueron analizadas y permiten afirmar que no hay diferencia significativa entre las medianas de los grupos experimental y control, en las prácticas que los estudiantes aplican al realizar desarrollo de software.

La tabla 31 muestra el análisis hecho para las medianas de las respuestas de la categoría Tamaño para los grupos experimental y control.

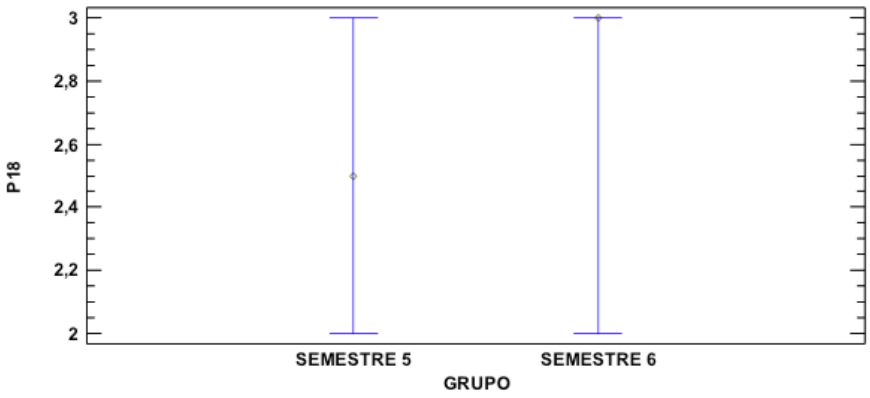
Tabla 31 Análisis de homogeneidad Categoría Tamaño – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
15	0,752193	0,385781
16	0,513703	0,47354
17	1,06579	0,301896
18	0,0606061	0,805541
19	0,330826	0,565173

Fuente: elaboración propia

Las medianas analizadas para la pregunta 18 ¿Estima la cantidad de líneas de código por hora que escribe cuando va a programar? muestran la homogeneidad de ambos grupos (control y experimental), al realizar esta práctica. En la figura 10 se puede visualizar dicha homogeneidad.

Figura 10 Gráfico comparativo - Pregunta 18 - Instrumento de recolección de información – Grupos experimental y control



Fuente: elaboración propia

- **Categoría Defectos:**

Al realizar el análisis asociado a las preguntas de la categoría Defectos, se evidencia que se presenta homogeneidad entre los grupos experimental y control cuando éstos ejecutan las prácticas relacionadas con la categoría.

La tabla 32 muestra el análisis que se realizó a las medianas de las respuestas de la categoría Defectos para los grupos experimental y control.

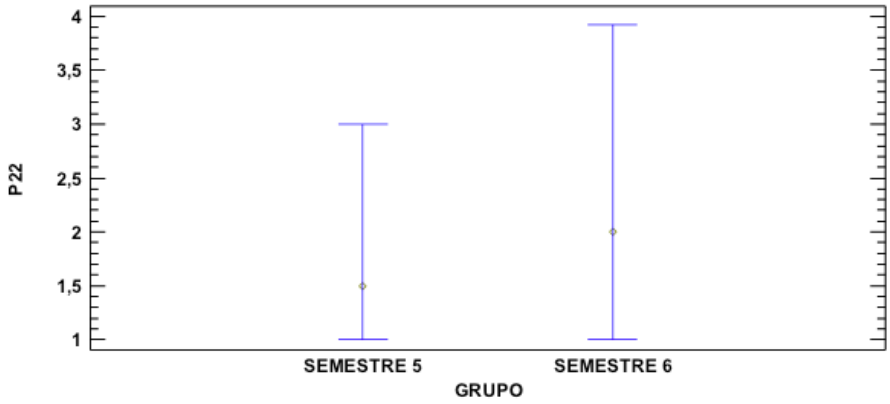
Tabla 32 Análisis de homogeneidad Categoría Defectos – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
20	0,988494	0,320109
21	2,21536	0,13664
22	0,0526316	0,818546
23	0,274378	0,60041
24	0,140902	0,707386
25	3,06806	0,0798412
26	0,140292	0,707991
27	0,612745	0,433755

Fuente: elaboración propia

Para la pregunta 22 ¿Registra información de los errores que comete al realizar un programa (su tipo, su causa, el lugar donde lo cometió, etc.)? los resultados muestran homogeneidad en las medianas de los grupos experimental y control al registrar datos de los defectos que los estudiantes inyectan cuando programan. La figura 11 muestra la homogeneidad entre los grupos.

Figura 11 Gráfico comparativo - Pregunta 22 - Instrumento de recolección de información – Grupos experimental y control



Fuente: elaboración propia

- **Categoría Planeación:**

Las preguntas asociadas a la categoría de Planeación fueron analizadas, permitiendo concluir que sus medianas muestran que existe homogeneidad en las prácticas PSP relacionadas.

La tabla 33 muestra el análisis asociado a las medianas de las respuestas de la categoría Planeación para los grupos experimental y control.

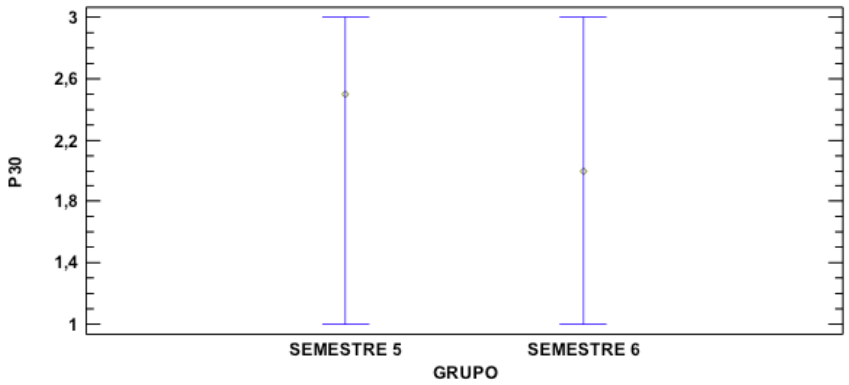
Tabla 33 Análisis de homogeneidad Categoría Planeación – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
28	0,42188	0,516
29	1,26812	0,260118
30	0,113636	0,736041
31	0,00582605	0,939158

Fuente: elaboración propia

El siguiente análisis está relacionado con la pregunta 30: ¿Planea la búsqueda de posibles errores en sus programas? evidencia la homogeneidad entre las medianas de los grupos control y experimental al momento de hacer la planeación de un desarrollo de software. La figura 12 muestra los datos relacionados.

Figura 12 Gráfico comparativo - Pregunta 30 - Instrumento de recolección de información – Grupos experimental y control



Fuente: elaboración propia

Teniendo en cuenta el análisis realizado con la prueba de Kruskal-Wallis, a los datos obtenidos con el instrumento de recolección, se pudo concluir que tanto el grupo control como el experimental son similares en cuanto a la forma con que habitualmente construyen su software.

El conjunto completo de las gráficas de homogeneidad al comparar los grupos control y experimental, se encuentran asociadas en el Anexo 2 del presente documento.

8.2 DISEÑO DEL CURSO

Al realizar el diseño del curso para realizar la intervención, se tuvo en cuenta el modelo curricular establecido para el programa de ISW de la EAM, buscando complementar la formación técnica de los estudiantes con el conocimiento y las habilidades necesarias para que gestionen apropiadamente su proceso de desarrollo de software.

La estructuración del proceso de aprendizaje del curso se hizo a partir de la metodología de proyectos formativos, aprovechando que el trabajo práctico de los estudiantes de programación es bastante similar al trabajo real para el cual son preparados, mediante la generación de proyectos de desarrollo de software que debieron hacer durante el curso de Programación Avanzada I.

8.2.1 Ruta formativa del proyecto

La definición de la ruta formativa permitió articular los elementos que la componen: Presentación del proyecto, las competencias a desarrollar, la estructura formal del programa, las actividades del proyecto y la estructura de la evaluación.

La ruta formativa fue presentada a los estudiantes en la primera sesión de trabajo. En ella se explicó el proceso general que se iba a iniciar, las competencias que se planeaban desarrollar, las actividades del proyecto y por último, la forma en la cual se evaluaría el proyecto.

Las competencias están divididas en dos grupos, el primero tiene que ver con las habilidades técnicas que los estudiantes deben desarrollar y que están relacionadas en la carta descriptiva del curso de Programación Avanzada I. La otra parte son las competencias en las prácticas PSP que se desea que los estudiantes puedan apropiar.

La tabla 34 muestra las competencias que se planearon desarrollar en los estudiantes.

Tabla 34 Competencias del proyecto formativo

Competencias a desarrollar	
Competencias técnicas	• Permitir al estudiante hacer uso de herramientas técnicas para solucionar aspectos de concurrencia en las soluciones. • Aprender a desacoplar con éxito las capas de presentación y lógica de negocio. • Estar en capacidad de trabajar con aplicaciones distribuidas y las diferentes formas de establecer conexiones a través de redes de computadores. • Comprender y utilizar la API JPA de Java.
Competencias PSP	• Construir programas de tamaño mediano y pequeño ejecutando las fases relacionadas con PSP nivel 0. • Aprender a gestionar la métrica del tiempo haciendo correcto uso del formato de registro de

	tiempos, registro de interrupciones, con datos completos y consistentes. • Registrar correctamente el formato de registro de defectos PSP. • Entender el estándar de defectos PSP que permita tenerlo como base para el registro de defectos. • Entender y seguir los guiones de PSP nivel 0. • Entender el uso y la importancia de los estándares de programación al construir software. • Aplicar el formato de registro de pruebas en la fase correspondiente. • Usar de manera consistente el formato de mejora de proceso durante la etapa de Post-Mortem.

Fuente: elaboración propia

La intervención tuvo una duración de dos meses, tiempo durante el cual se desarrollaron 16 sesiones de trabajo con los estudiantes. Las actividades realizadas se relacionan en detalle en el apartado 8.3.2 Desarrollo del curso.

La valoración de los diferentes entregables que los estudiantes generaron, se realizó a partir de rúbricas, instrumentos que permitieron agrupar los criterios a evaluar y los niveles que definían la evaluación de las competencias relacionadas con la apropiación de las buenas prácticas para el proceso personal de desarrollo de software PSP. Las rúbricas se encuentran establecidas en el anexo 4 Rúbricas, de este documento.

8.2.2 Fases del proyecto formativo

Con el fin de abordar la intervención sobre el curso, se establecieron 4 fases para desarrollar la estrategia de proyecto formativo, las cuales son las mínimas recomendadas por la metodología de proyectos formativos: direccionamiento, planeación, actuación-ejecución y socialización.

- **Fase de Direccionamiento:**

En esta fase, se estableció con los estudiantes la ruta formativa, que permitió guiar todo el proceso de enseñanza.

Se acordó con los estudiantes, trabajar temas y conceptos relacionados con la calidad en el software y como está influenciada por el proceso que se lleva a cabo para construir aplicaciones.

También se acordó trabajar las prácticas en torno a dos proyectos:

El primero es un proyecto basado en las dos primeras unidades temáticas: concurrencia y MVC; y sobre ese desarrollo, realizar las prácticas de registro de PSP0: tiempos y defectos.

El segundo proyecto está basado en otras dos unidades temáticas: programación distribuida y JPA, sobre el cual se ejecutarían prácticas relacionadas con la planeación, estándares de codificación, registro de pruebas y la ejecución de la fase de post-mortem con sus formatos de plan de resumen de proyecto y la propuesta de mejora de proceso.

La tabla 35 muestra las actividades, criterios y evidencias propuestas para la fase de Direccionamiento.

Tabla 35 Actividades de la fase de Direccionamiento

ACTIVIDADES	CRITERIOS	EVIDENCIAS
• Realizar la lectura del artículo "*La importancia de la calidad en el desarrollo de productos de software*" (Hernandez & Lomprey, 2008), y trabajar con el cuestionario entregado. • Participar con su opinión en el debate del tema de la calidad en el sofware.	• Comprendo el concepto de la calidad en el software y lo se ejemplificar y contextualizar en su entorno. • Entiendo como los errores en el desarrollo de software impactan en el producto final y en los objetivos para los cuales fue construido. • Valoro la importancia que tiene la calidad en el desarrollo de software y sus consecuencias cuando no es atendida adecuadamente.	Cuestionario resuelto de Introducción a la calidad del software.

- **Fase de Planeación:**

Para realizar esta fase, y con el fin de contextualizar a los estudiantes, se realizó una exposición donde el profesor explicó el concepto de proceso, su aplicación en la construcción de software y los principales conceptos de PSP.

Las características y aspectos más importantes del proyecto se discutieron en la fase de planeación en conjunto con los estudiantes del curso. Se contextualizó a los estudiantes de tal forma que comprendieran la relevancia de las actividades relacionadas con las buenas prácticas para desarrollo de software y del proyecto como evidencia de su trabajo.

Se estableció trabajar un primer momento del periodo de intervención explicando prácticas de PSP que se aplicarían sobre un proyecto del curso, con la asesoría del profesor. Las prácticas que se planearon trabajar fueron:

- Ejecución de la fase de planificación: Guión de la fase para PSP0.
- Ejecución de la fase de desarrollo: Guión de la fase para PSP0.
- Gestión del tiempo: Registro de tiempos a través del formato correspondiente.
- Registro de interrupciones.
- Gestión de defectos: Registro de tiempos a través del formato correspondiente.
- Manejo del estándar de defectos.
- Tamaño del software.
- Estándar de codificación.
- Ejecución de pruebas unitarias: Diligenciamiento de Formato de pruebas.
- Ejecución de la fase de post-mortem: Guión de la fase para PSP0.
- Diligenciamiento del formato de propuesta de mejora de proceso: PIP.
- Diligenciamiento de formato de resumen de plan de proyecto.
- Guión de proceso PSP0.

El proyecto sobre el cual se generaron estas prácticas estuvo basado en las unidades temáticas 1 y 2: Concurrencia y MVC.

El proyecto a desarrollar consiste en la implementación de un videojuego, donde el usuario selecciona un personaje que debe llevar desde un punto inicial hasta uno final. El personaje inicia su camino y mientras avanza se encontrará con diversos obstáculos que se mueven de manera horizontal, de izquierda a derecha, buscando detener su progreso.

Cada vez que el personaje choque con un obstáculo, perderá una vida y deberá retroceder hasta el punto inicial. Por el contrario, cuando termina de recorrer el camino, llegando a la meta establecida, pasará a un siguiente nivel, donde los obstáculos tienen un aumento en su velocidad de movimiento, haciendo más difícil conseguir el objetivo.

Las prácticas se fueron incorporando según iba avanzando el proceso de desarrollo. En esta oportunidad contaron con la asesoría del profesor para ejecutar el proceso y realizar el diligenciamiento de los formatos requeridos.

Posteriormente se tendría una segunda parte, donde se realizaría un nuevo proyecto para aplicar las prácticas vistas. Las prácticas PSP del segundo proyecto se realizarían de manera independiente por parte de los estudiantes y consistió en la implementación de sistemas que contenían chats, con el fin de intercambiar mensajes de texto a través de una red de computadores. La comunicación debió realizarse utilizando *sockets* como medio de intercambio y debía guardar la información en base de datos utilizando la API de JPA para ello.

Las actividades, criterios y evidencias propuestas para la fase de Planeación, se muestran en la tabla 36.

Tabla 36 Actividades de la fase de Planeación

ACTIVIDADES	CRITERIOS	EVIDENCIAS
• Los estudiantes deben desarrollar un taller basado en la presentación hecha por el profesor • El profesor expone el esquema de trabajo que se tendrá en la intervención: Inicialmente un proyecto donde se desarrollarán competencias de registro de tiempos y defectos. Posteriormente un segundo proyecto donde se trabajarán competencias de registro de tiempos, defectos, pruebas, desarrollo bajo un estándar de codificación y la ejecución de la etapa de post-mortem y los formatos PIP y resumen de plan de proyectos.	• Entiendo el concepto de Proceso y su aplicación en diversos contextos. • Identifico las ventajas que tiene el construir software bajo un proceso definido. • Estoy en capacidad de incorporar la calidad en un proceso que daba establecer. • Comprendo el concepto de métrica, su uso y propósito.	Taller resuelto de Proceso Personal de desarrollo de software.

Fuente: elaboración propia

- **Fase de Ejecución:**

En esta fase se realizó la exposición de los diferentes elementos PSP: formatos, estándares, guiones y métricas, su uso, sus ventajas y su aplicación en el proceso de desarrollo de software.

En la tabla 37 se exponen las actividades relacionadas con esta fase.

Tabla 37 Actividades de la fase de Actuación-Ejecución

ACTIVIDADES	CRITERIOS	EVIDENCIAS
• Aplicar los conceptos teóricos vistos en clase acerca de las prácticas PSP0 y PSP0.1, en cada una de las sesiones. • Realizar las actividades propuestas por el profesor.	• Comprendo los conceptos teóricos asociados a las prácticas PSP. • Estoy en capacidad de realizar las actividades propuestas.	Instrumentos de recolección de información en formatos PSP.

Fuente: elaboración propia

- **Fase de Socialización:**

La fase de socialización es la oportunidad del estudiante para mostrar la obtención de las competencias que se trazaron en el curso. Los estudiantes presentaron tanto los sistemas solicitados para la parte técnica: Videojuego construido con hilos-MVC y Sistema de chat con *sockets* y JPA, así como también los instrumentos de recolección de datos PSP propuestos: formato de registro de tiempos, formato de registro de defectos, formato de pruebas, formato de propuesta de mejora de proceso (PIP) y formato de resumen de plan de proyecto.

Las actividades de la fase de socialización se muestran en la tabla 38.

Tabla 38 Actividades de la fase de Socialización

ACTIVIDADES	CRITERIOS	EVIDENCIAS
• Presentar cada uno de los entregables propuestos en las actividades acordadas. • Valorar cada entregable y hacer la retroalimentación respectiva al estudiante.	• Entiendo la forma en la cual es posible incorporar buenas prácticas en mi proceso personal de desarrollo de software. • Entiendo y estoy en capacidad de aplicar los formatos de registro de tiempo, defectos, pruebas, PIP y resumen de plan de proyecto que fueron vistos en clase.	• Entregables generados por los estudiantes: Software solicitado y formatos de toma de datos PSP diligenciados. • Rúbricas con la valoración de las prácticas. La valoración es realiza por el profesor, compañeros del evaluado y por el propio evaluado.

Fuente: elaboración propia

8.2.3 Evaluación del proceso

Con el fin de realizar la evaluación de los resultados de la investigación, se estableció un proceso que consistió en la generación de rúbricas, una encuesta de buenas prácticas y un examen de conocimientos.

La valoración de las competencias se realizó en tres momentos diferentes:

- **Al inicio de la intervención:** Los estudiantes participaron de la aplicación de una encuesta de ejecución de prácticas PSP en el desarrollo de software por parte de cada uno de ellos.

- **Durante la intervención:** Se realizaron talleres, cuestionarios y el diligenciamiento de los formatos PSP que fueron valorados a través de rúbricas donde se determinó el nivel de dominio de los estudiantes con cada criterio evaluado. Estas rúbricas se encuentran relacionadas en el Anexo 4 del presente documento.

  Se realizaron 3 procesos interdependientes de evaluación de los entregables generados:

  - Evaluación del docente
  - Coevaluación, hecha por un estudiante compañero del evaluado
  - Autoevaluación, hecha por el propio estudiante evaluado

- **Al final de la intervención:** Los estudiantes participaron de un examen de conocimientos que se aplicó tanto al grupo experimental como al grupo control.

  El examen consistió en 16 preguntas, todas de selección múltiple con única respuesta, agrupadas por las mismas categorías de la encuesta de buenas prácticas de programación, distribuidas así:

  - 4 preguntas de proceso
  - 4 preguntas de administración de tiempo
  - 3 preguntas relacionadas con la métrica de tamaño
  - 3 preguntas de gestión de los defectos
  - 2 preguntas relacionadas con planeación

  De igual forma se ejecutó nuevamente la encuesta de buenas prácticas de programación que se aplicó al ciclo propedéutico de tecnología, es decir los grupos experimental y control.

8.2.4 Material de soporte

Para cada una de las sesiones intervenidas se construyó un conjunto de recursos de aprendizaje que permitió al profesor de la materia guiar cada uno de los temas tanto en la parte teórica como en la parte práctica.

- Guía: Este elemento permite al docente tener una hoja de ruta que lo contextualiza con el tema que se expondría en cada sesión. Un ejemplo de la guía se muestra en el anexo 5.

- Material teórico: Se entrega al profesor el contenido conceptual requerido para desarrollar la clase. Dado que los temas tratados son de prácticas PSP, regularmente se compone de conceptos teóricos y descripción del formato PSP de registro de información, de manera adicional se relaciona un ejemplo para ilustrar al profesor en el diligenciamiento del mismo del formato.
  Este elemento también inicia con un encabezado que contiene: Institución, Nombre del curso, Fecha y hora de la clase, Identificador de la guía y temática PSP.

- Actividad: Este elemento consiste en el trabajo que se propone al estudiante con el fin de trabajar en la práctica o tema visto en la clase. Las actividades consistieron en lecturas, talleres, cuestionarios y diligenciamiento de formatos PSP.

- Rúbrica: Se hizo entrega al profesor de la rúbrica correspondiente a cada una de las actividades que se realizaban por sesión. El profesor exponía a los estudiantes la rúbrica asociada a la actividad a realizar, con el fin de guiarlos hacia la consecución de la competencia vista desde el punto de vista del nivel de dominio.

- Instrumento de recolección de información: Este elemento se entregó en los casos que correspondían a prácticas PSP que requerían de registro de datos, como lo son tiempo, defectos, entre otros. Los elementos de recolección de información se relacionan en el anexo 6 de este documento.

8.2.5 Retroalimentación

El proceso de retroalimentación se realizó durante toda la intervención. Consistió en la evaluación de cada uno de los entregables generados por los estudiantes a partir de las sesiones del curso.

A través de presentaciones construidas a partir de las rúbricas, se muestra a los estudiantes una valoración cuantitativa entre 0.0 y 5.0 y una cualitativa consistente en la descripción de lo que se debe mejorar y el establecimiento del nivel de dominio de las competencias asociadas con la práctica PSP vista.

La retroalimentación hecha a los estudiantes se realizó a través de la plataforma de servicio de alojamiento de archivos en la nube que ofrece Google, llamada Google Drive. Su selección se debió a la familiaridad que los estudiantes tienen con las aplicaciones de Google, a la seguridad que ofrece, dado que maneja los mismos usuarios de gmail y sólo el profesor y el estudiante puede ver su información, y su acceso económico, dado que es una aplicación que ofrece 15 GB de almacenamiento de información de manera gratuita.

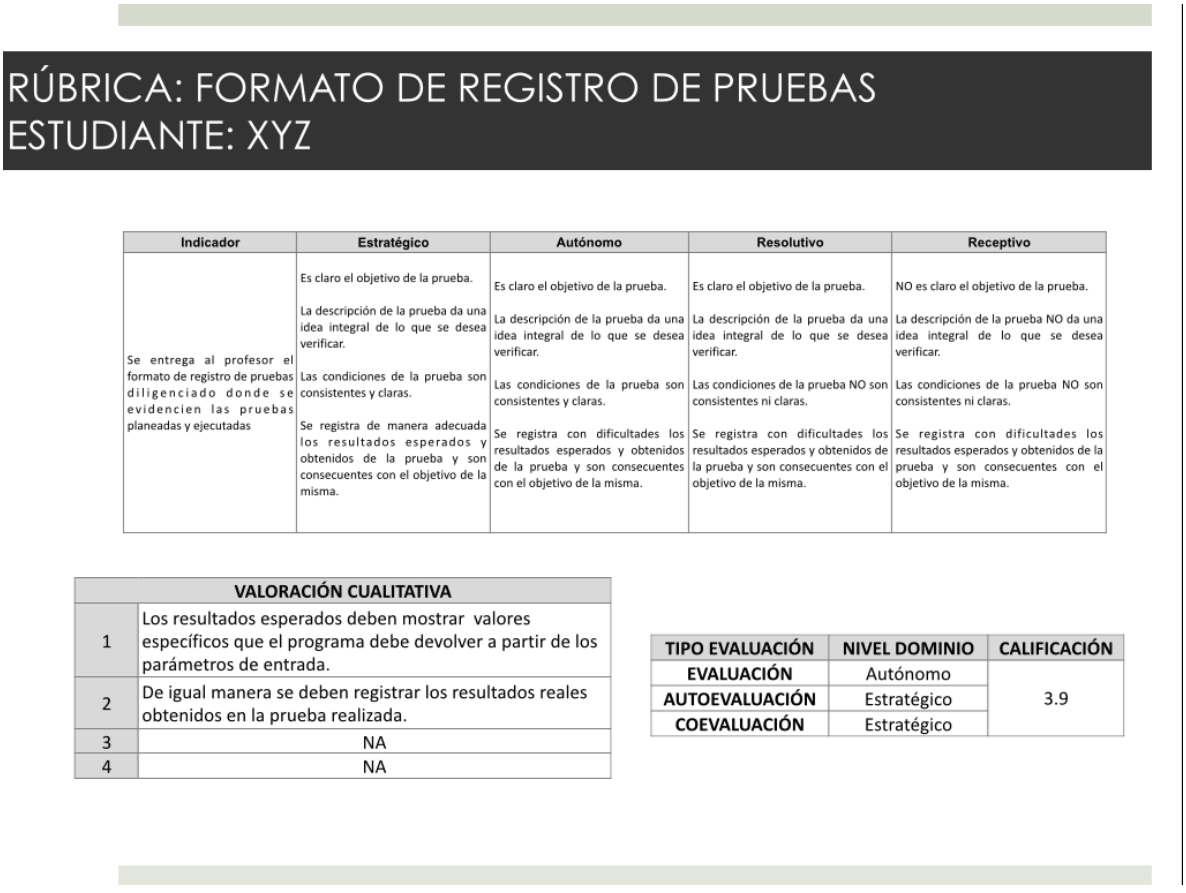
Por cada entregable al estudiante se le comparte la rúbrica, con la valoración del profesor, suya y la de uno de sus compañeros, así como también una nota cuantitativa del entregable y la descripción de los puntos a mejorar.

La siguiente es la relación entre ambos tipos de valoraciones:

- Calificación cualitativa: Nivel de dominio Receptivo y Calificación cuantitativa: Nota entre 0.0 y 2.9.
- Calificación cualitativa: Nivel de dominio Resolutivo y Calificación cuantitativa: Nota entre 3.0 y 3.7.
- Calificación cualitativa: Nivel de dominio Autónomo y Calificación cuantitativa: Nota entre 3.8 y 4.4.
- Calificación cualitativa: Nivel de dominio Estratégico y Calificación cuantitativa: Nota entre 4.5 y 5.0.

La figura 13 muestra el ejemplo de la retroalimentación realizada a un estudiante para una práctica PSP en particular:

Figura 13 Retroalimentación de práctica PSP



RÚBRICA: FORMATO DE REGISTRO DE PRUEBAS
ESTUDIANTE: XYZ

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se entrega al profesor el formato de registro de pruebas diligenciado donde se evidencien las pruebas planeadas y ejecutadas	Es claro el objetivo de la prueba. La descripción de la prueba da una idea integral de lo que se desea verificar. Las condiciones de la prueba son consistentes y claras. Se registra de manera adecuada los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.	Es claro el objetivo de la prueba. La descripción de la prueba da una idea integral de lo que se desea verificar. Las condiciones de la prueba son consistentes y claras. Se registra con dificultades los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.	Es claro el objetivo de la prueba. La descripción de la prueba da una idea integral de lo que se desea verificar. Las condiciones de la prueba NO son consistentes ni claras. Se registra con dificultades los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.	NO es claro el objetivo de la prueba. La descripción de la prueba NO da una idea integral de lo que se desea verificar. Las condiciones de la prueba NO son consistentes ni claras. Se registra con dificultades los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.

VALORACIÓN CUALITATIVA	
1	Los resultados esperados deben mostrar valores específicos que el programa debe devolver a partir de los parámetros de entrada.
2	De igual manera se deben registrar los resultados reales obtenidos en la prueba realizada.
3	NA
4	NA

TIPO EVALUACIÓN	NIVEL DOMINIO	CALIFICACIÓN
EVALUACIÓN	Autónomo	3.9
AUTOEVALUACIÓN	Estratégico	
COEVALUACIÓN	Estratégico	

Fuente: elaboración propia

8.3 IMPLEMENTACIÓN DE LA ESTRATEGIA DE ENSEÑANZA

8.3.1 Estructura del curso del grupo experimental

Para la intervención se planteó el siguiente esquema de articulación entre las temáticas PSP y la metodología a utilizar:

Unidad 1: Concurrencia. En la tabla 39 se muestra la intervención realizada en clase.

Tabla 39 Intervención Unidad 1: Concurrencia

TEMÁTICAS PSP	METODOLOGÍA
• Introducción a la calidad de software	Mediante técnica expositiva, el profesor expuso la necesidad de construir software con calidad y el compromiso que se debe adquirir como estudiantes de ingeniería de software para trabajar con buenas prácticas de programación y en un futuro como profesionales de la industria. A través de la técnica grupal de debate, los estudiantes expusieron sus puntos de vista acerca de la lectura que se realiza en clase.
• Proceso de desarrollo de software	El profesor explicó lo que es un proceso y expuso el proceso básico de PSP0 para construcción de software, destacando sus etapas y los principales entregables que en cada una se generan. Se resaltó la importancia de estructurar un proceso para construir software y como éste contribuye con la calidad del mismo. Se expusieron los principales conceptos PSP: métricas, formularios/logs, guiones y estándares. Se utilizó la técnica de mesa redonda para que los estudiantes expusieran sus ideas acerca de lo que es un proceso.
• Métricas en el software	Mediante técnica expositiva, el profesor explicó el concepto de Métrica, lo contextualizó en el proceso personal de desarrollo de software y destacó la importancia de las métricas en cualquier proceso y en particular en el proceso personal de desarrollo de software.

Fuente: elaboración propia

Unidad 2: Patrón Modelo-Vista-Controlador (MVC) usando el patrón *Observer*. En la tabla 40 se muestra la intervención realizada en clase.

Tabla 40 Intervención Unidad 2: MVC

TEMÁTICAS PSP	METODOLOGÍA
• Registro de tiempos (plantilla) • Gestión del tiempo (interrupciones) • Registro de defectos (plantilla y estándar) • Estándar tipos de defectos	A través de técnica expositiva, el profesor explicó las métricas de tiempo y defectos, su importancia, su registro y las contextualizó dentro del proceso personal de desarrollo de software. También se expusieron las plantillas para que los estudiantes aprendieran a incorporarlas en sus procesos. Debió destacar allí la importancia de las interrupciones y su correcto registro. Se presentó a los estudiantes el estándar de tipos de defectos, su uso en el formato de registro de defectos y las ventajas de utilizarlo.

Fuente: elaboración propia

Unidad 3: Programación Distribuida con *Sockets*. En la tabla 41 se muestra la intervención realizada en clase.

Tabla 41 Intervención Unidad 3: Programación Distribuida con Sockets

TEMÁTICAS PSP	METODOLOGÍA
• Guiones PSP0	El profesor entregó a los estudiantes los guiones de proceso, planificación, desarrollo y post-mortem, y explicó cada uno de los criterios de entrada, actividades y criterios de salida con el fin de motivar su uso en la construcción de programas por parte de los estudiantes.
• Planeación en el proceso de desarrollo de software	El profesor explicó la fase de planeación revisando con mayor detalle el guión correspondiente.
• Estándar de codificación	Mediante técnica expositiva, el profesor explicó lo que es un estándar de codificación, su uso y sus ventajas.

Fuente: elaboración propia

Unidad 4: Persistencia con JPA. En la tabla 42 se muestra la intervención realizada en clase.

Tabla 42 Intervención Unidad 4: Persistencia con JPA

TEMÁTICAS PSP	METODOLOGÍA
Pruebas unitarias - Registro de sus resultados	El profesor expuso la fase de pruebas y cómo hacerlas diligenciando el formato correspondiente. Explicó cada uno de los elementos que en el formato se encuentran.
Fase de Post-Mortem – Formato de resumen de plan de proyecto	A través de técnica expositiva, el profesor expone la fase de Post-Mortem que se realizar en el nivel PSP0, sus actividades y de manera adicional el formato de Resumen de Plan de Proyecto. Se explican las ventajas de esta que permite hacer una revisión retrospectiva de todo el proceso realizado.
Propuesta de mejora de proceso	El profesor mostró a los estudiantes el formato de Propuesta de mejora de proceso PIP, y explicó su uso.

Fuente: elaboración propia

Unidad 5: Aplicaciones Multicapa y RMI. En la tabla 43 se muestra la intervención realizada en clase.

Tabla 43 Intervención Unidad 5: Aplicaciones Multicapa y RMI

TEMÁTICAS PSP	METODOLOGÍA
• Examen de conocimientos • Recolección de información de buenas prácticas posterior a la intervención (Encuesta de buenas prácticas de programación) • Evaluación, autoevaluación, coevaluación • Retroalimentación final	El profesor y el investigador expusieron a los estudiantes la retroalimentación del curso: examen, aplicación de encuesta de buenas prácticas de programación, evaluación, coevaluación y autoevaluación. Posteriormente se aplicó un instrumento para conocer la opinión de los estudiantes acerca de la forma de evaluación de los proyectos formativos.

Fuente: elaboración propia

8.3.2 Desarrollo del curso

Las actividades se desarrollaron en una intervención que se realizó en 16 sesiones durante dos meses.

La estructura general de la intervención fue de la siguiente manera:

- Durante cada sesión se hizo una introducción inicial expositiva por parte del profesor de la materia, donde desarrollaba teóricamente la competencia PSP que se deseaba enseñar.
- Con el fin de tener una experiencia donde los estudiantes pudiesen aplicar sus conocimientos en su campo disciplinario, se propusieron dos proyectos cuya base técnica estuviera relacionada con las temáticas vistas en el curso:
    - El primer proyecto consistió en la construcción de un videojuego, haciendo uso de hilos para la concurrencia y el patrón de arquitectura MVC. El proyecto fue construido en lenguaje Java.
    - El segundo proyecto se construyó haciendo uso de la programación distribuida a través de *sockets* y con la API de JPA para la persistencia. La tecnología utilizada para la construcción de este proyecto también fue Java.

En la tabla 44, se describe el proyecto de concurrencia y MVC presentado a los estudiantes y con el cual se incorporaron prácticas PSP con la asesoría del profesor:

Tabla 44 Presentación del proyecto de concurrencia y MVC para apropiación de las prácticas PSP

Presentación del proyecto de Concurrencia y MVC	
Nombre	Videojuego de obstáculos
Justificación	Concurrencia: En las aplicaciones actuales, es bastante común requerir la ejecución de tareas de manera concurrente, es decir, múltiples tareas que se ejecutan de forma simultánea. Para el trabajo con concurrencia, Java cuenta con una estructura ligera que permite tener una línea de flujo de control propia y son los llamados threads o hilos. Modelo-Vista-Controlador: El Modelo-Vista-Controlador MVC es un patrón de arquitectura de software que permite separar los datos, la lógica y la presentación. Está basado en la reutilización de código y separación de conceptos para facilitar la construcción de aplicaciones y su mantenimiento. Teniendo en cuenta que estas aplicaciones tienen una cierta complejidad dado que se deben construir a partir de diversas tecnologías, es importante que los programadores utilicen buenas prácticas de programación para buscar generar una aplicación de calidad que satisfaga las necesidades que la generan. Estas buenas prácticas de programación pueden lograrse incorporando PSP en el proceso de los programadores.
Especificación	**Descripción General:** Los estudiantes deben implementar un videojuego donde un personaje evita obstáculos. El objetivo del juego es que se seleccione un personaje, que debe desplazarse desde un punto inicial hasta la meta. Éste debe esquivar una serie de obstáculos que se mueven de izquierda a derecha y que están por todo el recorrido que el personaje debe realizar. **Requerimientos funcionales a desarrollar:** • Requerimiento funcional 1: Este juego tendrá una temática de la EAM, con el fin de fomentar el proceso de acreditación. Para esto, los personajes a elegir serán un niño y una niña institucional e indicar su nombre de usuario. • Requerimiento funcional 2: El primer nivel se debe cargar con algunos obstáculos (pueden ser círculos o cualquier elemento que prefieran), que se moverán de izquierda a derecha. El personaje se controlará con el teclado, con movimientos de izquierda, derecha, arriba y abajo. Se deben esquivar los obstáculos para llegar a la meta (la meta simula uno de los 10 factores de acreditación que debe tener la EAM para alcanzar la acreditación). • Requerimiento funcional 3: Cuando se toca la meta (El factor de acreditación), se pasará al siguiente nivel, aumentando en 1 el número de obstáculos y su velocidad. Se indicará al lado derecho cuales han sido los niveles superados o los factores alcanzados. • Requerimiento funcional 4: El jugador cuenta con 2 vidas, y cada vez que es tocado por uno de

	los obstáculos perderá una vida y regresará al punto inicial del nivel. • Requerimiento funcional 5: Se debe contabilizar el tiempo que tarda el jugador en superar todos los niveles. Se debe registrar el mejor tiempo con su respectivo nombre de usuario. Este tiempo debe ser almacenado en un archivo para que pueda ser cargado posteriormente. • Requerimiento funcional 6: En cualquier momento el jugador puede reiniciar su juego. • Requerimiento funcional 7: Este debe ser desarrollado en JAVA, aplicando HILOS para su control. • Requerimiento funcional 8: La arquitectura utilizada para la construcción del videojuego, debe ser el Modelo-Vista-Controlador. **Requerimientos de calidad del software:** • Requerimiento de calidad 1: Realizar el registro de los tiempos (y sus interrupciones) y los defectos, de cada una de las fases ejecutadas. El registro se debe realizar en los formatos de registro de tiempos y de registro de defectos entregados en clase. Para la tipología de los defectos se debe tomar como base el estándar de tipos de defectos.

Fuente: elaboración propia

En la tabla 45, se describe el proyecto de *sockets* y JPA presentado a los estudiantes y con el cual se incorporaron prácticas PSP con la asesoría presencial y remota del profesor:

Tabla 45 Presentación del proyecto de sockets y JPA para apropiación de las prácticas PSP

Presentación del proyecto de *Sockets* y JPA	
Nombre	Sistema de intercambio de texto a través de una red de computadores.
Justificación	A continuación se exponen las 2 tecnologías que debe usar el proyecto a construir: Las aplicaciones distribuidas permiten acceso a ellas y la conducción de los datos a través de las redes de computadores. Una de estas formas de comunicación es a través de los *sockets*, que se pueden describir como un punto de enlace que permite a dos procesos comunicarse entre ellos. Al trabajar con *sockets*, se usan flujos de E/S que permiten enviar y recibir datos por la red. Estos datos hacen parte de mensajes que son construidos a través de protocolos que comunican a las aplicaciones distribuidas. Para atender el modelado de datos que requiera una aplicación, existe un framework para manejar datos relacionales en las aplicaciones. Este modelo es JPA, que utiliza unidades de persistencia que son clases Java cuyo estado es persistido en una tabla de una base de datos relacional. Con el fin de generar software de calidad, su construcción debe realizarse a partir de un proceso bien definido, como el que PSP ofrece con sus diversas prácticas.
Especificación	**Descripción General:** Los estudiantes deben generar una aplicación que permita acceder a un chat para facilitar la comunicación con otras máquinas en una misma red de datos. El chat se debe

	construir con *sockets* y los datos de los usuarios deben guardarse en bases de datos a través de JPA. La funcionalidad debe permitir a los usuarios comunicarse con otros que se encuentren en línea. Deben seguir las buenas prácticas de desarrollo de software PSP niveles 0 y 0.1 vistas en clase, con el fin de tener información relevante de su proceso de desarrollo que les permite hacer mejoras y conocer su desempeño. **Requerimientos funcionales a desarrollar:** • Requerimiento funcional 1: Cuando el usuario ingrese, la aplicación deberá cargar su nombre de usuario, y será este mismo el que lo identifique en el chat. • Requerimiento funcional 2: Cuando el usuario se conecte, el sistema revisará si existe algún historial de chat previo, para que éste sea cargado. • Requerimiento funcional 3: Deberá visualizarse la cantidad de usuarios conectados, realizando actualizaciones automáticas cada vez que un usuario se conecta o se desconecta. • Requerimiento funcional 4: Al utilizar el chat para enviar un mensaje, éste debe llegar a todos los usuarios que se encuentren conectados. • Requerimiento funcional 5: Cuando el usuario se desconecta, debe direccionarse a la página principal, que permita que el usuario u otro pueda ingresar otras credenciales de ingreso y pueda iniciar en el chat. • Requerimiento funcional 6: La tecnología utilizada para el desarrollo debe ser Java, y debe aplicarse hilos y *sockets* y persistir los datos utilizando JPA. **Requerimientos de calidad del software:** • Requerimiento de calidad 1: Realizar el registro de los tiempos (y sus interrupciones) y los defectos, de cada una de las fases ejecutadas. El registro se debe realizar en los formatos de registro de tiempos y de registro de defectos entregados en clase. Para la tipología de los defectos se debe tomar como base el estándar de tipos de defectos. • Requerimiento de calidad 2: En la fase de codificación, se debe seguir el estándar de codificación Java que propuesto y que permita seguir un mismo tipo de codificación para todos que haga el código entendible y mantenible. • Requerimiento de calidad 3: En la fase de pruebas, realizar el registro de sus resultados en el formato de registro de pruebas socializado en el curso. • Requerimiento de calidad 4: En la fase de Post-Mortem se debe realizar el registro del formato de Resumen de Plan de Proyecto entregado. • Requerimiento de calidad 5: En la fase de Post-Mortem se debe realizar el registro del formato de Propuesta de Mejora de Proceso (PIP: Process Improvement Proposal).

Fuente: elaboración propia

Para cada unidad temática se realizaron una serie de actividades que se ejecutaron durante las sesiones correspondientes.

Las actividades están organizadas por tipos:

- Técnica expositiva, corresponden a la exposición de manera verbal por parte del profesor.
- Actividad, son ejercicios que se realizaron en el salón de clases.
- Entregable, corresponden a ejercicios que debieron realizar de manera independiente.
- Producto, están relacionadas con prácticas PSP ejecutadas a partir de un proyecto formativo y que tienen asociada una rúbrica que permite valorar el dominio.

A continuación, se describe el trabajo realizado en el salón de clases, por unidad temática y su articulación con los temas PSP:

- **Unidad 1: Concurrencia**

Temática PSP: Introducción a la calidad del software. En la tabla 46 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 46 Intervención PSP: Introducción a la calidad del software

Objetivos	Unidad temática: • Aplicar hilos para crear aplicaciones concurrentes. • Usar el patrón Observador-Observable para la comunicación de la vista con la lógica de negocio. PSP: • Entender el concepto de calidad en el desarrollo de software. • Reconocer la importancia de construir software con calidad, teniendo en cuenta las altas exigencias requeridas de los sistemas actuales.
Tipo Actividad	**Descripción**
Técnica expositiva	El profesor realizó una introducción al curso en su parte técnica, destacando las diferentes tecnologías que estaban presentes en la temática para el semestre. Adicionalmente lo complementó hablando a los estudiantes de la necesidad de construir software de calidad, tal como lo requiere la industria actual. La charla tuvo como base el artículo "Importancia de la calidad en el desarrollo de productos de software" (Hernandez & Lomprey, 2008), que posteriormente sería leído y debatido en clase. Fue la oportunidad de iniciar el análisis acerca del papel de los estudiantes de programación entorno a la calidad, su responsabilidad al construir software y el impacto de los fallos del software a nivel todo nivel.
Técnica expositiva	El concepto entorno al cual gira toda la unidad temática 1 del curso de Programación Avanzada I es la concurrencia, que se entiende como el escenario en el cual un software requiere de la ejecución de varias tareas de manera simultánea. Se expone también la forma en la cual Java aborda esta necesidad y la propuesta para resolverlo. Los thread o hilos, es el objeto Java que permite, a través de la programación, ejecutar tareas que son lanzadas y detenidas de manera controlada. La clase, sus métodos y otros aspectos técnicos fueron abordados por el profesor.
Actividad	Los estudiantes reciben una copia física del artículo "Importancia de la calidad en el desarrollo de productos de software" (Hernandez & Lomprey, 2008), e inician su lectura con el fin de conocer con más detalle el concepto de calidad en el software y motivarlos a participar activamente durante la intervención.

Actividad	Posterior a la lectura del artículo, se realizó un debate con todos los estudiantes con el fin de escuchar sus opiniones acerca de la lectura, e ir identificando el interés por el tema. El debate se orientó a que los estudiantes expusieran su opinión acerca de los beneficios y dificultades de construir software con calidad. También se discutieron los inconvenientes de calidad en aplicaciones que usen concurrencia a través de hilos, como por ejemplo que distintos flujos de ejecución accedan a los mismos objetos del programa. Esto sin duda es un aspecto de calidad, que se puede prevenir con un diseño juicioso de la solución.
Entregable	Se solicitó a los estudiantes contestar un cuestionario escrito acerca del artículo, donde se les preguntó por su opinión con preguntas puntuales.

Fuente: elaboración propia

Temática PSP: Proceso de desarrollo de software. En la tabla 47 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 47 Intervención PSP: Proceso de desarrollo de software

Objetivos	Unidad temática: • Aplicar hilos para crear aplicaciones concurrentes. • Usar el patrón Observador-Observable para la comunicación de la vista con la lógica de negocio. PSP: • Entender el concepto de proceso y cómo éste se aplica al desarrollo de software. • Identificar las ventajas de tener establecido un proceso personal de desarrollo de software.
Tipo Actividad	**Descripción**
Técnica expositiva	Se aborda el concepto de Proceso y posteriormente se expone a los estudiantes en qué consiste el proceso personal de desarrollo de software PSP. Mediante exposición con diapositivas, se abordan los siguientes puntos de reflexión para los estudiantes: • Objetivos como programador • Qué es un proceso • Características de un buen proceso • Proceso personal de desarrollo de software • Características de un proceso personal (Consistente, eficiente, basado en el mejoramiento) • El foco de PSP • Principios PSP • Metodología PSP • Fases PSP • Ciclo de vida PSP
Técnica	Se exponen las fases de PSP de manera descriptiva y se indaga a los estudiantes para conocer si ejecutan

<table>
<tr><td>expositiva</td><td>algunas o todas las fases en su proceso de construcción de software.

Fase de planeación, Fase de desarrollo y Fase de Post-Mortem

Fase de desarrollo compuesta por : Diseño, Codificación, Compilación, Pruebas

Se explica el proceso PSP nivel 0 y se describen las fases existentes:

Fases del proceso (PSP0):

• Planeación: En esta fase se pretende tener un entendimiento de los requerimientos, que sean claros y no se presten para ambigüedades.
• Desarrollo
o Diseño: Busca que se genere un diseño conceptual de la solución. Pretende descomponer todo el sistema en partes para facilitar su análisis.
o Codificación: Construcción del programa en un lenguaje de programación.
o Compilación: Compilación del programa. Actualmente los IDE’s permiten hacer la compilación en tiempo de codificación. Por lo general esta etapa en lenguajes de alto nivel es obviada.
o UT (Unit Test): Las pruebas unitarias permiten validar que la solución construida es igual a la deseada. En esta etapa se generan los defectos que deben ser corregidos para tener una solución óptima.
• Post-mortem: Permite hacer comparaciones del trabajo planeado contra los resultados reales obtenidos.</td></tr>
<tr><td>Técnica expositiva</td><td>Se enuncian y describen los elementos PSP: guiones, estándares, formatos y métricas.

La descripción entregada a los estudiantes de cada uno de los elementos, fue:

Guiones:

• Guía experta, conjunto de pasos.
• Son menos específicos que los planes
• Están compuestos por criterios de entrada, pasos, criterios de salida.

Cómo proceder con los guiones:

• Verificar los criterios de entrada antes de iniciar.
• Registrar el tiempo de cada fase
• Ejecutar cada fase
• Registrar defectos inyectados y corregidos por fase.
• Verificar criterios de salida antes de terminar cada fase.
Se debe utilizar los guiones hasta que éstos sean un hábito al programar.

Métricas:

• Cuantifica procesos y productos.
• Permite determinar cuánto invierto en cada actividad.
• Cuantos defectos inserto y remuevo.
• Permite medir las fases (Tiempo invertido, defectos inyectados y removidos).
• Básicas: Tiempo, Tamaño, Calidad y Calendario

Formas:</td></tr>
</table>

	• Son los Repositorios donde se registra • Registro de tiempos • Registro de defectos Estándares: • Permiten realizar una clasificación uniforme. • Determinan los datos que se deben registrar siempre.
Técnica expositiva	Se aborda el concepto de hilo y se muestran los recursos Java para crearlos en los programas: la clase Thread y la interface Runnable. También se revisan sus constructores, atributos y métodos existentes.
Actividad	Se expone el proceso personal de desarrollo de software construyendo un ejemplo sencillo de una aplicación que usa hilos con la clase thread. La especificación solicita la creación de un programa que requiere: • Generar 3 tareas independientes que pueden ejecutarse en paralelo. • Cada vez que una tarea sea llamada, debe escribirse un mensaje en pantalla. • Debe poderse establecer el mensaje para cada proceso. El proceso personal que se debe realizar para abordar la solución es como sigue: Fase de planeación: El programador debe obtener la especificación y asegurarse de entenderla completamente, sin ambigüedades ni dudas. Fase de desarrollo: • Diseño: El problema a resolver es dividido en problemas pequeños, así: o Se generará una clase propia que extienda de Thread. o La clase debe tener un constructor y un método para enviarle el mensaje al proceso. o Debe sobreescribirse el método run y allí se realizará la impresión del mensaje en pantalla. o Para crear las 3 tareas, se creará un método principal. • Codificación: Basado en el diseño, el código es construido. public static void main(String[] args) { Tarea thr1 = new Tarea(Tarea 1"); Tarea thr2 = new Tarea(“Tarea 2"); Tarea thr3 = new Tarea(“Tarea 3"); thr1.setMensaje(“Mensaje de la tarea 1"); thr2.setMensaje("Mensaje de la tarea 2"); thr3.setMensaje("Mensaje de la tarea 3");

```
        thr1.start();

        thr2.start();

        thr3.start();

    }

    public class Tarea extends Thread{

        String msj;

        public Tarea(String mensaje){

            super(mensaje);

        }

        public void run(){

            for(int i =0;i<15;i++){

                System.out.println(msj);

            }

            System.out.println("Esta tarea ha terminado:"+this.getName());

        }

        public void setMensaje(String sMensaje){

            this.msj = sMensaje;

        }

    }
```

- Compilación: El código es compilado para encontrar posibles errores.
- Pruebas: Se establecen varias pruebas para validar el correcto funcionamiento de la solución. Todo error encontrado debe ser corregido.

Fase de Post-Mortem: Se hace una revisión del proceso. Se revisan tiempos. Se revisan los defectos encontrados. ¿Hay algún punto por mejorar? De ser así, debe aplicarse en el próximo desarrollo que se realice.

El objetivo del ejercicio fue solamente revisar cada fase y lo que a manera general debe hacerse, sin embargo, faltaron muchas actividades como la toma de tiempos, el registro de defectos, su corrección, etc.

Entregable	Se entrega un taller a los estudiantes, donde se les pide definir lo que entienden por Proceso y las ventajas que se tienen al establecer uno para desarrollar software. También se les pide: Teniendo en cuenta la presentación realizada acerca de lo que es el Proceso Personal de Desarrollo de Software, establezca cuál sería su proceso para la construcción de un automóvil, el cual debe poseer atributos de calidad, mediante los siguientes puntos: a. Liste las etapas que tendría el proceso y haga una breve descripción de cada una. b. ¿Qué atributos de calidad debe tener el automóvil y cómo los incorporaría mediante el proceso? c. ¿Cuáles serían las consecuencias de construir un automóvil sin atributos de calidad? d. ¿Qué métricas usaría y por qué?

Fuente: elaboración propia

Temática PSP: Métricas en el software. En la tabla 48 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 48 Intervención PSP: Métricas en el software

Objetivos	Unidad temática: • Aplicar hilos para crear aplicaciones concurrentes. • Usar el patrón Observador-Observable para la comunicación de la vista con la lógica de negocio. PSP: • Entender el concepto de métrica y su importancia en el análisis de información para mejorar el proceso personal de software. • Identificar las diferentes métricas que pueden tomarse, para ser analizadas y que permitan mejorar el proceso personal de desarrollo de software.
Tipo Actividad	**Descripción**
Técnica expositiva	Se expone el tema de las métricas: "Lo que no se mide, no se controla y lo que no se controla, no se puede mejorar". Basados en uno de los objetivos de PSP que es el mejoramiento continuo, el proceso busca llevar al programador a medir diversos aspectos de su proceso de desarrollo. **Métrica**: Medida que permite cuantificar procesos o productos. De manera adicional, se listan y describen algunas métricas de PSP: • Porcentaje de Tiempo en cada fase • Porcentaje de Defectos encontrados por fase • Porcentaje de Defectos removidos por fase • Tiempo en cada fase • Tiempo total de desarrollo • Tiempo de Interrupciones en cada fase • Tiempo de Interrupciones total • Cantidad de defectos encontrados por fase

	• Cantidad de defectos encontrados total • Cantidad de defectos removidos por fase • Cantidad de defectos removidos total • Tamaño del programa (KLOC) • Productividad (KLOC/HH) • Densidad de defectos (Defectos encontrados/KLOC)
Actividad	El profesor expone un ejemplo de métricas en el fútbol, con el fin que los estudiantes puedan entender el concepto y las ventajas de medir en pro de la mejora de resultados. Las métricas expuestas para el ejemplo son: Goles hechos, goles encajados, tiros al arco, tiros desviados, atajadas, pases acertados, pases errados, interceptaciones de pases. Los estudiantes proponen otras métricas en el fútbol y realizan el análisis de cómo conocer el dato entregado por la métrica, permite la mejora del equipo de fútbol.
Entregable	Se entrega un taller de métricas relacionado con el proyecto de hilos para que sea resuelto por los estudiantes, que contempla la selección de 3 métricas asociadas al proceso de construcción de ese proyecto, y sustentar: por qué la eligió (qué información esperaba que la medición le entregara), qué valores obtuvo y a qué conclusiones llegó.

Fuente: elaboración propia

- **Unidad 2: Patrón Modelo-Vista-Controlador (MVC) usando el patrón *Observer:***

Temática PSP: Administración de tiempo y defectos En la tabla 49 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 49 Intervención PSP: Administración de tiempo y defectos

Objetivos	Unidad temática: • Reconocer las ventajas de usar un patrón de diseño para resolver un problema. • Usar el patrón observador para desacoplar el modelo de la interfaz gráfica de usuario. PSP: • Incorporar el registro de las métricas Tiempo y Defectos en el proceso personal de desarrollo de software. • Aprender a utilizar las herramientas propuestas para el registro de tiempos y defectos. • Identificar las ventajas de llevar el registro de tiempos y defectos con el objetivo de tener información que permita mejorar el proceso personal de desarrollo de software.
Tipo Actividad	**Descripción**
Técnica expositiva	El profesor expone el tema de las métricas de tiempo, defectos inyectados y defectos encontrados. Se destacan como 2 de las métricas básicas de PSP.

	Métricas Básicas PSP: • **Tiempo** • Tamaño • **Calidad (Defectos)** • Datos de Calendario **Tiempo**: El registro de los tiempos destinados para cada una de las fases del proceso, permite hacer un análisis de la proporción de los tiempos invertidos en cada momento de la construcción de software. Proporciona al programador información para la mejora de su proceso personal de desarrollo de software. El registro de los tiempos debe incluir el registro de las interrupciones en cada una de las fases. Este es otro insumo que permite mejorar el proceso para la construcción de soluciones futuras. **Defectos:** El registro de los defectos permite tener información de los errores cometidos al momento de construir software. Su análisis posterior sirve como base para evitarlos y mejorar el proceso. Los defectos tienen dos momentos en PSP. El primero es su inyección, que corresponde al momento en el cual se comete el error en cualquier momento (fase) del proceso de construcción de software. Y el segundo es su detección e inmediata eliminación, que siempre es posterior a su inyección (por lógica) y que, de igual forma, también se hace en cualquier fase del proceso. El costo de un defecto encontrado y corregido es mayor cuando su fase es más avanzada. El proceso tiene como objetivo recolectar datos del trabajo realizado. ¿Qué datos se recolectan? 1. Tiempos por cada fase: Planeación, Desarrollo y postmortem. 2. Defectos inyectados en cada fase 3. Defectos removidos en cada fase
Técnica expositiva	• El profesor entrega a los estudiantes los formatos PSP de registro de tiempos y de registro de defectos, y el estándar de tipos defectos. Los formatos se encuentran en el Anexo 6 de este documento. • Se explica el uso del formato de registro de tiempos y el concepto y registro de interrupciones. • También se explica el formato de registro de defectos y como es usado en cualquier momento del proceso. • En esta sesión se hace entrega del estándar de tipos de defectos. Se explica como éste representa un insumo para diligenciar el formato de registro de defectos.

<table>
<tr><td></td><td>El Estándar entregado contiene los siguientes tipos de defectos:

<table>
<tr><th>Código</th><th>Nombre</th><th>Descripción</th></tr>
<tr><td>10</td><td>Documentación</td><td>Comentarios, mensajes</td></tr>
<tr><td>20</td><td>Sintaxis</td><td>Palabras mal escritas, puntuación, tipos, formatos de instrucciones</td></tr>
<tr><td>30</td><td>Empaquetamiento</td><td>Librerías, control de versiones</td></tr>
<tr><td>40</td><td>Asignación</td><td>Declaración, nombres duplicados, scope, limites</td></tr>
<tr><td>50</td><td>Interfaz</td><td>Llamado a procedimientos y referencias, entrada/salida, formatos de usuario</td></tr>
<tr><td>60</td><td>Chequeo</td><td>Mensajes de error, chequeos inadecuados</td></tr>
<tr><td>70</td><td>Datos</td><td>Estructura, contenido</td></tr>
<tr><td>80</td><td>Función</td><td>Lógica, apuntadores, loops, compilación, defectos de función</td></tr>
<tr><td>90</td><td>Sistema</td><td>Configuración, memoria, tiempo de sistema</td></tr>
<tr><td>100</td><td>Ambiente</td><td>Diseño, compilación, test o problemas con soporte de sistema</td></tr>
</table></td></tr>
<tr><td>Técnica expositiva</td><td>Se explica el proceso PSP con sus etapas de nivel 0, para ejemplificar la toma de tiempos y la detección y eliminación de defectos. Se aprovecha para explicar el proyecto de hilos y MVC y hacer un rápido proceso en el cual se toman datos de tiempo y se registran defectos encontrados y eliminados.

El proceso para desarrollar software se puede definir en la siguiente secuencia de tareas:

1. Identificar qué se quiere hacer (Planeación)
2. Establecer cómo hacerlo (Diseño)
3. Construirlo (Codificación)
4. Verificar su funcionamiento (Pruebas)
5. Corregir los errores encontrados (Pruebas)
6. Entregar el producto generado.</td></tr>
<tr><td>Producto</td><td>Teniendo en cuenta el proyecto de Hilos y MVC entregado a los estudiantes, se hace la solicitud del requerimiento de calidad número 1, relacionado en este documento en la tabla 43: Realizar el registro de los tiempos (y sus interrupciones) y los defectos, de cada una de las fases ejecutadas. El registro se debe realizar en los formatos de registro de tiempos y de registro de defectos entregados en clase. Para la tipología de los defectos se debe tomar como base el estándar de tipos de defectos.

Estimado estudiante, teniendo en cuenta el proyecto de hilos con MVC, siga el proceso de construcción de la solución a través de las siguientes fases:

Planeación

Diseño

Codificación</td></tr>
</table>

	UT-Pruebas unitarias Post-Mortem Realice el registro de tiempos y de defectos en las herramientas suministradas, con base en la explicación recibida en clase. Entregable: Adicional al videojuego, debe entregar la herramienta con los registros realizados de tiempos y de defectos.

Fuente: elaboración propia

- **Unidad 3: Programación Distribuida con *Sockets***

Temática PSP: Guiones PSP0. En la tabla 50 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 50 Intervención PSP: Guiones PSP

Objetivos	Unidad temática: • Crear aplicaciones distribuidas a través de *socket*. • Usar flujos de E/S para enviar y recibir datos por la red. • Crear protocolos de mensajes para la comunicación de aplicaciones distribuidas. • Usar la concurrencia en aplicaciones distribuidas. PSP: • Conocer los guiones de proceso, planificación, desarrollo y post-mortem de PSP0, sus criterios de entrada, actividades y criterios de salida.
Tipo Actividad	**Descripción**
Técnica expositiva	• Se expone el tema de los guiones como uno de los 4 elementos de PSP. • Se hace entrega de los guiones de proceso, planificación, desarrollo y post-mortem. • Se explica por cada uno los criterios de entrada, actividades y criterios de salida. • Se insta al grupo al seguimiento de los guiones para tener un proceso disciplinado y definido. **Guiones PSP** Consiste en tener una guía experta que explica cómo usar el proceso, y lo hace a través de un conjunto de pasos que corresponden a actividades que se deben realizar para la ejecución del proceso. Los guiones son menos específicos que los planes, y describen un propósito, unos criterios de entrada, unas directrices generales, unos pasos y unos criterios de salida. **Cómo proceder con los guiones** 1. Verificar los criterios de entrada antes de iniciar, es decir, los insumos que se deben tener para realizar el proceso.

2. Ejecutar cada fase contenida en el guión.
3. Registrar el tiempo de cada una de esas fases ejecutadas.
4. Registrar defectos inyectados y corregidos.
5. Verificar criterios de salida antes de terminar cada fase.
6. Se debe utilizar el guión hasta que se convierta en un hábito para el programador.

Los guiones de PSP0 hacen referencia al proceso actual del programador. A él se integran nuevos elementos que los guiones van describiendo. Este nivel de PSP está compuesto por 3 fases: Planificación, Desarrollo y Post-Mortem, cada uno con el guión correspondiente que ilustra la qué hacer. La fase de Desarrollo está conformada por las etapas de Diseño, Codificación, Compilación y Pruebas unitarias.

Este nivel permite recolectar datos del trabajo realizado, tales como tiempos por fase, defectos inyectados por fase y los defectos corregidos por fase. El objetivo es que provea una estructura que permita ejecutar tareas de mediana escala, además de proporcionar un marco para medir las tareas que se realizan. De manera adicional suministra una base de información que permite realizar mejoras al proceso actual de desarrollo de software.

Guión del proceso PSP0: Su propósito es guiar el desarrollo de los programas.

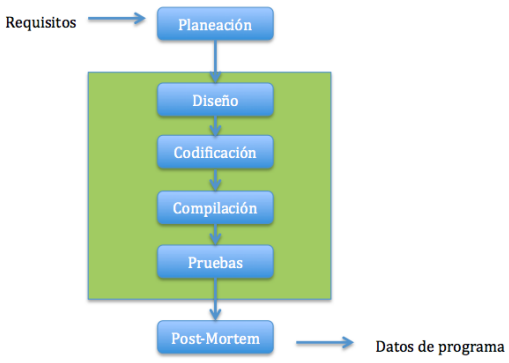
Guión de planeación PSP0: Su propósito es guiar el proceso de planeación PSP.

Guión de desarrollo PSP0: Busca guiar el desarrollo de pequeños programas.

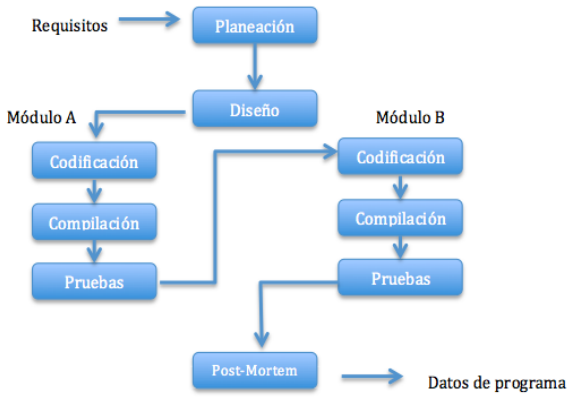
Guión de post-mortem PSP0: Su objetivo es guiar el proceso de post-mortem de PSP.

La ejecución de las fases de PSP0 descrita a través de sus guiones, permite tener diversas formas de abordar el proceso, dependiendo del tamaño del programa que se va a construir. A continuación se enuncian 3 ejemplos relacionados con cómo abordar la construcción de los programas según su tamaño y nivel de entendimiento de la funcionalidad a construir:

Para un programa de un tamaño pequeño y cuya funcionalidad se tenga bien entendida, es posible seguir el proceso ilustrado a continuación:



PSP0 también permite generar un proceso iterativo cuando se trata con programas grandes o que de los cuales no se tiene un completo entendimiento de la funcionalidad a construir. El siguiente esquema permite establecer cómo se puede ejecutar el proceso por iteraciones.

	La planeación y el diseño se hacen para todo el programa. En el diseño se determinan los módulos a realizar y posteriormente cada módulo se codifica, compila y prueba por separado. Los guiones PSP0 se ejecutan para cada fase establecida en las iteraciones sin modificación alguna. 
Técnica expositiva	El profesor expone los temas • Conceptos de comunicación TCP/IP • *Socket* y *ServerSocket* TCP/IP es un conjunto de guías para la implementación de protocolos de red, que habilitan el acceso de un equipo a una red. Los *sockets* representan uno de esos protocolos y sirve para el intercambio de flujos de datos. Java expone la clase *ServerSocket* como una de sus soluciones para conectar dos programas. Se revisó con los estudiantes la clase *ServerSocket*, sus constructores y métodos más comunes.
Actividad	Se explicó con un ejemplo sencillo: Juego de Ping Pong, cómo sería la construcción de un programa utilizando *sockets*, siguiendo el guión PSP de proceso. "El juego del Ping Pong consiste en enviar mensajes entre dos programas, el cliente inicia el juego enviando el mensaje "Ping" + consecutivo, y el servidor contesta "Pong" + consecutivo. Los mensajes deben enviarse utilizando *sockets* y debe ser construido en lenguaje Java". Para el proceso, se hace una revisión del guión y se especifica a los estudiantes que elementos tener en cuenta, teniendo como base la especificación dada.

Fuente: elaboración propia

Temática PSP: Planeación en el proceso de desarrollo de software. En la tabla 51 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 51 Intervención PSP: Planeación en el proceso de desarrollo de software

Objetivos	Unidad temática: • Crear aplicaciones distribuidas a través de *socket*. • Usar flujos de E/S para enviar y recibir datos por la red. • Crear protocolos de mensajes para la comunicación de aplicaciones distribuidas. • Usar la concurrencia en aplicaciones distribuidas. PSP: • Promover en los estudiantes la incorporación de la fase de planeación en los desarrollos que realicen.
Tipo Actividad	**Descripción**
Técnica expositiva	El profesor explica la fase de planeación revisando con mayor detalle el guión correspondiente. Se revisan los criterios de entrada, las actividades y los criterios de salida. Se motiva a los estudiantes para que hagan uso constante del guión de planeación.
Actividad	Con el mismo ejemplo del juego de Ping Pong que es construido utilizando *sockets*, se ejecuta la fase de planeación siguiendo el guión PSP correspondiente. Criterios de entrada: • Se verifica la descripción del problema: El juego del Ping Pong consiste en enviar mensajes entre dos programas, el cliente inicia el juego enviando el mensaje "Ping" + consecutivo, y el servidor contesta "Pong" + consecutivo. Los mensajes deben enviarse utilizando *sockets* y debe ser construido en lenguaje Java". • Se deben tener a la mano los formularios de registro de tiempo y resumen de plan de proyecto. Actividades: • Asegurarse que el programa es claro y sin ambigüedades. • Cada estudiante genera su declaración de requisitos del programa. • Cada estudiante debe hacer la estimación de tiempo para el programa. • Ingresar los datos en el formato de resumen de plan de proyecto. Criterios de salida: • Formatos de registro de tiempo y de resumen de plan de proyecto diligenciados. • Requisitos de programa documentados.

Fuente: elaboración propia

Temática PSP: Estándar de codificación. En la tabla 52 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 52 Intervención PSP: Estándar de codificación

<table>
<tr><td>Objetivos</td><td>Unidad temática:
Crear aplicaciones distribuidas a través de socket.
Usar flujos de E/S para enviar y recibir datos por la red.
Crear protocolos de mensajes para la comunicación de aplicaciones distribuidas.
Usar la concurrencia en aplicaciones distribuidas.
PSP:
Conocer y promover el uso de estándar de codificación.</td></tr>
<tr><th>Tipo Actividad</th><th>Descripción</th></tr>
<tr><td>Técnica expositiva</td><td>Se explica el estándar de codificación, que es el segundo estándar visto en el curso. El profesor hace entrega de un estándar de codificación de ejemplo, para el lenguaje Java.</td></tr>
<tr><td>Actividad</td><td>Se hace un ejemplo de cómo seguir el estándar de codificación, con el mismo juego de Ping Pong que se ha tenido en la unidad temática.

Estándar de codificación:
Comentarios de implementación: Existen 2 formas para comentar el código que escribimos:
Comentarios de bloque:
/*
* Se escribe el comentario tomando varias
* líneas consecutivas
*/
Comentarios de línea:
/* Se comenta en una sola línea */
// O también de esta forma
Comentarios de línea:
/* Se comenta en una sola línea */
// O también de esta forma</td></tr>
</table>

	Trozo de código del juego de ping pong:
	
	try {
	el_socket = conex.accept();
	ping = new DataInputStream(el_socket.getInputStream());
	pong = new PrintStream(el_socket.getOutputStream());
	la_pelota = ping.readLine();
	pong.println("Te devuelvo la pelota:" + la_pelota);
	pong.close();
	ping.close();
	el_socket.close();
	}
	
	Como se puede notar, el código no está siguiendo el estándar, dado que no se están realizando comentarios al interior del mismo. Este representa un error que se detecta en el proceso.

Actividad	Se solicita a los estudiantes que busquen en internet otros estándares Java, .NET y PHP.

Fuente: elaboración propia

- **Unidad 4: Persistencia con JPA:**

Temática PSP: Pruebas unitarias. En la tabla 53 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 53 Intervención PSP: Pruebas unitarias

Objetivos	Unidad temática: • Mapear una base de datos relacional a objetos. • Usar las anotaciones de JPA para crear entidades que representen una base de datos. • Crear una unidad de persistencia para configurar la conexión a la base de datos. • Crear consultas usando el JPQL. PSP: • Promover la incorporación de la fase de pruebas en el proceso personal de desarrollo de software. • Entender el uso del formato de registro de pruebas.
Tipo	**Descripción**

<table>
<tr><th>Actividad</th><th></th></tr>
<tr><td>Técnica expositiva</td><td>
El profesor describe la actividad de pruebas que debe seguirse en el proceso personal de desarrollo de software. Se toma como base el guión de desarrollo, en su actividad de pruebas, para explicar a los estudiantes.
<table>
<tr><th>Paso</th><th>Actividades</th><th>Descripción</th></tr>
<tr><td>4</td><td>Prueba</td><td>Probar hasta que no haya más errores
Corregir todos los defectos encontrados
Registrar los defectos en el formato de registro de defectos
Registrar los tiempos en el formato de registro de tiempos</td></tr>
</table>
También se entrega el formato de registro de pruebas y se explica cada una de sus partes. Este formato se encuentra en el Anexo 6 de este documento.
</td></tr>
<tr><td>Actividad</td><td>
El profesor muestra cómo realizar la fase y el registro de los datos encontrados en el formato de registro de pruebas, tomando como ejemplo un proyecto JPA ya construido. Estos son los resultados de la prueba realizada.
<table>
<tr><td>Institución</td><td>Escuela de Administración y Mercadotecnia del Quindío EAM</td></tr>
<tr><td>Programa</td><td>Ingeniería de Software</td></tr>
<tr><td>Semestre</td><td>5</td></tr>
<tr><td>Curso</td><td>Programación avanzada I</td></tr>
<tr><td>Nombre Estudiante</td><td>Juan Pérez</td></tr>
<tr><td>Identificador del programa</td><td>7</td></tr>
<tr><td>Nombre formato</td><td>Reporte de pruebas</td></tr>
<tr><td>Número/Nombre prueba</td><td>1 - Validación del registro de información del cliente</td></tr>
<tr><td>Objetivo de la prueba</td><td>Validar que al ingresar la información por pantalla, los datos se ven reflejados en la tabla Cliente en la base de datos.</td></tr>
<tr><td>Descripción de la prueba</td><td>1. Ingresar la url de la aplicación en el browser.
2. Ingresar los datos de login
3. En el menú, seleccionar Cliente - Adicionar nuevo cliente
4. Ingresar todos los datos del cliente
5. Seleccionar el botón Guardar</td></tr>
<tr><td>Condiciones de la prueba</td><td>1. El usuario que hace login debe existir en la aplicación
2. Debe existir la información paramétrica de tipo de documento y ciudad.</td></tr>
<tr><td>Resultados esperados</td><td>Debe mostrarse un mensaje en pantalla: "Cliente creado correctamente" y en BD en la tabla sCliente, deben aparecer los datos: "Pedro Arias", "CC", "1094098045", "Clle 1 Norte No 13-08", "Armenia"</td></tr>
<tr><td>Resultados obtenidos</td><td>Se muestra un mensaje en pantalla: "Cliente creado correctamente". En BD en la tabla sCliente, aparece los datos: "Pedro Arias", "CC", "1094098045", "Clle 1 Norte No 13-08", "Armenia"</td></tr>
</table>
</td></tr>
</table>

Producto	Mediante la especificación del proyecto de *Sockets* y JPA entregado a los estudiantes y referenciado en la tabla 44 de este documento, se solicita el Requerimiento de calidad 3: En la fase de pruebas, realizar el registro de sus resultados en el formato de registro de pruebas socializado en el curso.

Fuente: elaboración propia

Temática PSP: Fase de Post-Mortem. En la tabla 54 se muestra en detalle la intervención realizada en clase para esta temática PSP.

Tabla 54 Intervención PSP: Fase de Post-Mortem

Objetivos	Unidad temática: • Mapear una base de datos relacional a objetos. • Usar las anotaciones de JPA para crear entidades que representen una base de datos. • Crear una unidad de persistencia para configurar la conexión a la base de datos. • Crear consultas usando el JPQL. PSP: • Conocer la fase de Post-Mortem con el fin de promover en los estudiantes su ejecución en cada proceso de desarrollo de software que emprendan. • Identificar las actividades básicas que se deben realizar en la etapa de Post-Mortem. • Conocer el formato de Resumen de Plan de Proyecto, entender su uso y fin principal. • Promover en los estudiantes el uso del formato de Resumen de Plan de Proyecto.
Tipo Actividad	**Descripción**
Técnica expositiva	Se expone la fase de Post-Mortem que se realiza en el nivel PSP0, haciendo uso del guión correspondiente: criterios de entrada, sus actividades y criterios de salida y de la guía instruccional que se entregó al profesor. **Fase de Post-Mortem en PSP0** En esta etapa se pretende realizar el registro final de los datos del proceso, permitiendo resumir y analizar los datos del programa en el resumen del plan. Esta información adquiere gran valor, dado que da una idea general del proceso realizado, comparando el plan trazado con el desempeño real obtenido. Para el caso del nivel PSP0, los registros tenidos en cuenta son los de tiempo y de defectos. En esta fase se solicita al programador el registro de todos los defectos encontrados durante la ejecución del programa. De manera adicional, se solicita el tiempo total que se destinó para la construcción del

<table>
<tr><td></td><td>programa en todas sus fases, con el fin de tener un historial final que permita realizar la comparación para utilizarlo posteriormente y evitar cometer los mismos errores.

¿Qué debe hacer el programador en esta fase?

Después de terminar cada programa, los programadores deben realizar un análisis postmortem de la labor realizada. En esta fase, se debe actualizar el Resumen del Plan de Proyecto con datos reales y revisan cómo fue el proceso hecho al realizar su programa comparándolo contra el plan definido desde el principio.

Como complemento, los programadores deben actualizar sus datos históricos con la información de tiempos y defectos encontrados y corregidos. Cabe aclarar que, en un proceso adecuado, la cantidad de defectos corregidos debe ser igual a la cantidad de defectos encontrados. Esta fase sirve como reflexión para los programadores, para que examinen cualquier posibilidad de mejora que pueda tener el proceso y de ser necesario le hacen los ajustes necesarios al proceso completo o solamente a una parte de éste.</td></tr>
<tr><td>Técnica expositiva</td><td>Se expone el formato de Resumen de Plan de Proyecto. Se explican las ventajas de esta que permite hacer una revisión retrospectiva de todo el proceso realizado.

Formulario de Resumen de Plan de Proyecto

Este formulario permite agrupar en un sólo documento, los datos de tiempos del programa por fase y sus porcentajes, defectos inyectados por fase y sus porcentajes y defectos removidos por fase y sus porcentajes.

Por cada una de estas métricas, se toman los valores correspondientes a cada fase en el actual programa, también la sumatoria por fase de los programas construidos y por último el porcentaje correspondiente a esa sumatoria.

Ejemplo:

<table>
<tr><th>Tiempo en Fase (min.)</th><th>Plan</th><th>Real</th><th>A la Fecha</th><th>%A la Fecha</th></tr>
<tr><td>Planificación</td><td></td><td>20</td><td>110</td><td>2,53%</td></tr>
<tr><td>Diseño</td><td></td><td>15</td><td>85</td><td>1,95%</td></tr>
<tr><td>Codificación</td><td></td><td>250</td><td>1090</td><td>24,97%</td></tr>
<tr><td>Compilación</td><td></td><td>0</td><td>0</td><td>0%</td></tr>
<tr><td>Prueba</td><td></td><td>330</td><td>2590</td><td>59,33%</td></tr>
<tr><td>Postmortem</td><td></td><td>25</td><td>490</td><td>11,22%</td></tr>
<tr><td>Total</td><td>180</td><td>640</td><td>4365</td><td>100%</td></tr>
</table>

Ventajas de la fase de Post-Mortem

• Permite tener datos actualizados del programa (real, a la fecha y porcentaje a la fecha).

• Tener una revisión del desempeño real con respecto al plan establecido.

• Usar los datos personales para hacer ajustes en el proceso.

El formato de resumen de plan de proyecto se encuentra asociado al Anexo 6 de este documento.</td></tr>
<tr><td>Técnica expositiva</td><td>El profesor muestra y entrega a los estudiantes el formato de Propuesta de mejora de proceso PIP, y explica su uso.</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Institución</td><td>Escuela de Administración y Mercadotecnia del Quindío EAM</td></tr>
<tr><td>Programa</td><td>Ingeniería de Software</td></tr>
<tr><td>Semestre</td><td>5</td></tr>
<tr><td>Curso</td><td>Programación avanzada I</td></tr>
<tr><td>Nombre Estudiante</td><td>Juan Pérez</td></tr>
<tr><td>Identificador del programa</td><td>7</td></tr>
<tr><td>Nombre formato</td><td>Propuesta de Mejora de Proceso PIP</td></tr>
</table>
<table>
<tr><th>Número PIP</th><th>Descripción del problema</th></tr>
<tr><td>1</td><td>No se realizó el diligenciamiento del formato de tiempos de forma correcta, pues no se registró tiempo para la etapa de planeación</td></tr>
<tr><td>2</td><td>Se omitió la fase de diseño para iniciar más rápidamente la codificación, lo que originó un número bastante alto de defectos inyectados en codificación</td></tr>
<tr><td>3</td><td>NA</td></tr>
<tr><th>Propuesta PIP #</th><th>Descripción de la propuesta</th></tr>
<tr><td>1</td><td>Tener desde el inicio del proceso a mano el formato de registro de tiempos.</td></tr>
<tr><td>2</td><td>Seguir el guión de desarrollo tal como está especificado.</td></tr>
<tr><td>3</td><td>NA</td></tr>
<tr><td colspan="2">Notas y comentarios</td></tr>
<tr><td colspan="2">NA</td></tr>
</table>
</td></tr>
<tr><td>Producto</td><td>Mediante la especificación del proyecto de Sockets y JPA entregado a los estudiantes y referenciado en la tabla 44 de este documento, se solicita el requerimiento de calidad 4: En la fase de Post-Mortem se debe realizar el registro del formato de Resumen de Plan de Proyecto entregado.</td></tr>
<tr><td>Producto</td><td>Mediante la especificación del proyecto de Sockets y JPA entregado a los estudiantes y referenciado en la tabla 44 de este documento, se solicita el requerimiento de calidad 5: En la fase de Post-Mortem se debe realizar el registro del formato de Propuesta de Mejora de Proceso (PIP: Process Improvement Proposal).</td></tr>
</table>

Fuente: elaboración propia

8.3.3 Evaluación y Proceso de retroalimentación

A lo largo de toda la intervención hecha con el curso, se realizó el proceso de evaluación y correspondiente retroalimentación a los estudiantes. En esta parte del proceso se generaron dos instrumentos de evaluación:

1. Las rúbricas: Permitieron realizar una valoración cualitativa de las prácticas apropiadas por el estudiante, midiéndolo en niveles de dominio. Todas las rúbricas generadas se encuentran en el Anexo 4 del presente documento. Esta valoración se realizó sólo para el grupo experimental.

   A través de las rúbricas fueron evaluados los siguientes entregables y sus correspondientes indicadores:

   - Cuestionario de introducción a la calidad del software:
     - Indicador 1: Concepto de calidad en el desarrollo de software
     - Indicador 2: Evaluación del impacto de los errores en la calidad del software
     - Indicador 3: Comprensión de la importancia de la calidad en el software
   - Taller de Proceso personal de desarrollo de software:
     - Indicador 1: Concepto de proceso
     - Indicador 2: Ventajas de un proceso de software
     - Indicador 3: Incorporación del concepto de calidad en un proceso
     - Indicador 4: Concepto de Métrica
   - Taller de métricas en el desarrollo de software:
     - Indicador 1: Concepto de métrica
     - Indicador 2: Selección de métrica
     - Indicador 3: Uso de las métricas de software
     - Indicador 4: Valoración de las métricas
   - Formato de registro de tiempos de proyecto de Concurrencia y MVC:
     - Indicador 1: Ejecución de registro de tiempos
     - Indicador 2: Diligenciamiento de interrupciones
   - Formato de registro de defectos de proyecto de Concurrencia y MVC:

        - Indicador 1: Ejecución de registro de defectos
    - Formato de registro de tiempos de proyecto de *Sockets* y JPA:
        - Indicador: Se entrega al profesor el formato de registro de tiempos para todas las etapas que evidencien gestión del tiempo
    - Formato de registro de defectos de proyecto de *Sockets* y JPA:
        - Indicador: Se entrega al profesor el formato de registro de defectos que evidencien gestión de los defectos
    - Uso de estándar de codificación de proyecto de *Sockets* y JPA:
        - Indicador: Se presenta al profesor en un archivo comprimido (.zip), el código fuente del programa conforme al estándar de codificación Java entregado
    - Formato de propuesta de mejora del proceso de proyecto de *Sockets* y JPA:
        - Indicador: Se entrega al profesor el formato de Propuesta de Mejora de Proceso, donde identifique problemas encontrados y las propuestas de solución para próximas construcciones de programas
    - Formato de registro de pruebas de proyecto de *Sockets* y JPA:
        - Indicador: Se entrega al profesor el formato de registro de pruebas diligenciado donde se evidencien las pruebas planeadas y ejecutadas
    - Formato de resumen de plan de proyecto:
        - Indicador: Se entrega al profesor el formato de Resumen de Plan de Proyecto, donde se evidencie tiempo planeado inicialmente, tiempos por fase, defectos inyectados y eliminados por fase y en general la información correspondiente del formato

2. Examen de conocimientos: Validó los conceptos relacionados con el proceso y sus diferentes elementos. Permitió tener una valoración cuantitativa del conocimiento de los estudiantes. El examen de conocimientos fue aplicado tanto al grupo experimental como al grupo control.

   El examen validó conocimientos en las mismas 5 categorías en las que se direccionó la encuesta de buenas prácticas de programación: proceso, administración del tiempo, gestión de defectos, métrica de tamaño y planeación.

El anexo 7 del presente documento contiene el examen realizado.

Las valoraciones tanto cualitativas como cuantitativas fueron entregadas a los estudiantes durante todo el proceso, con el fin que pudieran analizar los puntos sobre los cuales debían mejorar. La información fue compartida a través de una plataforma de servicio de alojamiento de archivos en la nube, y es compartido con el estudiante y el profesor de la materia.

3. Percepción de los estudiantes sobre la evaluación: A los estudiantes se les realizó una encuesta para que compartieran sus opiniones acerca de su experiencia al tener la oportunidad de evaluar a sus compañeros, ser evaluados por ellos y autoevaluarse. La figura 14 muestra el instrumento utilizado.

Figura 14 Instrumento para conocer la percepción del estudiante ante la evaluación, autoevaluación y coevaluación de proyectos formativos

Esta encuesta tiene por objeto, conocer las opinión que los estudiantes del curso de Programación Avanzada I del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío, tienen acerca de su participación en la evaluación de los entregables asociados a la apropiación de competencias en las prácticas PSP.

Instrucciones: Registre a través de este instrumento, su opinión acerca de su participción en el proceso de evaluación de los entregables relacionados con la apropiación de competencias en las prácticas PSP, calificando las preguntas de uno (1) a cinco (5), donde:
(1) Está en total desacuerdo. (2) En desacuerdo . (3) Ni de acuerdo, ni en desacuerdo. (4) De acuerdo. (5) Está totalmente de acuerdo.

Pregunta	1	2	3	4	5
1. Hacer autoevaluación de mis evidencias, es de utilidad para mi proceso de formación?					
2. Evaluar las evidencias generadas por mis compañeros (coevaluación), es de utilidad para mi proceso de formación?					
3. La retroalimentación de la valoración de mis evidencias por parte del profesor, contribuye a mi proceso de formación?					
4. La retroalimentación de la valoración de mis evidencias por parte mis compañeros, contribuye a mi proceso de formación?					
5. La retroalimentación promueve la reflexión y la autoevaluación de los aprendizajes?					
6. La evaluación de evidencias por medio de rúbricas contribuyen a mi proceso de formación?					

Fuente: elaboración propia

8.4 POSTEST GRUPO CONTROL Y EXPERIMENTAL

Terminada la intervención con el curso, se aplicó nuevamente la encuesta de buenas prácticas de programación, de las cuales se describen sus resultados a continuación.

8.4.1 Análisis de homogeneidad entre los grupos control y experimental

Se describe a continuación el análisis de los datos realizado posterior a la intervención, y que corresponden a la revisión de la heterogeneidad presentada entre los grupos control y experimental, agrupado por las categorías PSP propuestas para la presente investigación: Proceso, Tiempo, Tamaño, Defectos y Planeación.

- **Categoría Proceso:**

Las preguntas asociadas a la categoría de Proceso, fueron analizadas y teniendo en cuenta los resultados obtenidos para los P-valor de las preguntas 1 a 7, se puede afirmar que se presenta una diferencia significativa entre las medianas de las respuestas asociadas con las prácticas correspondientes. Es decir, ya no existe la homogeneidad que se evidenció al inicio de la investigación entre los grupos para esta categoría.

La tabla 55 muestra el análisis para las medianas de las respuestas de la categoría Proceso para los grupos experimental y control, posterior a la intervención.

Tabla 55 Análisis de homogeneidad Categoría Proceso – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
1	6,09026	0,0135906
2	10,7768	0,00102726
3	10,725	0,00105645
4	9,17945	0,00244656
5	12,5263	0,000400992

6	9,51603	0,00203603
7	5,48145	0,0192167

Fuente: elaboración propia

Posterior a la intervención se pudo verificar a través de la herramienta de postest, que los estudiantes del grupo control tienen un uso menor de las prácticas de Proceso que los estudiantes del grupo experimental.

- **Categoría Tiempo:**

El análisis realizado a las preguntas de la categoría de Tiempo, muestra diferencias significativas en todas las preguntas, entre las medianas de los grupos experimental y control, demostrando heterogeneidad entre ellos, contrario a lo mostrado antes de la intervención.

La tabla 56 muestra el análisis para las medianas de las respuestas de la categoría Tiempo, para los grupos.

Tabla 56 Análisis de homogeneidad Categoría Tiempo – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
8	11,9306	0,000551843
9	10,9302	0,000945605
10	11,7743	0,000600189
11	9,61177	0,00193256
12	9,11467	0,00253474
13	13,4085	0,000250313
14	12,6764	0,000370042

Fuente: elaboración propia

El análisis muestra que los estudiantes del grupo experimental han apropiado y están usando algunas prácticas relacionadas con el Tiempo en mayor número que los estudiantes del grupo control, quienes no participaron de la intervención.

- **Categoría Tamaño:**

Tomando como base las medianas correspondientes a los grupos experimental y control para todas las preguntas analizadas con la prueba de Kruskal-Wallis de la categoría de Tamaño, que se realizó posterior a la intervención, se puede visualizar que se presentan diferencias significativas entre ellas.

La tabla 57 muestra el análisis hecho a las medianas de las respuestas de la categoría Tamaño para los 2 grupos después de la intervención.

Tabla 57 Análisis de homogeneidad Categoría Tamaño – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
15	11,1946	0,000819928
16	11,0374	0,00089246
17	12,1469	0,000491379
18	11,9535	0,000545108
19	11,0777	0,000873256

Fuente: elaboración propia

Los registros realizados en el instrumento de postest por parte de los estudiantes, indican que aquellos pertenecientes al grupo control tienen un menor uso de las prácticas relacionadas con el tiempo que los estudiantes que conforman el grupo experimental.

- **Categoría Defectos:**

Las preguntas correspondientes a esta categoría muestran en su análisis que hay diferencias significativas entre las medianas de los grupos, es decir que posterior a la intervención son heterogéneos.

La tabla 58 muestra el análisis para las medianas de las respuestas de la categoría Defectos para los 2 grupos.

Tabla 58 Análisis de homogeneidad Categoría Defectos – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
20	11,0629	0,000880273
21	12,9348	0,000322307
22	12,8475	0,000337696
23	10,71	0,00106505
24	12,7785	0,00035038
25	12,4771	0,000411706
26	10,9027	0,000959748
27	13,6782	0,000216808

Fuente: elaboración propia

El registro de defectos y la información relacionada en el instrumento de postest, es más, utilizada en las prácticas realizadas por los estudiantes del grupo experimental que los estudiantes del grupo control.

- **Categoría Planeación:**

Para todas las preguntas asociadas a la categoría de Planeación, se encuentra diferencias significativas entre las medianas para los grupos control y experimental, posterior a la intervención realizada.

En la tabla 59 se visualizan los datos del análisis para las medianas correspondientes a las respuestas de la categoría Planeación para los 2 grupos.

Tabla 59 Análisis de homogeneidad Categoría Planeación – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
28	12,9348	0,000322307
29	11,9784	0,000537885
30	12,2354	0,000468618
31	10,754	0,00104003

Fuente: elaboración propia

Las respuestas reportadas por los estudiantes en el instrumento de postest que se les realizó, mostraron que los integrantes del grupo control realizan menos prácticas de planeación en su proceso personal de desarrollo de software que los estudiantes del grupo experimental.

8.4.2 Análisis de homogeneidad entre el pretest y el postest para el grupo experimental

Se presenta el análisis realizado al grupo experimental, tomando como base los datos registrados a través de la encuesta de buenas prácticas de programación, comparando las medianas generadas antes de la intervención contra las medianas generadas posterior a ella. Se analizan las medianas para determinar la homogeneidad entre ambos momentos, agrupado por las categorías PSP propuestas para la investigación: Proceso, Tiempo, Tamaño, Defectos y Planeación.

- **Categoría Proceso:**

Tomando como base las medianas correspondientes a los datos del pretest y postest para el grupo experimental, para todas las preguntas analizadas con la prueba de Kruskal-Wallis de la categoría de Proceso, se puede visualizar que se presentan diferencias significativas entre ellas. La tabla 60 muestra el análisis mencionado.

Tabla 60 Análisis de homogeneidad Categoría Proceso – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
1	12,4318	0,000421796
2	15,8037	0,0000702651
3	10,0081	0,00155784
4	16,5	0,0000486509
5	16,6988	0,0000438095
6	10,491	0,00119898
7	11,1037	0,000861082

Fuente: elaboración propia

El análisis realizado muestra que los estudiantes del grupo experimental aumentaron la cantidad de prácticas relacionadas con su proceso personal de desarrollo de software posterior a la intervención realizada con la presente investigación.

- **Categoría Tiempo:**

Para todas las preguntas asociadas a la categoría de Tiempo, se encuentran diferencias significativas entre las medianas de los resultados del pretest y el postest para el grupo experimental.

La tabla 61 muestra los datos el análisis de las medianas para las respuestas de la categoría Tiempo.

Tabla 61 Análisis de homogeneidad Categoría Tiempo – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
8	13,1485	0,000287543
9	11,7939	0,000593878
10	14,245	0,000160487
11	12,5432	0,000397381
12	9,7618	0,00178101
13	12,5566	0,000394555
14	16,9853	0,0000376711

Fuente: elaboración propia

Se tuvo un aumento en las prácticas relacionadas con Tiempo, como el registro del mismo y la administración de las interrupciones en el proceso de desarrollo de los estudiantes del grupo experimental, posterior a la intervención realizada.

- **Categoría Tamaño:**

Las preguntas correspondientes a esta categoría muestran en su análisis que hay diferencias entre las medianas obtenidas en el pretest y el postest para el grupo experimental, es decir que el análisis muestra que son heterogéneos.

La tabla 62 muestra los datos del análisis respectivo.

Tabla 62 Análisis de homogeneidad Categoría Tamaño – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
15	11,3557	0,00075175
16	13,1196	0,000292016
17	14,3842	0,000149046
18	15,675	0,000075213
19	13,1196	0,000292016

Fuente: elaboración propia

La intervención en el grupo experimental generó un aumento de sus prácticas relacionadas con el tamaño del software que desarrollan, como lo muestran los resultados del instrumento de pretest comparado contra los resultados del instrumento de postest.

- **Categoría Defectos:**

El análisis realizado a las preguntas de la categoría Defectos, muestra diferencias significativas en todas las preguntas, entre las medianas de los resultados del pretest y postest del grupo experimental, demostrando heterogeneidad entre los datos.

La tabla 63 muestra el análisis para las medianas de las respuestas de la categoría Tiempo.

Tabla 63 Análisis de homogeneidad Categoría Defectos – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
20	10,1089	0,0014749
21	15,145	0,0000995634
22	16,4218	0,0000506996

23	14,8164	0,000118505
24	14,8164	0,000118505
25	13,6328	0,000222111
26	16,7797	0,0000419815
27	17,7465	0,0000252388

Fuente: elaboración propia

Todas las prácticas relacionadas con la categoría de defectos mostraron un aumento en su ejecución por parte de los estudiantes del grupo experimental, como lo muestran el análisis estadístico realizado. Actividades como encontrar los errores y tipificarlos, identificar la etapa donde más se inyectan errores o el propio registro de los defectos es más utilizado después de la intervención.

- **Categoría Planeación:**

Las preguntas asociadas a la categoría Planeación, fueron analizadas y los resultados muestran que en las preguntas 28 a 31, se presentan diferencias estadísticamente significativas entre las medianas del grupo experimental entre sus respuestas de pretest y postest.

La tabla 64 muestra el análisis para las medianas de las respuestas de la categoría Planeación.

Tabla 64 Análisis de homogeneidad Categoría Planeación – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
28	14,9606	0,000109779
29	15,8266	0,0000694201

30	13,2368	0,000274305
31	15,0254	0,000106075

Fuente: elaboración propia

Los estudiantes entendieron la importancia de realizar prácticas relacionadas con la planeación del proceso de desarrollo de software, por tal razón elevaron la práctica de las mismas posterior a la intervención.

8.4.3 Análisis de homogeneidad entre el pretest y el postest para el grupo control

En el presente apartado se realiza la comparación de los resultados obtenidos al aplicar el instrumento de pretest y posteriormente el instrumento de postest al grupo control. Dado que el grupo control no fue intervenido, se presupone de la homogeneidad de los datos a partir del análisis de las medianas, agrupado por las categorías PSP propuestas para la investigación: Proceso, Tiempo, Tamaño, Defectos y Planeación.

- **Categoría Proceso:**

Las preguntas asociadas a la categoría de Proceso fueron analizadas, permitiendo concluir que existe homogeneidad en las prácticas PSP relacionadas en los datos de pretest y postest del grupo control, pues no existe una diferencia significativa entre sus medianas.

La tabla 65 muestra el análisis de la categoría Planeación.

Tabla 65 Análisis de homogeneidad Categoría Proceso – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
1	1,16942	0,279518
2	1,92063	0,165783
3	0,754527	0,385045

4	0,17094	0,679277
5	0,00327279	0,954379
6	1,52129	0,217421
7	0,0030722	0,955798

Fuente: elaboración propia

Al aplicar los instrumentos de prestest y de postest a los estudiantes del grupo control, los resultados de la categoría de planeación, no muestran aumento ni decremento de su ejecución entre los dos momentos de la toma de información.

- **Categoría Tiempo:**

Las medianas que están asociadas a las preguntas de la categoría Tiempo para los datos de pretest y postest del grupo control, muestran homogeneidad en los resultados, dado que no existe diferencia significativa entre las medianas, exceptuando la pregunta 14: ¿Registra el tiempo de las interrupciones que se le presentan al construir software?

La tabla 66 muestra el análisis que se realizó a dichas medianas.

Tabla 66 Análisis de homogeneidad Categoría Tiempo – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
8	0,554113	0,456641
9	0,16637	0,683358
10	0,47619	0,490152
11	0,222049	0,637483
12	0,353388	0,552201

13	0,738706	0,390074
14	4,57143	0,0325063

Fuente: elaboración propia

Dado que no se realizó una intervención y que no se influyó de ninguna forma las prácticas PSP con el grupo control, no se mostraron cambios en la administración del tiempo por parte de los estudiantes al inicio y finalización del estudio.

- **Categoría Tamaño:**

Las preguntas asociadas a la categoría de tamaño, fueron analizadas y permiten afirmar que no hay diferencia significativa entre las medianas asociadas a los resultados de los estudiantes del grupo control, que registraron en el pretest y el postest.

La tabla 67 muestra el análisis hecho para las medianas de las respuestas de la categoría Tamaño.

Tabla 67 Análisis de homogeneidad Categoría Tamaño – Grupos experimental y control

No. Pregunta	**Estadístico de prueba**	**P-valor**
15	0,311688	0,576646
16	0,386999	0,53388
17	0,457143	0,498962
18	0,00377929	0,95098
19	0,740741	0,389422

Fuente: elaboración propia

No se encontraron cambios en la ejecución de las prácticas de la categoría Tamaño por parte de los estudiantes del grupo control, tomando como base los resultados de los instrumentos de pretest y postest para esa categoría.

- **Categoría Defectos:**

El análisis realizado a las medianas de las preguntas asociadas a la categoría de Tiempo, muestran a través del P-valor, que hay homogeneidad en los resultados de pretest y postest del grupo control.

La tabla 68 muestra el análisis hecho a las medianas.

Tabla 68 Análisis de homogeneidad Categoría Defectos – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
20	0,0130911	0,908908
21	0,47952	0,488639
22	1,12412	0,289031
23	0,00323939	0,954612
24	0,00333	0,953983
25	0,0520961	0,819455
26	0,153257	0,695442
27	0,0340136	0,853678

Fuente: elaboración propia

Los estudiantes del grupo control no aumentaron ni disminuyeron de manera significativa la ejecución las prácticas relacionadas con la administración de los defectos, teniendo en cuenta los resultados de los instrumentos de prestest y postest.

- **Categoría Planeación:**

Las preguntas asociadas a la categoría de Planeación, fueron analizadas y permiten afirmar que se presenta homogeneidad en las medianas de las respuestas asociadas con las prácticas correspondientes tanto antes de la intervención como después de ella para el grupo control.

La tabla 69 muestra el análisis estadístico para las medianas de las respuestas de la categoría de Planeación para el grupo control.

Tabla 69 Análisis de homogeneidad Categoría Planeación – Grupos experimental y control

No. Pregunta	Estadístico de prueba	P-valor
28	0,47952	0,488639
29	0,154525	0,694248
30	0,125589	0,72305
31	0,208741	0,647756

Fuente: elaboración propia

En los resultados de las prácticas relacionadas con la planeación del proceso personal de desarrollo de software por parte de los estudiantes del grupo control no se evidencian cambios relacionados con aumento o disminución de su ejecución.

8.5 RÚBRICAS: PROCESO DE EVALUACIÓN, COEVALUACIÓN Y AUTOEVALUACIÓN

Las rúbricas permitieron establecer un nivel de dominio para diferentes indicadores por práctica para cada estudiante. Los resultados se relacionan a continuación.

8.5.1 Evaluación

En la tabla 70 se muestran los porcentajes de estudiantes que en la evaluación fueron valorados en un nivel de dominio específico por cada uno de los entregables realizados en la intervención:

Tabla 70 Porcentaje de valoraciones de los estudiantes por niveles de dominio – Evaluación

Entregable	Niveles de dominio			
	Estratégico	**Autónomo**	**Resolutivo**	**Receptivo**
Cuestionario de introducción a la calidad del software	19,35%	80,65%	0,00%	0,00%
Taller de Proceso personal de desarrollo de software	32,35%	44,12%	17,65%	5,88%
Taller de métricas en el desarrollo de software	40,63%	40,63%	15,63%	3,13%
Formato de registro de tiempos	25,00%	37,50%	37,50%	0,00%
Formato de registro de defectos	12,50%	75,00%	0,00%	12,50%
Formato de registro de tiempos	0,00%	66,67%	22,22%	11,11%
Formato de registro de defectos	0,00%	88,89%	0,00%	11,11%
Uso de estándar de codificación	11,11%	33,33%	44,44%	11,11%
Formato de registro de pruebas	0,00%	66,67%	33,33%	0,00%
Formato de propuesta de mejora	33,33%	44,44%	22,22%	0,00%
Formato de resumen de plan de proyecto	22,22%	66,67%	11,11%	0,00%

Fuente: elaboración propia

La valoración a través de las rúbricas muestra que los estudiantes en términos generales apropiaron adecuadamente la mayoría de las prácticas. El entregable con mayor porcentaje de estudiantes valorados en el nivel de dominio Estratégico con un 40,63%, fue el Taller de métricas en el desarrollo de software, donde se les ilustró a los estudiantes la importancia de las métricas en el proceso de desarrollo de software. Por el contrario, el entregable con mayor porcentaje de estudiantes valorados en el nivel de dominio Receptivo con un 12,5% fue el relacionado con el formato de registro de defectos, muy probablemente por la dificultad al tipificar los defectos que encontraron en su proceso de desarrollo de software.

También se puede observar que la parte introductoria del curso (Introducción a la calidad del software, Proceso personal de software y Métricas de software) tuvo altos porcentajes

en niveles estratégico y autónomo, lo que indica que a nivel conceptual se lograron buenos resultados.

Los promedios mostrados muestran que el nivel de dominio más encontrado fue el Autónomo con un 58,6%.

8.5.2 Coevaluación

En la tabla 71 se muestran los porcentajes de estudiantes valorados en los diferentes niveles de dominio por cada uno de los entregables realizados, pero esta vez evaluados por un compañero del grupo (coevaluación).

Tabla 71 Porcentajes de estudiantes valorados por los niveles de dominio por entregable – Coevaluación

Entregable	**Niveles de dominio**			
	Estratégico	**Autónomo**	**Resolutivo**	**Receptivo**
Formato de registro de tiempos	44,44%	33,33%	22,22%	0,00%
Formato de registro de defectos	44,44%	33,33%	11,11%	11,11%
Uso de estándar de codificación	22,22%	33,33%	33,33%	11,11%
Formato de registro de pruebas	55,56%	33,33%	11,11%	0,00%
Formato de propuesta de mejora	30,00%	50,00%	20,00%	0,00%
Formato de resumen de plan de proyecto	55,56%	44,44%	0,00%	0,00%
Promedio	**42,06%**	**37,94%**	**16,30%**	**3,70%**

Fuente: elaboración propia

Las coevaluaciones en comparación con las evaluaciones se muestran superiores para varios de los entregables. Se puede observar que los estudiantes tienden a calificar a sus compañeros en un alto porcentaje en el nivel de dominio estratégico y autónomo. Esto puede deberse a la presión de no querer afectar a sus compañeros de clase.

Los entregables con el mayor porcentaje para el nivel de dominio Estratégico son los que tienen que ver con los formatos de registro de pruebas y de resumen de plan de proyecto.

El nivel de dominio con el porcentaje más alto es el Estratégico, lo que muestra que los estudiantes asumen que sus compañeros apropiaron las prácticas.

8.5.3 Autoevaluación

La tabla 72 se muestran el porcentaje de estudiantes valorados en cada nivel de dominio por cada entregable, en su autoevaluación. Sólo se aplicó para el proyecto final:

Tabla 72 Porcentajes de estudiantes valorados en niveles de dominio por entregable – Autoevaluación

Entregable	**Niveles de dominio**			
	Estratégico	**Autónomo**	**Resolutivo**	**Receptivo**
Formato de registro de tiempos	44,44%	44,44%	0,00%	11,11%
Formato de registro de defectos	22,22%	77,78%	0,00%	0,00%
Uso de estándar de codificación	33,33%	33,33%	33,33%	0,00%
Formato de registro de pruebas	22,22%	44,44%	33,33%	0,00%
Formato de propuesta de mejora	55,56%	33,33%	11,11%	0,00%
Formato de resumen de plan de proyecto	22,22%	77,78%	0,00%	0,00%
Promedio	**33,33%**	**51,85%**	**12,96%**	**1,85%**

Fuente: elaboración propia

En la autoevaluación se puede observar que en su mayoría los estudiantes se califican a sí mismos en el nivel de dominio Autónomo. Es probable que no quieran parecer pretenciosos valorándose en el nivel más alto, pero tampoco se quieren afectar con una valoración que esté por debajo de esa.

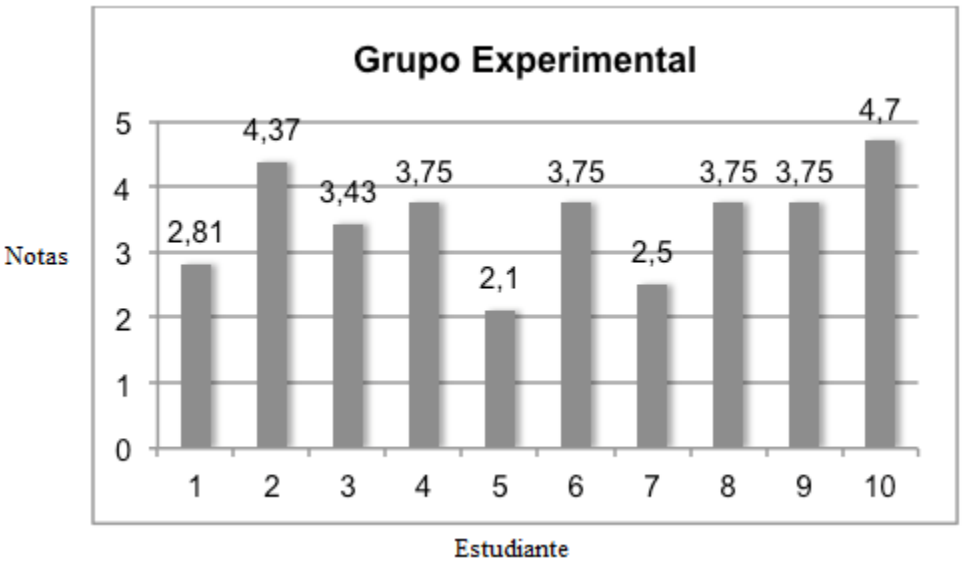
8.6 EXAMEN DE CONOCIMIENTOS

Al finalizar la incorporación de las prácticas y la realización de los proyectos formativos, se aplicó un examen de conocimientos que se puede visualizar en el Anexo 7 de este documento. Las preguntas fueron agrupadas con las mismas categorías PSP que se han

relacionado en este informe y validaron conceptos de las diferentes prácticas vistas. El examen tuvo 16 preguntas, todas con una igual ponderación, que sumadas generaron una nota entre 0 y 5.

La figura 15 muestra las notas obtenidas por estudiantes del grupo experimental.

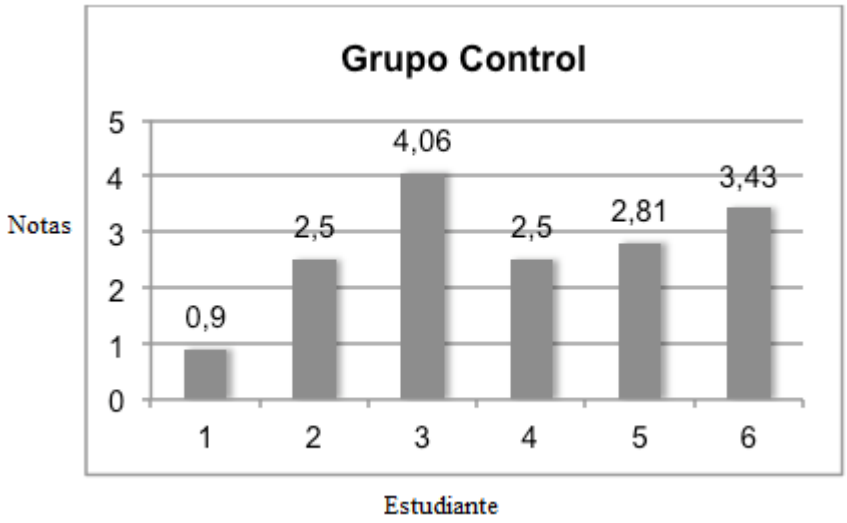
Figura 15 Notas de examen de conocimientos – Grupo control



Fuente: elaboración propia

Por su parte, la figura 16 muestra las notas obtenidas por estudiantes del grupo control.

Figura 16 Notas de examen de conocimientos – Grupo Control



Fuente: elaboración propia

Al realizar la revisión de los exámenes se encuentra que el promedio del grupo experimental es de 3,49, mientras que el promedio del grupo control fue de 2.7.

La tabla 73 muestra la cantidad de preguntas correctamente respondidas y erradas por cada categoría.

Tabla 73 Resultados por categoría – Examen de conocimientos

GRUPO EXPERIMENTAL				
Categoría	**Cantidad de respuestas correctas**	**Cantidad de respuestas incorrectas**	**Porcentaje bien respondidas**	**Porcentaje mal respondidas**
Proceso	34	6	85%	15%
Tiempo	25	15	62,5%	37,5%
Tamaño	25	5	83,33%	16,67%
Defectos	13	17	43,33%	56,67%
Planeación	16	4	80%	20%
GRUPO CONTROL				
Categoría	**Cantidad de respuestas correctas**	**Cantidad de respuestas incorrectas**	**Promedio Bien respondidas**	**Porcentaje mal respondidas**
Proceso	15	9	62,50%	37,50%

Tiempo	11	13	45,83%	54,17%
Tamaño	12	6	66,67%	33,33%
Defectos	6	12	33,33%	66,67%
Planeación	8	4	66,67%	33,33%

Fuente: elaboración propia

Los resultados que se muestran en las tablas anteriores, dejan las siguientes conclusiones:

1. La categoría donde hay un mayor dominio por parte del grupo experimental es la de Proceso, con un 85% de acierto. Para el grupo control las categorías de mayor dominio fueron Tamaño y Planeación, con un 66.67% de acierto.
2. La categoría donde más errores cometieron ambos grupos fue la de Defectos con un 56,67% para el grupo experimental y 66,67% para el grupo control.
3. La brecha en conocimiento más grande entre el grupo experimental y el grupo control está en la categroría de Proceso, donde hay un 22,5% de diferencia, a favor del grupo experimental.
4. En todas las categorías, el nivel de acierto del grupo control estuvo por debajo que el nivel de acierto del grupo experimental.
5. El estudiante con la mayor nota fue del grupo experimental con una nota de 4.7.
6. El estudiante con la menor nota fue del grupo control con una nota de 0.9.
7. Dos estudiantes del grupo experimental tuvieron notas superiores a 4.0, mientras que un estudiante del grupo control superó ese valor.
8. En el grupo experimental el 70% de los estudiantes ganó el examen, mientras que en el grupo control sólo el 33% pudieron ganarlo.

8.7 PERCEPCIÓN Y PARTICIPACIÓN EN LA EVALUACIÓN

Se realizó a los estudiantes una encuesta en la cual se les indagó por su apreciación acerca del aporte que la evaluación de los proyectos formativos tuvo en la intervención que se realizó con ellos.

Para ello, se pidió a los estudiantes valorar en una escala de 1 a 5, donde (1) Está en total desacuerdo. (2) En desacuerdo, (3) Ni de acuerdo, ni en desacuerdo. (4) De acuerdo. (5) Está totalmente de acuerdo.

Las preguntas fueron:

1. ¿Hacer autoevaluación de mis evidencias, es de utilidad para mi proceso de formación?

2. ¿Evaluar las evidencias generadas por mis compañeros (coevaluación), es de utilidad para mi proceso de formación?
3. ¿ La retroalimentación de la valoración de mis evidencias por parte del profesor, contribuye a mi proceso de formación?
4. ¿La retroalimentación de la valoración de mis evidencias por parte mis compañeros, contribuye a mi proceso de formación?
5. ¿La retroalimentación promueve la reflexión y la autoevaluación de los aprendizajes?
6. ¿La evaluación de evidencias por medio de rúbricas contribuyen a mi proceso de formación?

Los resultados de la encuesta de percepción de los estudiantes se muestra a continuación en la tabla 74:

Tabla 74 Encuesta de percepción

	Pregunta					
Estudiante	**1**	**2**	**3**	**4**	**5**	**6**
Estudiante 1	5	4	5	3	4	4
Estudiante 2	5	5	5	5	5	5
Estudiante 3	5	5	5	5	5	5
Estudiante 4	4	3	4	3	4	4
Estudiante 5	5	4	5	4	5	4
Estudiante 6	5	4	5	4	4	4
Estudiante 7	5	3	5	4	5	5
Estudiante 8	5	4	4	4	4	4
Estudiante 9	5	2	5	5	5	5
Estudiante 10	4	4	4	5	5	5

Fuente: elaboración propia

Los altos valores indican que en su mayoría, los estudiantes estuvieron de acuerdo en que los diferentes procesos de evaluación realizados (evaluación del docente, coevaluación y autoevaluación) les sirvieron para aportar en su formación en torno a las prácticas de desarrollo de software. La pregunta con mayor cantidad de estudiantes que estuvieron totalmente de acuerdo fue acerca de la autoevaluación de los aprendizajes (pregunta 1); por su parte donde los estudiantes se mostraron más en desacuerdo según los resultados, fue la pregunta que les indagó por su percepción de aporte en su formación cuando evalúan los trabajos de sus compañeros (pregunta 2).

9 DISCUSIÓN DE RESULTADOS

Desde el inicio de la investigación se propuso la generación de diversos productos de trabajo; éstos se construyeron a partir de la ejecución de las actividades planeadas. Inicialmente se generó el diagnóstico del nivel de apropiación y el uso de buenas prácticas de desarrollo por parte de los estudiantes a través de la encuesta acerca del proceso de desarrollo de software que ellos realizaban. Se encontró que había noción y ejecución de algunas prácticas, sin embargo se pudo evidenciar la oportunidad de incorporar otras no usadas en su proceso de construcción de software.

En el análisis del diagnóstico inicial se encontraron diferencias en las prácticas que los estudiantes realizaban que están directamente relacionadas con el ciclo propedéutico en el cual se encontraban. Los resultados mostraron que quienes menos buenas prácticas ejecutaban al construir software, eran los estudiantes del ciclo técnico, comparados con los estudiantes del ciclo tecnológico y éstos a su vez, por debajo de los estudiantes del ciclo profesional. El mismo análisis permitió determinar que cursos del programa de Ingeniería de Software serían los indicados para convertirse en el grupo control y experimental. Al valorar las prácticas de ambos grupos, se evidenció homogeneidad en la forma de construir software.

Esta información fue uno de los insumos para determinar el contenido teórico que se iba a impartir, así como también diversos aspectos integrados en el diseño de la estrategia de enseñanza y la metodología a utilizar. Apoyado en la metodología de Proyectos Formativos se buscó generar nuevas competencias no técnicas en los estudiantes a través de la práctica. La metodología era desconocida para el profesor titular de la materia, sin embargo fue desarrollada por fases, lo que facilitó su implementación. De manera adicional se diseñó el material de soporte que permitió al docente y estudiantes tener las herramientas necesarias para el desarrollo de las clases. También se diseñaron los momentos y tipos de evaluación que se tendrían, así como también los elementos que permitieran la validación, como los exámenes, las encuestas y las rúbricas.

La implementación de la estrategia motivó la integración de las prácticas PSP en las actividades propias de la materia de Programación Avanzada I, mediante la elaboración de

talleres, lecturas y proyectos donde se ponían en práctica los elementos teóricos impartidos, y su respectiva evaluación tal como se planteó en el diseño de la estrategia, que permitieron hacer seguimiento y retroalimentación a los estudiantes a través de las rúbricas. Finalmente se ejecutó la prueba piloto con la estrategia de enseñanza a través de proyecto formativos.

Se tuvo una respuesta positiva por parte de los estudiantes del curso hacia la generación de los entregables solicitados en clase a través de los proyectos formativos. Es probable que con otras estrategias de enseñanza se tuviesen de igual manera resultados positivos, sin embargo la metodología de proyectos formativos permitió alinear con facilidad las actividades prácticas propuestas y su evaluación con los objetivos del curso.

La totalidad de las actividades propuestas y descritas en el numeral 10.4.1 fueron ejecutadas conforme a la estructura del curso con el grupo experimental, cubriendo las temáticas técnicas del curso, temáticas PSP, la metodología y actividades propuestas.

Posterior a la intervención se realizaron diversas actividades que permitieron analizar con un mayor criterio los resultados obtenidos en el trabajo elaborado: encuesta de buenas prácticas de programación, encuesta de percepción de los estudiantes acerca de la evaluación, examen de conocimientos y las mismas evaluaciones realizadas durante el proceso.

De manera general, se puede afirmar que en todas las categorías se evidenció que existía diferencias entre los grupos experimental y control, situación que presentaba homogeneidad al iniciar la intervención. En ambos grupos se consiguieron los objetivos de formación técnica, sin embargo el grupo experimental obtuvo competencias asociadas a la calidad en su proceso pesonal de desarrollo de software.

Los estudiantes del grupo experimental presentaron buenos resultados en los conceptos relacionados con calidad en el software, el concepto de métrica, el registro de defectos y el diligenciamiento de resumen de plan de proyectos.

Al momento de registrar tiempos y defectos, algunos estudiantes tuvieron buenos resultados, sin embargo en otros se encontraron problemas de consistencia de los datos.

A nivel práctico se encontraron problemas al momento de hacer registros en dos etapas: planeación y post-mortem, pues varios estudiantes no realizaron los registros de tiempos de estas dos etapas. Al indagar la razón, argumentaron que desconocían que en esas etapas se debía registrar tiempos.

Como se mencionó en el numeral 3.2 de este documento, son ya varias las experiencias similares que se han realizado en diversas universidades alrededor del mundo, y todas coinciden en la necesidad de formar en competencias que puedan generar hábitos de calidad en los desarrolladores, sólo algunas relacionan una estrategia de enseñanza que facilite la apropiación de los buenos hábitos en el proceso personal de desarrollo de software, pero en ellas no se relaciona la generación de elementos de evaluación y retroalimentación de competencias como si se generó en la presente investigación.

Por último, los estudiantes tienen una percepción de mejora de su proceso de desarrollo de software, dada la autoevaluación realizada.

10 CONCLUSIONES

Los estudiantes inicialmente son reacios a trabajar con PSP, dado que no lo conocen y sienten que no le da un valor agregado a su labor, de allí que abordar el desarrollo de estas habilidades, deba ser un proceso cuidadosamente analizado teniendo en cuenta el currículo y los hábitos de codificación que tengan los estudiantes de programación.

Adicional a la incorporación de las buenas prácticas PSP en el curso, se tuvo un desafío con respecto a la utilización de la metodología de Proyectos formativos, dado que ni el profesor titular de la materia, ni los estudiantes habían trabajado previamente de esta forma. La orientación al profesor se basó en la entrega de las guías instruccionales del curso, que se crearon para cada sesión de trabajo con los estudiantes, el material teórico, los instrumentos de recolección de datos, las actividades propuestas para los estudiantes y las rúbricas.

Los proyectos a desarrollar fueron los sugeridos por el profesor titular del curso. Estos se adaptaron a la metodología de proyectos formativos, con el fin de solicitar los productos de trabajo que finalmente fueron evaluados, sin embargo, tener diferentes proyectos durante el curso no permitió generar un proyecto evolutivo realizado por ciclos donde se incorporaran paulatinamente las prácticas sugeridas.

La experiencia de este trabajo también puede considerar un acierto la selección de los grupos control y experimental, donde sus estudiantes se encuentran en semestres intermedios de su carrera. Fueron evidentes las competencias ya desarrolladas en el campo de la programación por parte de los estudiantes de grupo experimental, lo que les permitió dedicar parte de su esfuerzo en aprender de PSP, y no sólo enfocarse en adquirir habilidades técnicas. El grupo control, con sólo un semestre adicional en su formación, era bastante similar al grupo sobre el cual se realizó la intervención.

En algunas ocasiones los estudiantes expresaron su inconformidad con el registro de los datos en los formatos sugeridos, dado que debían cumplir tanto con los requerimientos funcionales como con el desarrollo de las buenas prácticas de desarrollo de software, sin embargo concluían que realizar los registros les permitía conocer su proceso y buscar la mejora del mismo.

El profesor titular de la materia manifestó que los estudiantes cumplieron con la entrega de los requerimientos funcionales de los proyectos asignados. No obstante, algunos puntos funcionales solicitados no se entregaban, pero no hay evidencias que permitan determinar que fuesen consecuencia del desarrollo de las buenas prácticas sugeridas para cada proyecto.

Se trabajaron prácticas y conocimientos agrupados en 5 categorías, y en todas se evidenciaron mejoras tanto teóricas como prácticas, en algunas en mayor medida que en otras. También se tuvieron buenos resultados relacionados con el compromiso con la calidad de sus proyectos, que fueron evidentes en las prácticas que realizaron.

Para la mayoría de los estudiantes se trató de su primera experiencia con PSP, porque si bien manifestaron en la encuesta de buenas prácticas de programación que se realizó al inicio de la investigación, que llevaban a cabo algunas prácticas, lo sostenían porque de manera intuitiva las realizaban, mas no tenían un proceso definido y disciplinado para construir software.

A través de la encuesta de buenas prácticas aplicada a la totalidad de estudiantes del programa en la etapa de diagnóstico, se pudo evidenciar que la ejecución de las prácticas se encontraba directamente relacionada con el ciclo propedéutico que estaban cursando, es decir, los estudiantes del ciclo técnico tienen una percepción similar del uso de las prácticas. Lo mismo pasó con el ciclo tecnológico, del cual se hizo puntualmente el análisis de homogeneidad entre los estudiantes de los cursos de Programación Avanzada I y Programación Avanzada II; y de igual forma con el ciclo profesional. También se encontró que los estudiantes tienden a utilizar con mayor regularidad las buenas prácticas entre más avanzado sea el ciclo propedéutico al cual pertenecen, los estudiantes del ciclo técnico utilizan menos prácticas que los estudiantes del ciclo tecnológico, y estos a su vez las utilizan menos que los estudiantes del ciclo profesional.

Posterior a la intervención, y después de ejecutar nuevamente la encuesta de buenas prácticas, pero en esta ocasión solamente con los grupos experimental y control, se realizó el análisis de los resultados comparando la incorporación de las prácticas PSP en el proceso de desarrollo de software de los estudiantes, encontrando que el grupo experimental tiende

a utilizar más prácticas que el grupo control. También se evidenciaron diferencias entre las prácticas realizadas por los estudiantes del grupo experimental antes y después de la intervención, aumentando la ejecución de las prácticas al momento de construir software en sus labores académicas.

11 RECOMENDACIONES

Intervenir un grupo de estudiantes de programación para tratar de incorporar buenas prácticas en sus procesos de desarrollo de software es una labor que requiere de analizar diversas condiciones, por ello se exponen acá algunas sugerencias a tener en cuenta:

1. Antes de realizar cada una de las prácticas PSP, es importante resaltar como cada una de ellas permite mejorar la calidad del desarrollo, esto con el fin de motivar al estudiante en ejecutar la práctica a conciencia y con la finalidad de definir un proceso de desarrollo que le aporte a generar un producto final de buena calidad. La motivación a trabajar con PSP puede ser apoyada en cifras que muestren la mejora obtenida en términos de tiempo, cantidad de defectos y aproximación al tamaño final del software.

2. Es recomendable que el profesor conozca la metodología a utilizar, pero en caso de no conocerla, se debe revisar previamente con él la capacitación a que haya lugar, con el fin que pueda constribuir en el direccionamiento adecuado del curso a través de la metodología seleccionada.

3. Como proyecto formativo, es recomendable tener uno que se pueda realizar por ciclos, permitiendo ejecutar el proceso varias ocasiones y poder evidenciar más fácilmente la mejora en la incorporación de las prácticas por parte de los estudiantes o en su defecto, validar en varios proyectos pequeños donde se inicie con un número reducido de prácticas y que se aumenten de manera gradual.

4. Sería ideal que los grupos experimental y control tuvieran estudiantes pertenecieran al mismo curso, sin embargo, si esto no es posible, se recomienda que los cursos sean similares y tengan el mismo profesor.

5. Se debe trabajar en un registro más automatizado de tiempos, y revisar estrategias entorno a la toma de los datos, formatos a través de alguna herramienta informática y todo aquello que reduzca el tiempo y esfuerzo invertido por los estudiantes en hacer los registros.

6. Es importante poder determinar si algún incumplimiento o la no obtención de las competencias técnicas es causada por la sobrecarga de trabajo al adicionar asignaciones relacionadas con las prácticas PSP.

7. Para la selección de las categorías PSP sobre las cuales trabajar, se recomienda identificar aquellas en las cuales los estudiantes tienen menos experticia o son las menos utilizadas. Este proceso puede guiar desde el diagnóstico los elementos a ilustrar en toda la investigación.

8. Para la presente investigación se trabajó durante el periodo académico correspondiente a un semestre. Se podría revisar si este tiempo puede aumentarse a dos semestres para tener mayor información, incorporar más prácticas, involucrar más proyectos, entre otras ventajas.

9. Los instrumentos de diagnóstico deben ser precisos y se debe buscar que suministren información que sea clara, no ambigua, que permita su interpretación y análisis posterior. Debe permitir ejecutarse varias veces para validar los avances en la adquisición de competencias de los estudiantes.

10. Sería interesante poder involucrar personas ya egresadas y con experiencia en la industria, para validar la complejidad de la apropiación de las buenas prácticas.

12 REFERENCIAS

Anijovich, R., & Mora, S. (2009). *Estrategias de enseñanza - Otra mirada al quehacer en el aula*. Buenos Aires, Argentina: Aique Grupo Editor.

Bermón-angarita, L., Fernandez, A., Carpio, D., Sanchez-segura, M. I., García-guzmán, J., & Seco, A. A. (2009). Experiencias Docentes en la Aplicación del Proceso Software Personal en Primero de Grado de Ingeniería Informática. In *FINTDI - Fomento e Innovación con Nuevas Tecnologías en la Docencia de la Ingeniería* (pp. 107–114). Vigo, España.

Börstler, J., Hislop, G. W., & Lisack, S. (2002). Teaching PSP : Challenges and Lessons Learned. *IEEE Software*, *19*(5), 42–48.

Cardona, S. A. (2011). *Diseño de una estrategia de aprendizaje para implementar prácticas de PSP y TSP en cursos básicos de programación. Caso programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío (Trabajo fin de máster)*. Universidad EAFIT.

Cardona, S., Vélez, J., & Tobón, S. (2014). Metodología de Proyectos Formativos: Estudio de Caso en un Curso de Fundamentos de Programación. In *IX Conferencia Latinoamericana de Objetos y Tecnologías de Aprendizaje LACLO* (pp. 225–236). Manizales, Colombia.

Diaz-Barriga, F., & Rojas, G. H. (2002). Estrategias docentes para un aprendizaje significativo (p. 148). México: Ed McGraw Hill.

Dymenstein, M., Etchamendi, A., & Maidana, F. (2011). Towards a student oriented approach to teaching PSP discipline A Brief overview of the Personal Software Process. In *Congreso Iberoamericano de Educación Superior en Computación*. Quito, Ecuador.

EAM, E. de A. y M. del Q. (2013). *Proyecto Educativo del Programa (PEP) por Ciclos Propedéuticos*. Armenia, Quindío.

Gobernación del Quindío. (2013). *Plan Regional de Competitividad del Quindío Comisión Regional*. Armenia, Quindío.

Hernández, M. A. (2013). Ventas de "software" en el país superarían los $5 billones. Retrieved from http://www.portafolio.co/negocios/ventas-%E2%80%98software%E2%80%99-el-pais-superarian-los-5-billones

Hernandez, S., & Lomprey, G. (2008). La importancia de la calidad en el desarrollo de productos de software. *Technical Report COMP-018-2008*. Retrieved from http://fit.um.edu.mx/CI3/publicaciones/Technical Report COMP-018-2008.pdf

Humphrey, W., & Over, J. (2000). *The Personal Software Process (PSP) tutorial*. *Software*

Engineering Institute. Pittsburgh: Carnegie Mellon University.

Humphrey, W. S. (2001). *Introducción al proceso personal de software PSP.pdf*. Madrid: Pearson Educación, S.A.

Kilpatrick, W. H. (1918). *The Project Method*. Teachers College.

Lisack, S. K. (2000). The Personal Software Process in the classroom: student reactions (an experience report). In *Software Engineering Education Training 2000 Proceedings 13th Conference on* (pp. 169–175). Austin, Texas. http://doi.org/10.1109/CSEE.2000.827035

Manrique, J. F. N., & Anaya, R. (2012). *Estudio empírico de aplicación de PSP para el desarrollo transversal de competencias de gestión, en estudiantes de un programa de tecnología de sistemas (Trabajo fin de máster)*. Universidad EAFIT.

Martínez Rodríguez, J. (2011). Métodos de Investigación Cualitativa. *Revista de Investigación Silogismo, 1*(08), 14. http://doi.org/10.5455/msm.2014.26.405-410

Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia. (2012). Talento Digital. Retrieved from http://talentodigital.mintic.gov.co/625/w3-article-8170.html

Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia. (2013). Convocatoria para el fortalecimiento de la competitividad de la Industria TI de Colombia. Retrieved from http://www.mintic.gov.co/portal/604/w3-article-7178.html

Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia. (2015a). Fondo para certificación en competencias específicas y transversales. Retrieved from https://www.icetex.gov.co/dnnpro5/es-co/fondos/fondosparaeldesarrollodeti/certificaci%C3%B3ncomptransversalesyespec%C3%ADficas.aspx

Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia. (2015b). Fortalecimiento de la Industria TI de Colombia. Retrieved from http://www.fiti.gov.co/

Pimienta, J. H. (2012). *Estrategias de Enseñanza-Aprendizaje*. México: Pearson Educación, S.A.

Pomeroy-huff, M., Cannon, R., Chick, T. A., Mullaney, J. L., Nichols, W. R., & Mullaney, J. (2009). *The Personal Software Process (PSP) Body of Knowledge , Version 2.0*. Pittsburgh.

Pozo, J. I. (2001). *Aprendices y maestros: La nueva cultura del aprendizaje*. Madrid: Alianza Editorial.

Rincón, R. D. (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*.

Universidad EAFIT.

Runeson, P. (2003). Using Students as Experiment Subjects – An Analysis on Graduate and Freshmen Student Data Per Runeson. In Citeseer (Ed.), *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering* (pp. 95–102). UK: Keele University.

SEI. (2015). Software Engineering Institute. Retrieved from http://www.sei.cmu.edu/about/organization/index.cfm

Solingen, R., & Berghout, E. (1999). *The goal/question/metric method*. *A Practical Guide for Quality Improvement of Software Development.* England: McGraw Hill. http://doi.org/10.1002/tcr.201090014

Soto, D. E., & Reyes, A. X. (2010). Introduciendo PSP (Proceso Personal de Software) en el aula. *Revista Colombiana de Tecnologías de Avanzada*, 1–5.

Tobón, S. (2005). *Formación basada en competencias: Pensamiento complejo, diseño curricular y didáctica* (Segunda Ed). Bogotá: ECOE Ediciones.

Torres-Gordillo, J. J., & Perera Rodríguez, V. H. (2010). La rúbrica como instrumento pedagógico para la tutorización y evaluación de los aprendizajes en el foro online en educación superior. *Pixel-Bit. Revista de Medios Y Educación*, (36), 141–149.

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). http://doi.org/10.1109/FIE.1997.644832

Universidad Pedagógica Experimental Libertador. (2006). Estrategias de aprendizaje - investigación documental- (parte B). Retrieved October 12, 2015, from http://www.redalyc.org/articulo.oa?id=76109916

Valtierra, M. G. G., Salinas, G. F., García, G. G., Murillo, J. de J. C., Santoyo, M. C., & Bárcenas, L. M. V. (2007). Trayecto formativo. Programa nacional para la actualización permanente de los maestros de educación básica en servicio. México D.F.: SEP Secretaría de Educación Pública.

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*. Charlotte, North Carolina, USA.

Wackerly, D., Mendenhall, W., & Scheaffer, R. (2010). *Estadística matemática con aplicaciones* (Séptima Ed). México D.F.: Cengage Learning.

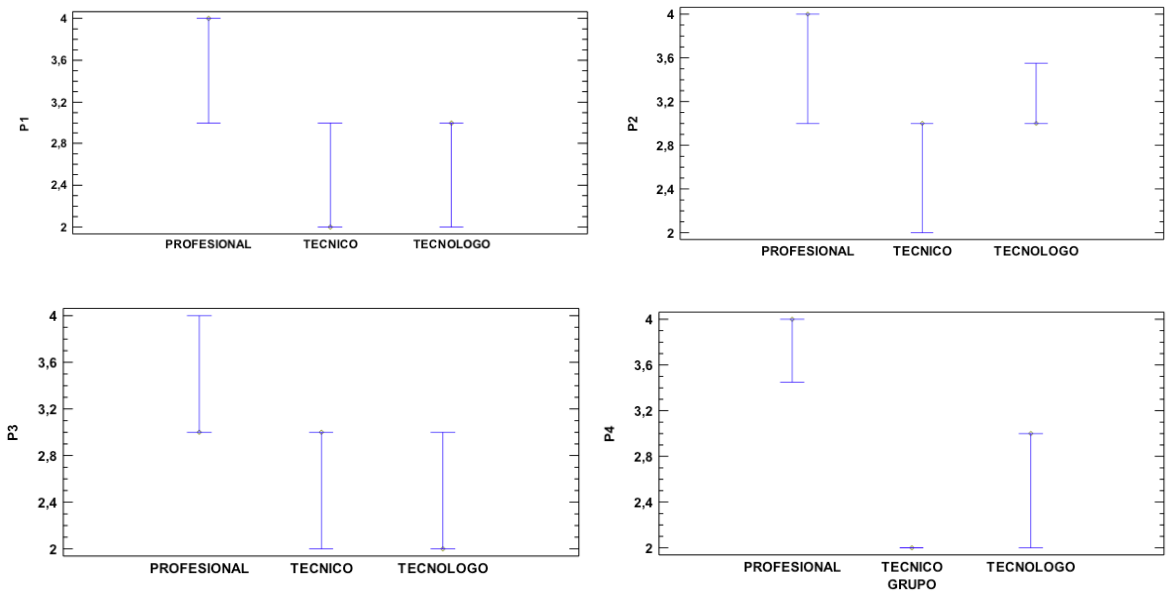
13 ANEXOS

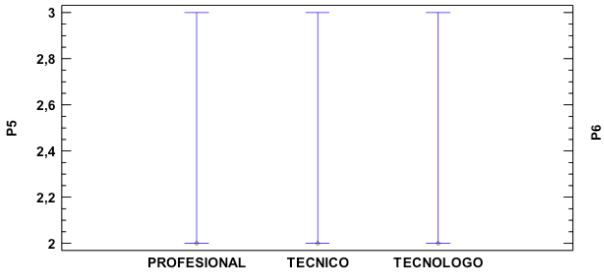
Anexo 1: Gráficos de homogeneidad del análisis estadístico de diagnóstico para los ciclos propedéuticos

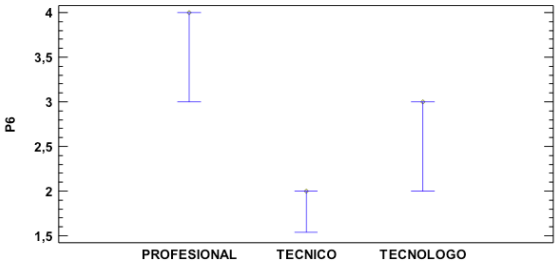
- **Categoría Proceso PSP:**

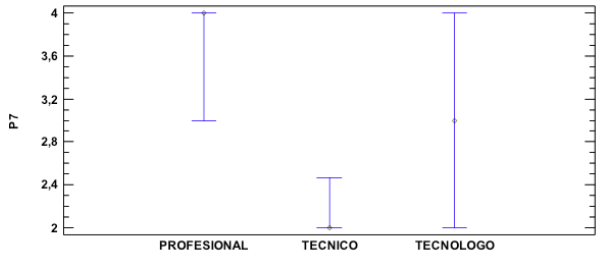
La heterogeneidad entre las medianas de las respuestas de la categoría Proceso entre los ciclos propedéuticos es mostrada en las gráficas de las preguntas 1, 2, 3, 4, 6 y 7, que se relacionan a continuación. Sin embargo, la pregunta 5 muestra homogeneidad en la práctica relacionada.

Para la figura que ilustra la pregunta 6: "Cuando usted programa, tiene en cuenta las actividades que realizan otros desarrolladores para incorporarlas dentro de sus propias actividades y obtener mejores resultados?", los valores obtenidos de las medianas muestran que existe una diferencia significativa entre ellas. Los resultados de todas las preguntas, exceptuando la número 5, son similares en su análisis.





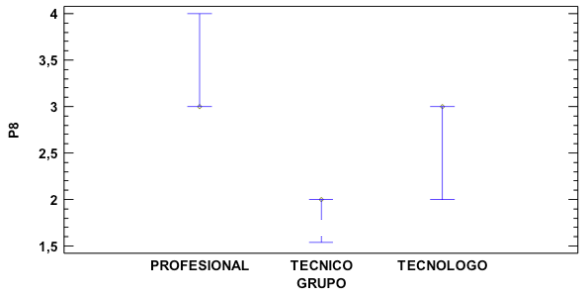


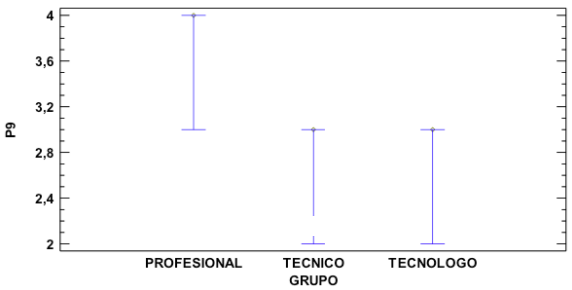


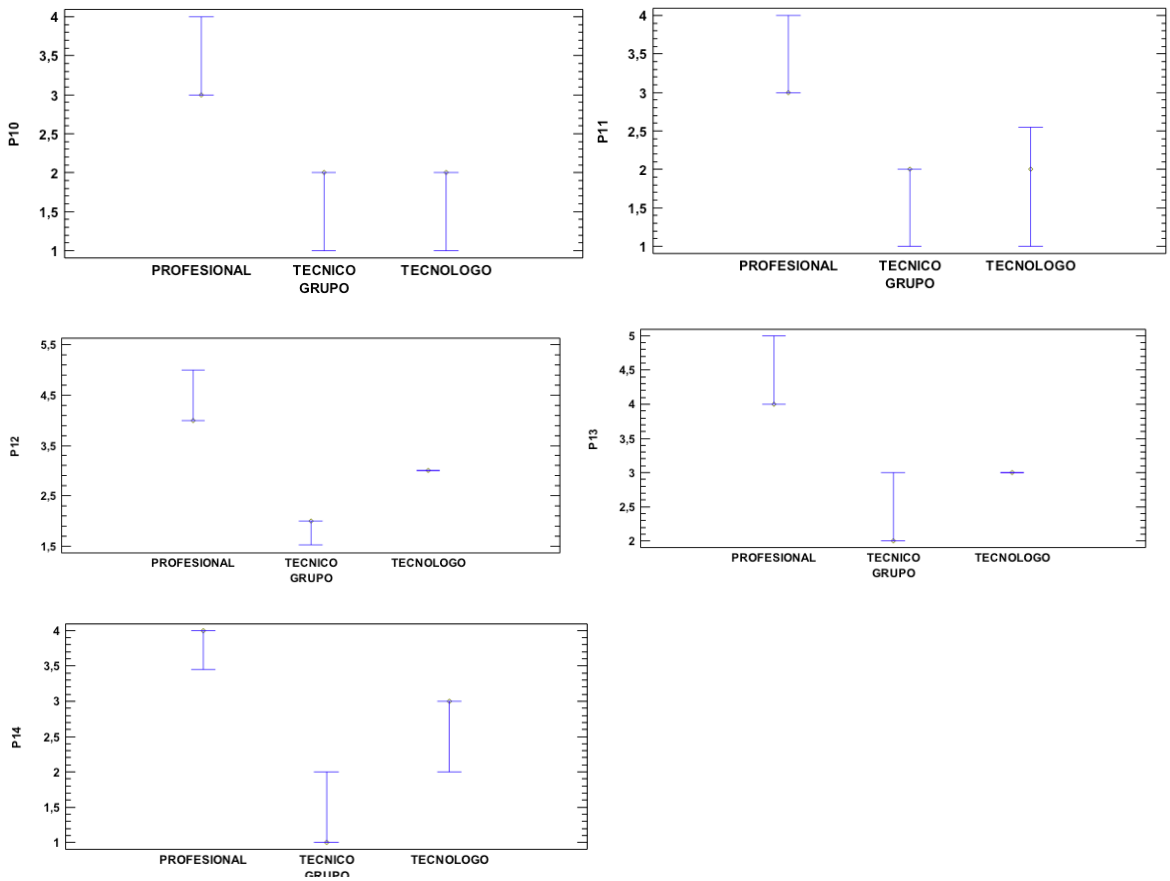
- **Categoría Administración de Tiempo PSP:**

Las gráficas de esta categoría muestran la heterogeneidad entre las medianas de todas las preguntas, que están relacionadas con el registro de tiempos de las actividades asociadas al desarrollo de software.

Las respuestas de la pregunta 14: "Registra el tiempo de las interrupciones que se le presentan al construir software?", muestran que existe una diferencia significativa entre las medianas de los ciclos propedéuticos. En la figura correspondiente a esa pregunta se visualiza la comparación entre las medianas de los 3 ciclos con respecto a las interrupciones. Existe un comportamiento similar en todas las preguntas de la categoría.



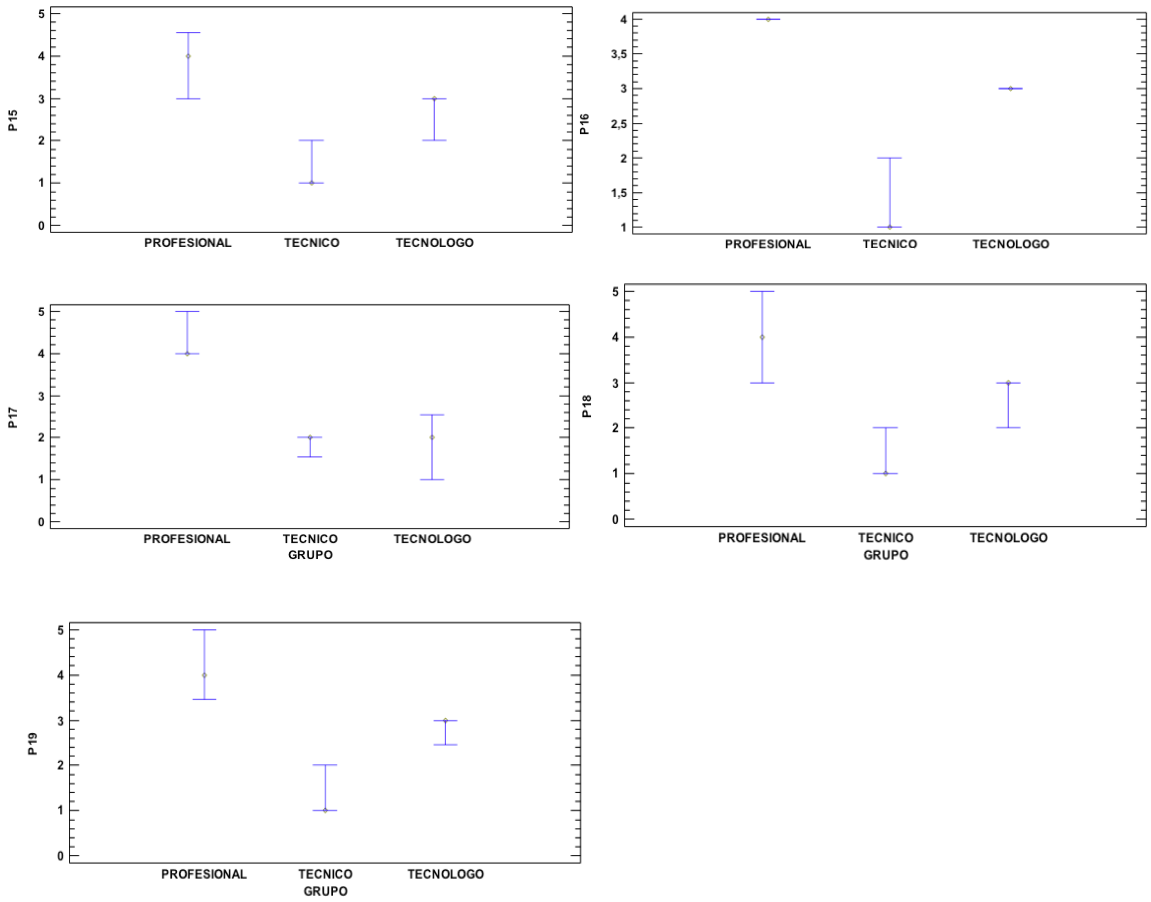




Categoría Administración de Tamaño PSP:

Se pudo encontrar que los ciclos propedéuticos presentan diferencia significativa entre las medianas relacionadas con el tamaño, tal como se puede visualizar en las gráficas siguientes, correspondientes a las preguntas 15 a 20.

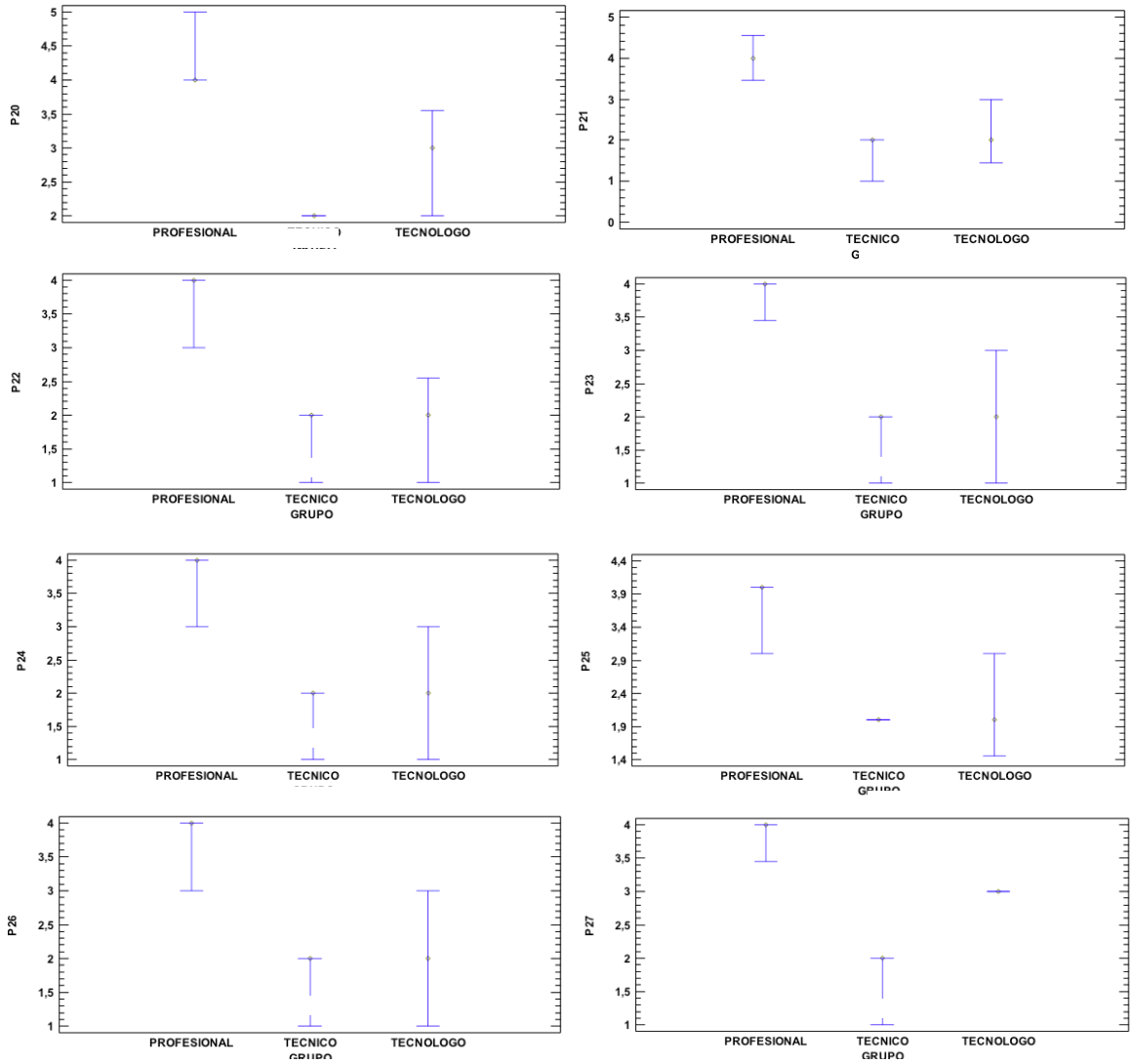
El análisis estadístico realizado en la pregunta 17: "Realiza un conteo de las líneas de código que actualiza cuando está modificando el programa o corrigiendo un defecto?", muestra que las medianas de los ciclos propedéuticos presentan diferencias estadísticamente significativas para esta práctica como se evidencia en la figura correspondiente a la pregunta citada. De igual manera pasa con el resto de preguntas de la categoría.



- **Categoría Administración de Defectos PSP:**

En las figuras que se muestran a continuación, se pueden visualizar que las medianas de las preguntas de la categoría de la administración de defectos tienen una diferencia significativa entre los ciclos propedéuticos. Las preguntas relacionadas son las que se identifican con los números del 20 al 27.

La frecuencia en que los estudiantes registran los resultados de sus pruebas, es evaluada en las respuestas de la pregunta 23: "Cuando verifica el correcto funcionamiento del software que construye, registra los resultados que obtiene?". Con medianas que difieren entre un ciclo propedéutico y otro, se visualiza en la figura asociada a la pregunta mencionada, la heterogeneidad de los resultados para la correspondiente práctica. Dicha heterogeneidad también es evidente en las otras preguntas de la categoría.

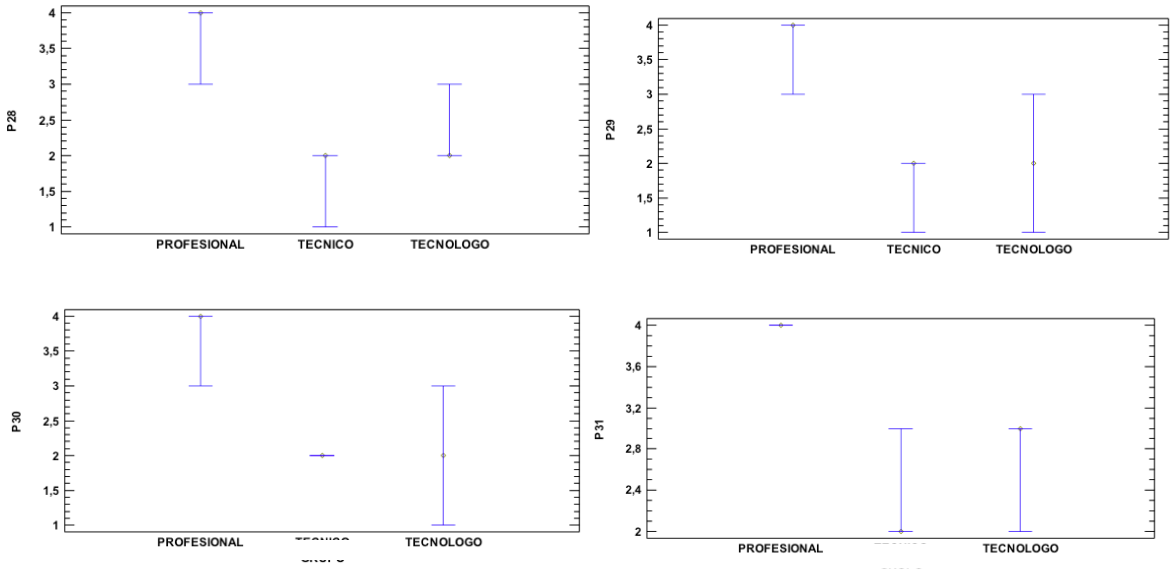


Categoría Planeación PSP:

A continuación se muestran las figuras relacionadas con las preguntas 28 a 31, todas correspondientes a prácticas de planeación PSP, donde se muestra la heterogeneidad de los resultados entre los ciclos técnico, tecnológico y universitario.

Para la pregunta 29, relacionada con la generación de cronogramas y su ejecución, la prueba No paramétrica de Kruskal-Wallis permite afirmar que no existe diferencia significativa entre las medianas de los ciclos propedéuticos. La figura correspondiente a esa pregunta, muestra los resultados de los ciclos propedéuticos, al momento de llevar a

cabo la práctica relacionada. Los resultados de las preguntas 28, 30 y 31 son similares al análisis comentado.

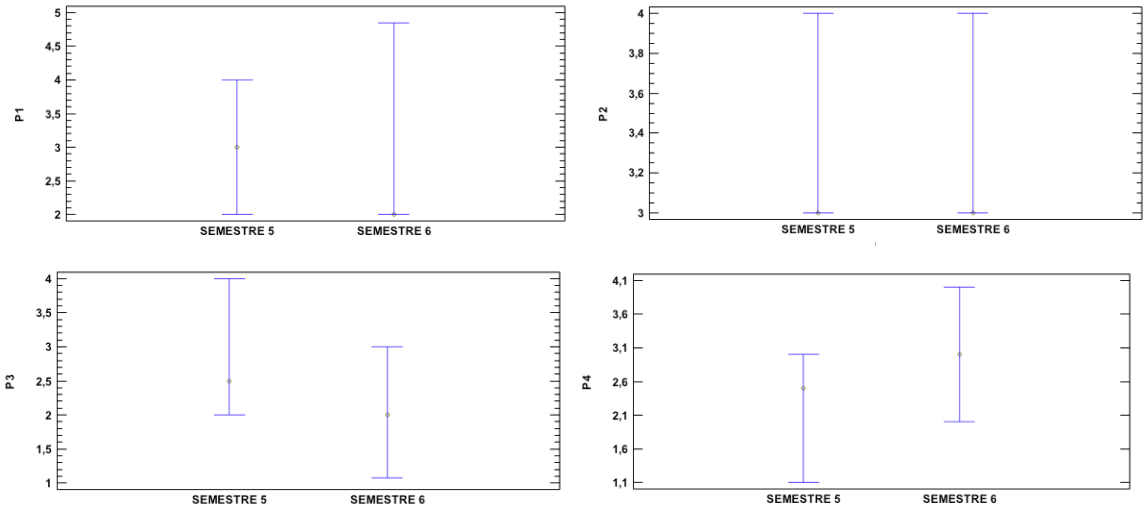


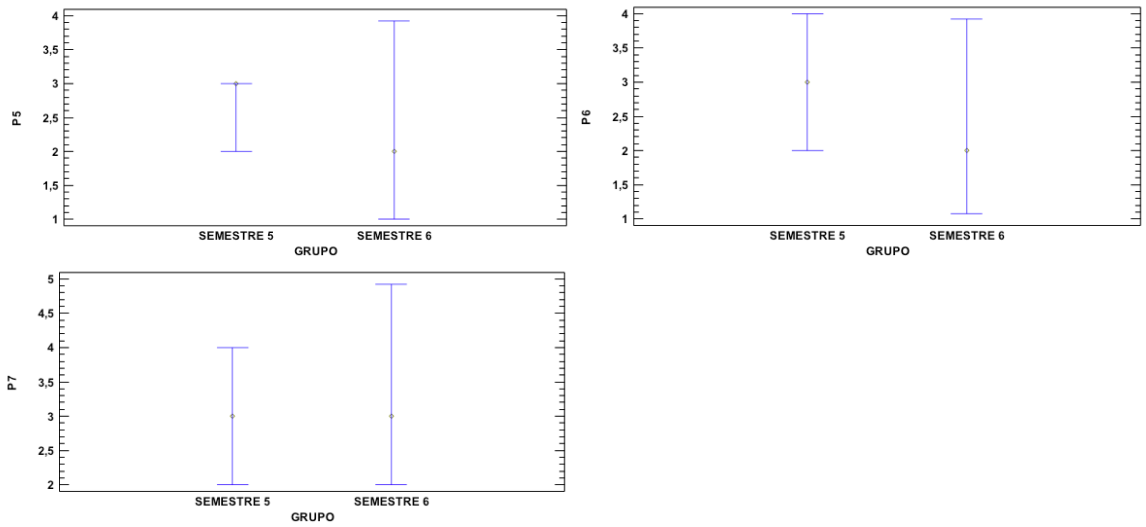
Anexo 2: Gráficos de homogeneidad del análisis estadístico de diagnóstico para los grupos experimental y control

- **Categoría Proceso PSP:**

Las preguntas 1 a la 7 están representadas en las figuras que se muestran a continuación. Todas ellas asociadas a prácticas PSP relacionadas con el proceso que realizan los estudiantes de los grupos experimental y control de la investigación. Se muestran en las figuras la homogeneidad en sus prácticas.

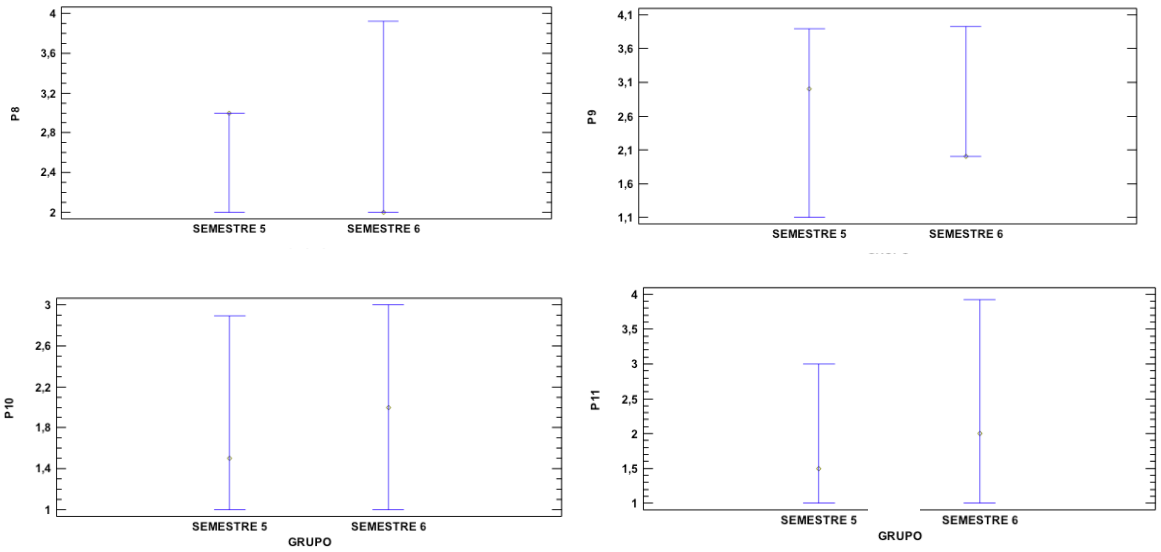
El análisis realizado que se muestra en las gráficas, permite concluir que no existe una diferencia significativa entre las medianas resultantes. Como ejemplo podemos tomar la correspondiente a la pregunta 7: "Al programar, utiliza estándares de programación (estilos o convenciones para escribir código)?". Allí se muestra la homogeneidad de los resultados asociados con la práctica correspondiente a esa pregunta. Esa homogeneidad también se presenta en las restantes preguntas de la categoría.

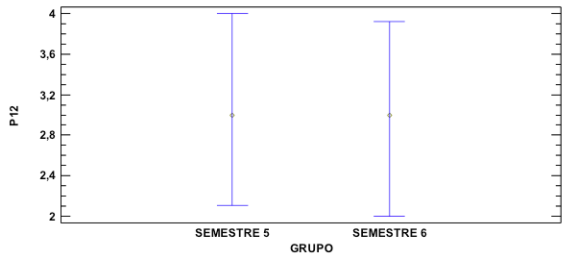


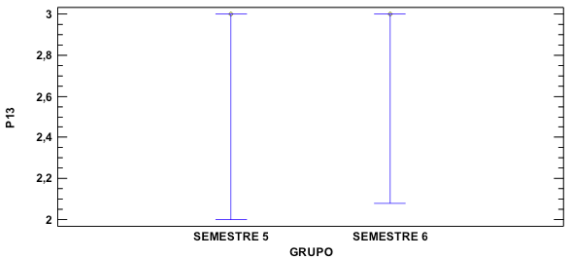


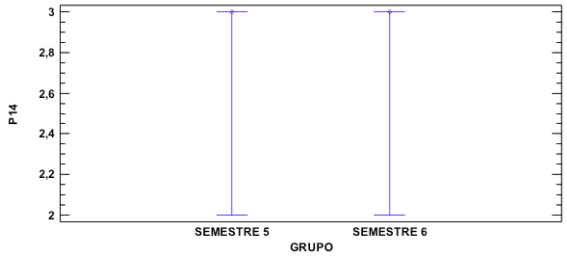
- **Categoría Administración de Tiempo PSP:**

Las prácticas relacionadas con el tiempo mostraron la homogeneidad al comparar las medianas de los grupos experimental y control. A continuación se muestran las figuras relacionadas con las preguntas 8 a 14, donde podemos tomar como ejemplo la figura que representa los resultados de la pregunta 14: "Registra el tiempo de las interrupciones que se le presentan al construir software?", que muestran que no hay diferencia estadísticamente significativa entre los resultados de las medianas correspondientes a cada grupo. Las prácticas relacionadas con todas las preguntas de esta categoría presentaron resultados similares.





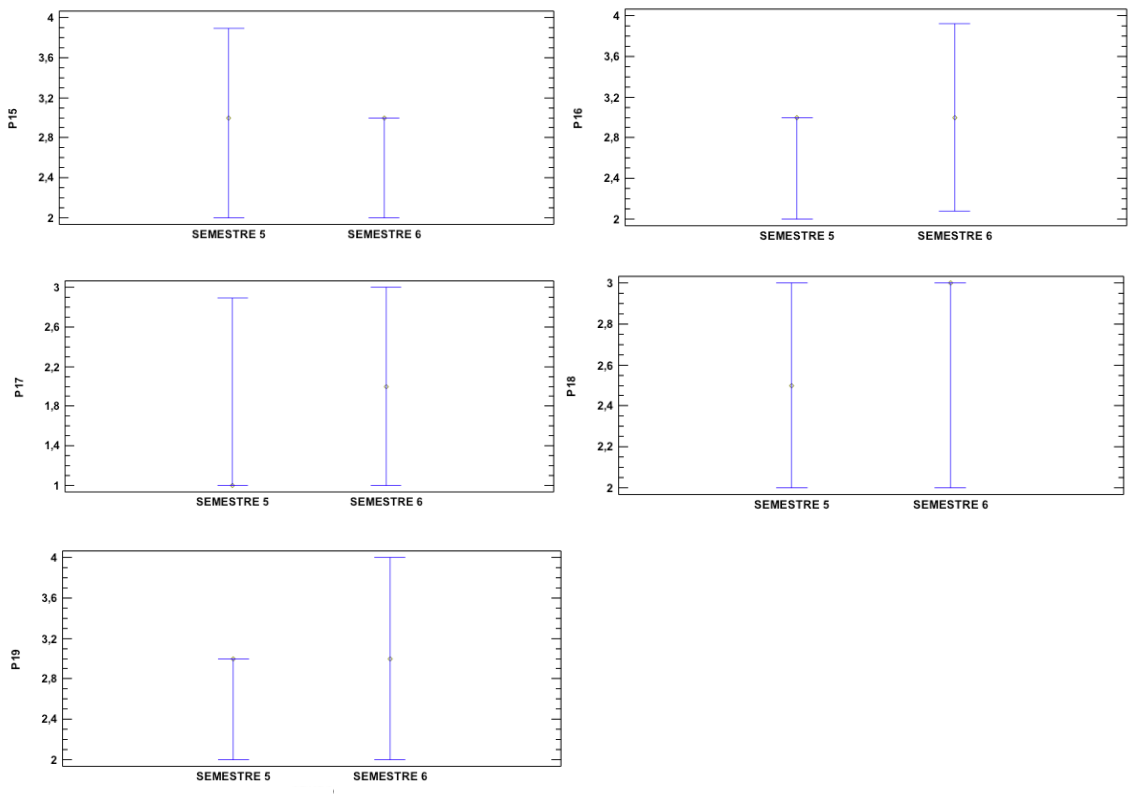




- **Categoría Administración de Tamaño PSP:**

Las preguntas 15 a 19 corresponden a la categoría de administración del tiempo y son las que se muestran a continuación. Las medianas mostraron que existe homogeneidad en las prácticas asociadas en todas las preguntas de esta categoría.

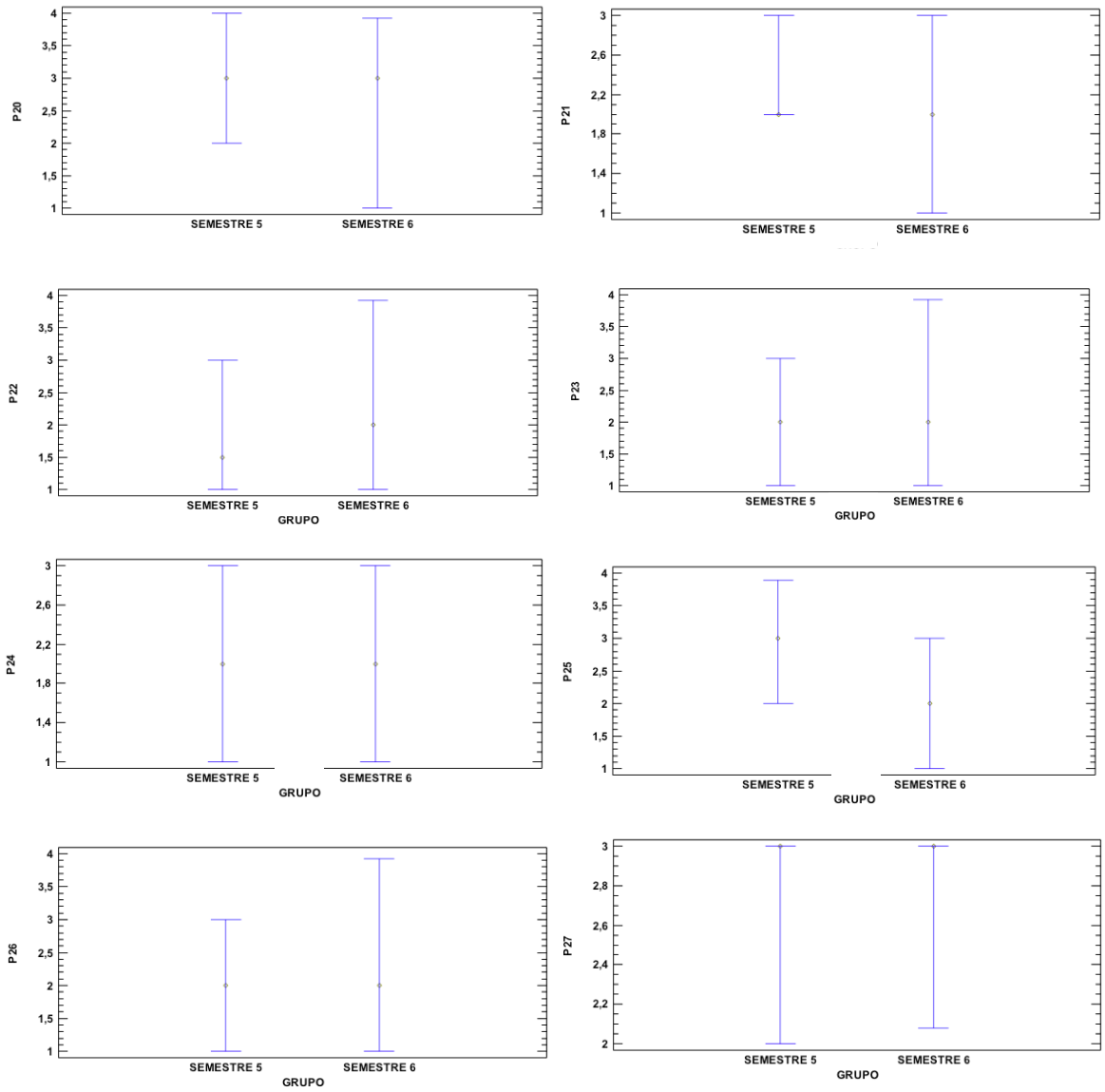
Por ejemplo, las respuestas encontradas a la pregunta 19: "Cuenta las líneas de código por hora que genera cuando está programando?", muestran que las medianas de los grupos control y experimental presentan homogeneidad entre ellos, como se visualiza en la figura correspondiente.



- **Categoría Administración de Defectos PSP:**

Los resultados correspondientes a las preguntas 20 a 27, relacionadas con prácticas de defectos, muestran homogeneidad entre las medianas de los grupos experimental y control.

Los resultados de todas las preguntas tuvieron un comportamiento similar. Se puede tomar como ejemplo la pregunta 24: "Para verificar el correcto funcionamiento del software que construye, registra los resultados que espera obtener antes de realizar una prueba?". La figura de esa pregunta, muestra la homogeneidad entre las medianas correspondientes a la práctica para ambos grupos (control y experimental).

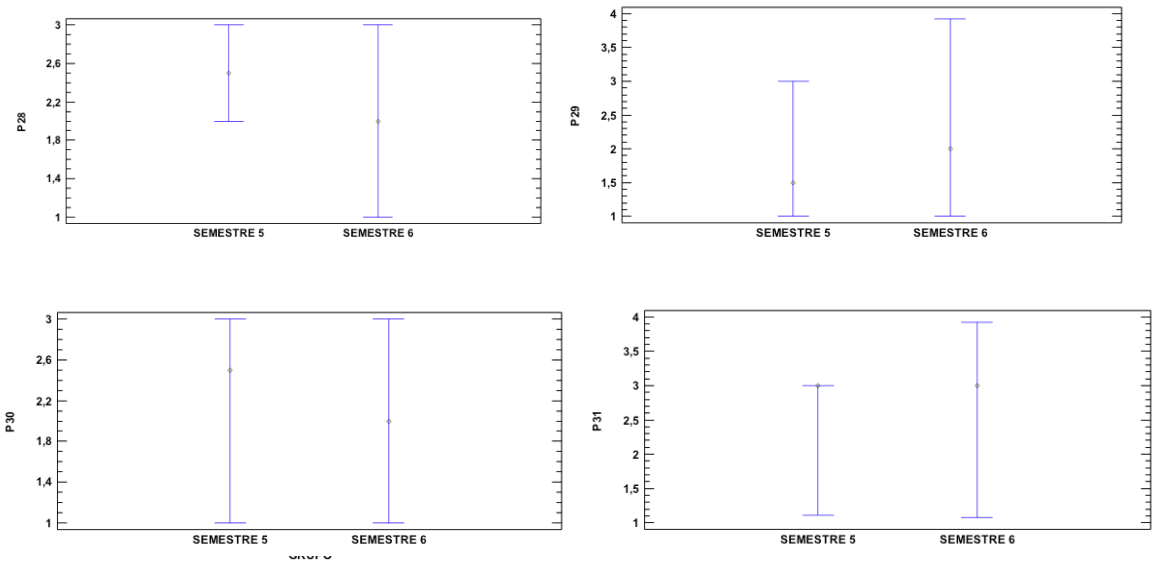


- **Categoría Planeación PSP:**

Los grupos experimental y control mostraron ser homogéneos en todas las prácticas PSP relacionadas con la planeación de sus procesos de desarrollo de software que fueron indagadas a través de la encuesta de buenas prácticas de programación. La homogeneidad de sus medianas se muestra en las figuras de las preguntas 28 a 31 que se encuentran a continuación.

Tal es el caso de la pregunta 31: "Reúne información de los programas anteriormente construidos por usted, para hacer mejoras en los siguientes proyectos que va a realizar?",

donde se muestra que la práctica es tomada de manera similar por los estudiantes de ambos grupos (experimental y control).



Anexo 3: Encuesta de buenas prácticas de programación

Esta encuesta tiene por objeto, conocer las diferentes prácticas que los estudiantes del programa de Ingeniería de Software de la Escuela de Administración y Mercadotecnia del Quindío, realizan al momento de construir software y que componen su proceso personal de desarrollo de software actualmente. La información suministrada será con propósitos exclusivamente investigativos y tendrá carácter confidencial.

Instrucciones: Registre a través de este instrumento, la forma en que habitualmente usted realiza la construcción de software, calificando las preguntas de uno (1) a cinco (5), donde:

(1) Nunca lo realiza. (2) En muy pocas ocasiones. (3) A veces. (4) Casi siempre. (5) Siempre.

En esta sección, refiera las actividades que realiza cuando ejecuta un proceso de construcción de software (Tiene relación con los pasos que realiza al programar).	**1**	**2**	**3**	**4**	**5**
1. ¿Identifica las actividades que le representan mayor dificultad al programar?					
2. Al momento de programar, ¿valida a través de ensayo y error que está obteniendo resultados deseados?					
3. ¿Sólo inicia la codificación de sus programas, después de tener completamente claro lo que debe realizar?					
4. Cuando programa, ¿suele dividir el problema principal en problemas más pequeños para facilitar su solución?					
5. ¿Realiza un diseño (diagramas y modelos) de lo que va a construir antes de iniciar su codificación?					
6. Cuando usted programa, ¿tiene en cuenta las actividades que realizan otros desarrolladores para incorporarlas dentro de sus propias actividades y obtener mejores resultados?					
7. Al programar, ¿utiliza estándares de programación (estilos o convenciones para escribir código)?					

En esta sección, refiera las características relacionadas con el tiempo que destina en su proceso de construcción de software.	**1**	**2**	**3**	**4**	**5**
8. ¿Registra el tiempo que destina para cada actividad que realiza al programar?					
9. ¿Identifica las actividades que le toman más tiempo al programar?					
10. ¿Estima el tiempo que le toma cada actividad y el tiempo total cuando programa?					
11. ¿Compara el tiempo total que le toma hacer un programa contra el tiempo que estimó que iba a tardar?					
12. ¿Generalmente el tiempo que le tomó construir un programa es superior al tiempo que creía que se iba a tardar?					
13. ¿Identifica las principales causas de interrupción (que detienen alguna actividad de su proyecto) cuando está construyendo software?					
14. ¿Registra el tiempo de las interrupciones que se le presentan al construir software?					

En esta sección, refiera las características asociadas al tamaño de los programas generados cuando ejecuta un proceso de construcción de software.	**1**	**2**	**3**	**4**	**5**
15. ¿Al finalizar la construcción de un programa, mide el tamaño del software (LOC: Lines Of Codes/Líneas de Código) que construyó?					
16. ¿Estima el tamaño (en líneas de código) que tendrán los programas de software que construye antes de empezar a programar?					
17. ¿Realiza un conteo de las líneas de código que actualiza cuando está modificando el programa o corrigiendo un defecto?					
18. ¿Estima la cantidad de líneas de código por hora que escribe cuando va a programar?					
19. ¿Cuenta las líneas de código por hora que genera cuando está programando?					

En esta sección, refiera todo lo relacionado con los errores en que incurre cuando está en un proceso de construcción de software.	**1**	**2**	**3**	**4**	**5**
20. ¿Tiene identificados los errores que más comete al programar?					
21. ¿Establece algún tipo de clasificación cuando comete errores al construir software?					
22. ¿Registra información de los errores que comete al realizar un programa (su tipo, su causa, el lugar donde lo cometió, etc.)?					
23. ¿Cuando verifica el correcto funcionamiento del software que construye, registra los resultados que obtiene?					
24. ¿Para verificar el correcto funcionamiento del software que construye, registra los resultados que espera obtener antes de realizar una prueba?					
25. Al programar, ¿identifica en qué actividad de la construcción de software (levantamiento de requerimientos, análisis, diseño, codificación, pruebas u otros) comete más errores?					
26. ¿Aplica algún método para encontrar y corregir errores cuando construye software?					
27. ¿Al terminar de programar cuenta la cantidad de errores que cometió?					

En esta sección, refiera las características relacionadas con la planeación de las actividades que realiza cuando está en un proceso de construcción de software.	**1**	**2**	**3**	**4**	**5**
28. ¿Establece las actividades (levantamiento de requerimientos, análisis, diseño, codificación, pruebas u otros) que va a realizar cuando programa?					
29. ¿Construye un cronograma y lo ejecuta cuando está programando?					
30. ¿Planea la búsqueda de posibles errores en sus programas?					
31. ¿Reúne información de los programas anteriormente construidos por usted, para hacer mejoras en los siguientes proyectos que va a realizar?					

Anexo 4: Rúbricas

- Rúbrica de fase de Direccionamiento: Introducción a la calidad del software

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Concepto de calidad en el desarrollo de software	Comprende de manera satisfactoria el concepto de la calidad en el software y lo sabe ejemplificar y contextualizar claramente en su entorno.	Comprende el concepto de la calidad en el software y lo sabe ejemplificar y contextualizar claramente en su entorno.	Tienen inconvenientes definiendo el concepto de calidad en el software y le es difícil dar un ejemplo y contextualizarlo.	No comprende el concepto de la calidad en el software y tiene dificultades para ejemplificar y contextualizar el concepto.
Evaluación del impacto de los errores en la calidad del software	Entiende plenamente como los errores en el desarrollo de software impactan en el producto final y en los objetivos para los cuales fue construido.	Entiende que los errores en el desarrollo de software impactan en el producto final y en los objetivos para los cuales fue construido.	Tiene dificultades para entender que los errores en el desarrollo de software impactan en el producto final y en los objetivos para los cuales fue construido.	No entiende que los errores en el desarrollo de software impactan en el producto final y en los objetivos para los cuales fue construido.
Comprensión de la importancia de la calidad en el software	Valora ampliamente la importancia que tiene la calidad en el desarrollo de software y sus consecuencias.	Valora la importancia que tiene la calidad en el desarrollo de software y sus consecuencias.	Valora parcialmente la importancia que tiene la calidad en el desarrollo de software y sus consecuencias.	No valora la importancia que tiene la calidad en el desarrollo de software y sus consecuencias.

- Rúbrica de fase de Planeación: Proceso personal de desarrollo de software

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Concepto de proceso	Entiende plenamente el concepto de Proceso y está en capacidad de establecer de manera satisfactoria, un proceso para diversos contextos.	Entiende el concepto de Proceso y está en capacidad de establecer uno para diversos contextos.	Tiene dificultades para entender el concepto de proceso y para establecer procesos para diversos contextos.	No entiende el concepto de Proceso y no está en capacidad de establecer uno para diversos contextos.
Ventajas de un proceso de software	Identifica fácilmente y de manera satisfactoria las ventajas que tiene el construir software bajo un proceso definido.	Identifica las ventajas que tiene el construir software bajo un proceso definido.	No le son plenamente claras las ventajas que tiene el construir software bajo un proceso definido.	No reconoce las ventajas que tiene el construir software bajo un proceso definido.

Incorporación del concepto de calidad en un proceso	Destaca de manera amplia la importancia de la calidad y lo incorpora con facilidad en un proceso que deba establecer.	Está en capacidad de incorporar la calidad en un proceso que daba establecer.	Presenta dificultad al incorporar el concepto de calidad en un proceso que deba establecer	No destaca la importancia de introducir atributos de calidad en un proceso que deba establecer.
Métrica	Comprende de manera satisfactoria el concepto de las métricas, su uso y propósito.	Comprende el concepto de métrica, su uso y propósito.	Presenta dificultades para comprender las métricas, su uso y su propósito.	No comprende la definición de métrica, su uso ni su propósito.

- Rúbricas de fase de Actuación-Ejecución:

1. Formato de registro de tiempos para todas las etapas que evidencien gestión del tiempo.

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se entrega al profesor el formato de registro de tiempos para todas las etapas que evidencien gestión del tiempo	Se diligencian todos los datos de cada registro hecho. Se evidencia el registro de tiempos en todas las fases de desarrollo (Planeación, Desarrollo<Diseño, Codificación, Pruebas> y Post-Mortem). Es consistente la hora de inicio con la hora de fin de cada uno de los registros. Registra tiempos de interrupción y estos son coherentes con respecto al tiempo total. Hace comentarios describiendo las razones de sus interrupciones u otras observaciones de valor para el proceso.	Se diligencian todos los datos de cada registro hecho. Se evidencia el registro de tiempos en todas las fases de desarrollo (Planeación, Desarrollo<Diseño, Codificación, Pruebas> y Post-Mortem). Es consistente la hora de inicio con la hora de fin de cada uno de los registros. NO registra tiempos de interrupción y estos son coherentes con respecto al tiempo total. NO hace comentarios describiendo las razones de sus interrupciones u otras observaciones de valor para el proceso.	Se diligencian todos los datos de cada registro hecho. Se evidencia el registro de tiempos en todas las fases de desarrollo (Planeación, Desarrollo<Diseño, Codificación, Pruebas> y Post-Mortem). NO es consistente la hora de inicio con la hora de fin de cada uno de los registros. NO registra tiempos de interrupción y estos son coherentes con respecto al tiempo total. NO hace comentarios describiendo las razones de sus interrupciones u otras observaciones de valor para el proceso.	NO se diligencian todos los datos de cada registro hecho. NO se evidencia el registro de tiempos en todas las fases de desarrollo (Planeación, Desarrollo<Diseño, Codificación, Pruebas> y Post-Mortem). NO es consistente la hora de inicio con la hora de fin de cada uno de los registros. NO registra tiempos de interrupción y estos son coherentes con respecto al tiempo total. NO hace comentarios describiendo las razones de sus interrupciones u otras observaciones de valor para el proceso.

2. Formato de registro de defectos que evidencien gestión de los defectos.

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se entrega al profesor el formato de registro de defectos que evidencien gestión de los defectos	Se diligencian todos los datos de cada registro hecho. Utiliza correctamente el identificador del defecto. El tipo corresponde a uno existente en el estándar de tipos de defectos. Son consistentes la fase de inyección con la fase de eliminación registradas. Son consistentes el tipo especificado con la descripción del defecto registrada.	Se diligencian todos los datos de cada registro hecho. Utiliza correctamente el identificador del defecto. El tipo NO corresponde a uno existente en el estándar de tipos de defectos. Son consistentes la fase de inyección con la fase de eliminación registradas. Son consistentes el tipo especificado con la descripción del defecto registrada.	Se diligencian todos los datos de cada registro hecho. Utiliza correctamente el identificador del defecto. El tipo NO corresponde a uno existente en el estándar de tipos de defectos. NO son consistentes la fase de inyección con la fase de eliminación registradas. NO son consistentes el tipo especificado con la descripción del defecto registrada.	NO diligencian todos los datos de cada registro hecho. NO utiliza correctamente el identificador del defecto. El tipo NO corresponde a uno existente en el estándar de tipos de defectos. NO son consistentes la fase de inyección con la fase de eliminación registradas. NO son consistentes el tipo especificado con la descripción del defecto registrada.

3. Código fuente del programa conforme al estándar de codificación Java entregado.

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se presenta al profesor en un archivo comprimido (.zip), el código fuente del programa conforme al estándar de codificación Java entregado	Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata la generación de los nombres de los archivos y elementos como paquetes, clases e interfaces, métodos, variables y constantes. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata sobre la indentación. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de los comentarios. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de declaraciones. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de sentencias. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de espacios en blanco.	Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata la generación de los nombres de los archivos y elementos como paquetes, clases e interfaces, métodos, variables y constantes. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata sobre la indentación. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de los comentarios. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de declaraciones. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de sentencias. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de espacios en blanco.	Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata la generación de los nombres de los archivos y elementos como paquetes, clases e interfaces, métodos, variables y constantes. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata sobre la indentación. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado de comentarios. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de declaraciones. Su código fuente cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de sentencias. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de espacios en blanco.	Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata la generación de los nombres de los archivos y elementos como paquetes, clases e interfaces, métodos, variables y constantes. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata sobre la indentación. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de los comentarios. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de declaraciones. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de sentencias. Su código fuente NO cumple con lo establecido en el estándar de codificación en el apartado que trata acerca de espacios en blanco.

4. Formato de registro de pruebas diligenciado donde se evidencien las pruebas planeadas y ejecutadas.

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se entrega al profesor el formato de registro de pruebas diligenciado donde se evidencien las pruebas planeadas y ejecutadas	Es claro el objetivo de la prueba. La descripción de la prueba da una idea integral de lo que se desea verificar. Las condiciones de la prueba son consistentes y claras. Se registra de manera adecuada los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.	Es claro el objetivo de la prueba. La descripción de la prueba da una idea integral de lo que se desea verificar. Las condiciones de la prueba son consistentes y claras. Se registra con dificultades los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.	Es claro el objetivo de la prueba. La descripción de la prueba da una idea integral de lo que se desea verificar. Las condiciones de la prueba NO son consistentes ni claras. Se registra con dificultades los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.	NO es claro el objetivo de la prueba. La descripción de la prueba NO da una idea integral de lo que se desea verificar. Las condiciones de la prueba NO son consistentes ni claras. Se registra con dificultades los resultados esperados y obtenidos de la prueba y son consecuentes con el objetivo de la misma.

5. Formato de Propuesta de Mejora de Proceso, donde identifique problemas encontrados y las propuestas de solución para próximas construcciones de programas.

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se entrega al profesor el formato de Propuesta de Mejora de Proceso, donde identifique problemas encontrados y las propuestas de solución para próximas construcciones de programas	La propuesta de mejora describe de manera clara el problema a resolver o mejorar. La propuesta es pertinente y alineada al problema correspondiente registrado. Agrega notas y comentarios de gran valor para la mejora del proceso.	La propuesta de mejora describe de manera clara el problema a resolver o mejorar. La propuesta es pertinente y alineada al problema correspondiente registrado. Las notas y comentarios no son lo suficientemente claras o no agregan valor para la mejora del proceso.	La propuesta de mejora describe de manera clara el problema a resolver o mejorar. La propuesta NO es pertinente y/o NO se encuentra alineada al problema correspondiente registrado. Las notas y comentarios no son lo suficientemente claras o no agregan valor para la mejora del proceso.	Tiene dificultades para identificar y/o describir claramente el problema a resolver o mejorar. La propuesta NO es pertinente y/o NO se encuentra alineada al problema correspondiente registrado. Las notas y comentarios no son lo suficientemente claras o no agregan valor para la mejora del proceso.

6. Formato de Resumen de Plan de Proyecto, donde se evidencie tiempo planeado inicialmente, tiempos por fase, defectos inyectados y eliminados por fase y en general la información correspondiente del formato.

Indicador	Estratégico	Autónomo	Resolutivo	Receptivo
Se entrega al profesor el formato de Resumen de Plan de Proyecto, donde se evidencie tiempo planeado inicialmente, tiempos por fase, defectos inyectados y eliminados por fase y en general la información correspondiente del formato	Los datos registrados de tiempo por fase del proyecto, tiempo por fase a la fecha (por todos los proyectos) y porcentaje de tiempo por fase a la fecha son consistentes y coherentes. Los datos registrados de defectos introducidos por fase del proyecto, defectos introducidos por fase a la fecha (por todos los proyectos) y porcentaje de defectos introducidos por fase a la fecha son consistentes y coherentes. Los datos registrados de defectos eliminados por fase del proyecto, defectos eliminados por fase a la fecha (por todos los proyectos) y porcentaje de defectos eliminados por fase a la fecha son consistentes y coherentes.	Los datos registrados de tiempo por fase del proyecto, tiempo por fase a la fecha (por todos los proyectos) y porcentaje de tiempo por fase a la fecha son consistentes y coherentes. Los datos registrados de defectos introducidos por fase del proyecto, defectos introducidos por fase a la fecha (por todos los proyectos) y porcentaje de defectos introducidos por fase a la fecha son consistentes y coherentes. Los datos registrados de defectos eliminados por fase del proyecto, defectos eliminados por fase a la fecha (por todos los proyectos) y porcentaje de defectos eliminados por fase a la fecha NO son consistentes ni coherentes, mostrando dificultades en su entendimiento.	Los datos registrados de tiempo por fase del proyecto, tiempo por fase a la fecha (por todos los proyectos) y porcentaje de tiempo por fase a la fecha son consistentes y coherentes. Los datos registrados de defectos introducidos por fase del proyecto, defectos introducidos por fase a la fecha (por todos los proyectos) y porcentaje de defectos introducidos por fase a la fecha NO son consistentes ni coherentes, mostrando dificultades en su entendimiento. Los datos registrados de defectos eliminados por fase del proyecto, defectos eliminados por fase a la fecha (por todos los proyectos) y porcentaje de defectos eliminados por fase a la fecha NO son consistentes ni coherentes, mostrando dificultades en su entendimiento.	Los datos registrados de tiempo por fase del proyecto, tiempo por fase a la fecha (por todos los proyectos) y porcentaje de tiempo por fase a la fecha NO son consistentes ni coherentes, mostrando dificultades en su entendimiento. Los datos registrados de defectos introducidos por fase del proyecto, defectos introducidos por fase a la fecha (por todos los proyectos) y porcentaje de defectos introducidos por fase a la fecha NO son consistentes ni coherentes, mostrando dificultades en su entendimiento. Los datos registrados de defectos eliminados por fase del proyecto, defectos eliminados por fase a la fecha (por todos los proyectos) y porcentaje de defectos eliminados por fase a la fecha son consistentes y coherentes, mostrando dificultades en su entendimiento.

Anexo 5: Guía instruccional del curso

EAM
Escuela de Administración y Mercadotecnia del Quindío

ESTRATEGIA DE ENSEÑANZA PARA LA APROPIACIÓN DE PRÁCTICAS PSP

GUÍA INSTRUCCIONAL DEL CURSO

Institución	Escuela de Administración y Mercadotecnia del Quindío - EAM
Nombre Curso Intervenido	Programación Avanzada I
Fecha y hora	22 de septiembre de 2015 – 10:00 am
Número Guía	5
Temática Programación	Hilos
Temática PSP	Fase de Post-Mortem – Formato de resumen de plan de proyecto
Intensidad Horaria	2 horas

Objetivos PSP

- Conocer el objetivo de la fase de Post-Mortem con el fin de promover en los estudiantes su ejecución en cada proceso de desarrollo de software que emprendan.
- Identificar las actividades básicas que se deben realizar en la etapa de Post-Mortem.
- Conocer el formato de Resumen de Plan de Proyecto, entender su uso y fin principal.
- Promover en los estudiantes el uso del formato de Resumen de Plan de Proyecto.

Metodología

- A través de técnica expositiva, el profesor expone la fase de Post-Mortem que se realizar en el nivel PSP0, sus actividades y de manera adicional el formato de Resumen de Plan de Proyecto.

Actividades

- Haciendo una exposición verbal, el profesor explica en qué consiste la fase de Post-Mortem y las ventajas que se obtienen al ejecutar esta etapa cuando finaliza el proceso, destacándola como una buena práctica que permite hacer una revisión del proceso realizado.
- Teniendo como base el script de la fase de Post-Mortem de PSP0, el profesor listará los instrumentos requeridos para iniciar la fase (criterios de entrada), las actividades que se realizan en la misma y los productos generados a partir de esta (criterios de salida).
- Se socializará el formulario de Resumen de Plan de Proyecto, su estructura e instrucciones de llenado. Es importante que se expongan sus ventajas para que los estudiantes se vean motivados a utilizarlo.
- Se expondrán el taller que los estudiantes deben realizar y la rúbrica que permitirá posteriormente, retroalimentarlos con los resultados obtenidos.

Material entregado al profesor

- Guía Instruccional del curso – Fase de Post-Mortem y Resumen de Plan de Proyecto.
- Material teórico de la sesión.
- Formato de Resumen de Plan de Proyecto
- Taller que establece los entregables que deben generar los estudiantes a partir de la temática tratada.
- Rúbrica de Fase de Post-Mortem y Resumen de Plan de Proyecto.

Anexo 6: Elementos de recolección de información

- Formato de Registro de tiempos

EAM
Escuela de Administración y Mercadotecnia del Quindío

Institución	Escuela de Administración y Mercadotecnia del Quindío EAM
Programa	Ingeniería de Software
Semestre	
Curso	Programación avanzada I
Nombre Estudiante	
Identificador del programa	
Nombre formato	Formato de registro de Tiempos

FASE	Fecha y hora inicio (dd/mm/aa - 00:00 am/pm)	Tiempo de Interrupción (minutos)	Fecha y hora finalización (dd/mm/aa - 00:00 am/pm)	Tiempo efectivo (minutos)	Comentarios

- Formato de registro de defectos

EAM
Escuela de Administración y Mercadotecnia del Quindío

Institución	Escuela de Administración y Mercadotecnia del Quindío EAM
Programa	Ingeniería de Software
Semestre	6
Curso	Programación avanzada I
Nombre Estudiante	
Identificador del programa	1
Nombre formato	Formato de registro de defectos

Fecha (dd/mm/aa)	Identificador	Tipo	Fase Inyección	Fase detección-eliminación	Defecto Inyección
Descripción:					
Descripción:					
Descripción:					

- Propuesta de Mejora de Proceso PIP

EAM
Escuela de Administración y Mercadotecnia del Quindío

Institución	Escuela de Administración y Mercadotecnia del Quindío EAM
Programa	Ingeniería de Software
Semestre	6
Curso	Programación avanzada I
Nombre Estudiante	
Identificador del programa	
Nombre formato	Propuesta de Mejora de Proceso PIP

Número PIP	Descripción del problema

Propuesta PIP #	Descripción de la propuesta

Notas y comentarios

- Formato de reporte de pruebas

EAM
Escuela de Administración y Mercadotecnia del Quindío

Institución	Escuela de Administración y Mercadotecnia del Quindío EAM
Programa	Ingeniería de Software
Semestre	6
Curso	Programación avanzada I
Nombre Estudiante	
Identificador del programa	
Nombre formato	Reporte de pruebas
Identificador del Programa:	

Número/Nombre prueba	
Objetivo de la prueba	
Descripción de la prueba	
Condiciones de la prueba	
Resultados esperados	
Resultados obtenidos	

- Formato de resumen de plan de proyecto

EAM
Escuela de Administración y Mercadotecnia del Quindío

Institución	Escuela de Administración y Mercadotecnia del Quindío EAM
Programa	Ingeniería de Software
Semestre	6
Curso	Programación avanzada I
Nombre Estudiante	
Identificador del programa	
Nombre formato	Resumen del plan de proyecto

Tiempo en fase (min)	Plan	Real	A la fecha	% A la Fecha
Planeación				
Diseño				
Codificación				
Prueba				
Post-mortem				
Total				

Defectos introducidos	Real	A la fecha	% A la Fecha
Planeación			
Diseño			
Codificación			
Prueba			
Post-mortem			
Total			

Defectos eliminados	Real	A la fecha	% A la Fecha
Planeación			
Diseño			
Codificación			
Prueba			
Post-mortem			
Total			

Anexo 7: Examen de conocimientos en PSP0

ESTRATEGIA DE ENSEÑANZA PARA LA APROPIACIÓN DE PRÁCTICAS PSP

EVALUACIÓN DE CONOCIMIENTOS PSP

Institución	Escuela de Administración y Mercadotecnia del Quindío - EAM
Nombre Curso	
Fecha y hora	
Nombre estudiante	
Nota	

El presente cuestionario tiene por objeto evaluar los conocimientos relacionados con las prácticas para el Proceso Personal de desarrollo de Software PSP.

Todas las preguntas son de tipo Selección Múltiple y tienen una única respuesta correcta.

PREGUNTAS RELACIONADAS CON EL PROCESO EN PSP

1. El proceso personal de desarrollo de software (PSP) permite mejorar el desempeño de los programadores, usando como elementos:
    a. Planeación, Diseño, Codificación, Pruebas, Compilación y Post-Mortem.
    b. Métricas, estándares, guiones y formatos.
    c. Estándar de codificación y estándar de defectos.
    d. Registro de defectos y registro de tiempos.

2. Es una métricas en PSP :
    a. Cantidad de programadores
    b. Tiempo por fase
    c. Post-mortem

d. Número de fases de PSP

3. Las fases de PSP0 son en su orden:
    a. Planeación, Diseño, Codificación, Pruebas y Análisis.
    b. Diseño, Codificación y Pruebas.
    c. Planeación, Diseño, Codificación, Compilación, Pruebas y Post-mortem.
    d. Planeación, Codificación y Post-mortem.

4. La fase de Post-Mortem permite:
    a. Encontrar los defectos inyectados en el proceso personal de desarrollo de software realizado.
    b. Corregir los defectos inyectados.
    c. Realizar una revisión de todos los registros realizados y generar conclusiones que permitan establecer mejoras para los siguientes programas.
    d. Planear todo el proceso de desarrollo de software.

PREGUNTAS RELACIONADAS CON EL TIEMPO EN PSP

5. El registro de tiempos en PSP se realiza sobre las fases de:
    a. Planeación, Codificación y Post-Mortem
    b. Planeación, Diseño, Codificación, Compilación, Pruebas y Post-Mortem
    c. Codificación
    d. Todas las fases PSP, excepto en Post-Mortem

6. Las métricas básicas de PSP son
    a. Tiempo, Defectos, Tamaño y Calendario
    b. Productividad, defectos inyectados y defectos corregidos.
    c. Tiempos por fase y tiempo total.
    d. Tamaño del software y Cantidad de interrupciones.

7. Registrar las interrupciones que se tienen en las actividades del Proceso Personal de Desarrollo de Software sirve para:
    a. Determinar en qué fases un programador presenta más dificultades para dedicarse a su labor.
    b. Determinar los tipos de interrupciones que se presentaron durante el proceso de desarrollo del software.

c. Mejorar el proceso para el siguiente desarrollo, basado en las descripciones de las interrupciones registradas.
d. Todas las anteriores.

8. Un programador se encuentra en la fase de Pruebas, y encuentra un defecto que debe corregir. Al hacerlo, toma 5 horas de su tiempo para eliminarlo. El registro de tiempo destinado para la corrección de ese defecto, lo debe registrar en:
    a. Fase de Codificación
    b. Fase de Corrección de Defectos
    c. Fase de Pruebas
    d. Fase de Post-Mortem.

PREGUNTAS RELACIONADAS CON EL TAMAÑO EN PSP

9. Es una medida válida en PSP para determinar el tamaño del software que se construye:
    a. La cantidad de líneas de código (LOC: Lines Of Code) del programa.
    b. La cantidad de ciclos (for, while, Do_while) del programa.
    c. La cantidad de variables del programa.
    d. El número de páginas de la especificación del programa.

10. Medir la cantidad de líneas de código (LOC: Lines Of Code) construidas en un desarrollo de software, me permite:
    a. Conocer la productividad del programador, al determinar cuántas líneas se construyen por unidad de tiempo, por ejemplo, Líneas de código construidas por hora.
    b. Determinar en qué fases del desarrollo se tienen más inconvenientes dentro del proceso.
    c. Disminuir la cantidad de defectos inyectados en el siguiente desarrollo.
    d. Encontrar los defectos más fácilmente.

11. No es una ventaja de medir el tamaño del software:
    a. Estimar la cantidad de personas que se pueden requerir para desarrollar el programa.
    b. Permitir estimar el tiempo total que se puede tener de todo el desarrollo.
    c. Permitir encontrar más fácilmente los defectos
    d. Cotizar la cantidad de dinero que cuesta construir el programa.

PREGUNTAS RELACIONADAS CON LOS DEFECTOS EN PSP

12. ¿Cuál es el objetivo del estándar de defectos?
    a. Determinar en qué fases del desarrollo se encuentran más defectos.
    b. Permitir la reducción la cantidad de defectos que se pudieran inyectar en un desarrollo.
    c. Establecer el conjunto de pasos necesarios para eliminar un defecto.
    d. Establecer los diferentes tipos de defectos que se pueden inyectar y eliminar.

13. ¿En cuales fases de PSP es posible inyectar defectos?
    a. Planeación y Pruebas
    b. Diseño y Codificación
    c. A y B son correctas
    d. Ninguna de las anteriores.

14. ¿Cuál de las siguientes sentencias es falsa?
    a. Un defecto inyectado en fase de Codificación puede ser corregido en cualquier fase posterior.
    b. Todos los defectos inyectados en fase de Codificación deben ser corregidos en fase de Pruebas.
    c. Registrar la información asociada con los defectos, permite plantear propuestas de mejora para ciclos de desarrollo posteriores que lleven a evitar cometerlos nuevamente.
    d. Entre más tarde se encuentran los defectos, más costoso es corregirlos.

PREGUNTAS RELACIONADAS CON LA PLANEACIÓN EN PSP

15. Es una de las precondiciones para iniciar la fase de Planeación de PSP:
    a. El programa ya probado y ejecutado.
    b. Estándar de tipo de defectos
    c. Descripción del problema que se va a resolver.
    d. Guión de desarrollo

16. En PSP, la fase de Planeación permite:
    a. Realizar el diseño de la solución que se pretende construir.
    b. Obtener la declaración de requisitos del programa y estimar el tiempo que se tardará la construcción de la solución.
    c. Corregir defectos de programas construidos anteriormente.
    d. Ninguna de las anteriores.