APLICACIÓN DE TÉCNICAS DE APRENDIZAJE COOPERATIVO PARA LA ADOPCIÓN DE PRÁCTICAS DE PSP Y TSP EN UN CURSO DE PROGRAMACIÓN DE COMPUTADORES

JORGE ORLANDO HERRERA MORALES

UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS
MAESTRÍA EN INGENIERÍA
DICIEMBRE 2016

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE COOPERATIVO PARA LA ADOPCIÓN DE PRÁCTICAS DE PSP Y TSP EN UN CURSO DE PROGRAMACIÓN DE COMPUTADORES

JORGE ORLANDO HERRERA MORALES

Tesis para acceder al título de Magister en Ingeniería

Asesor
SERGIO AUGUSTO CARDONA TORRES

UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS
MAESTRÍA EN INGENIERÍA
DICIEMBRE 2016

NOTA DE ACEPTACION

_____

_____

_____

AGRADECIMIENTOS

Agradezco infinitamente a mi asesor Sergio Augusto Cardona, ya que sin su colaboración hubiese sido imposible la elaboración de este proyecto.

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

Durante las últimas décadas la sociedad ha sufrido constantes cambios; más aún, con la aparición de las tecnologías de la Información y la Comunicación, se han dado variaciones significativas en todas las actividades del quehacer humano. La educación no ha sido ajena a estos cambios. En la actualidad, con la posibilidad que tienen los estudiantes de acceder al conocimiento a través de distintas fuentes, se hace necesario diseñar estrategias formativas que desarrollen diversos tipos de habilidades durante su formación profesional. Dentro de estas estrategias se encuentra el desarrollo de actividades cooperativas, a través de las cuales se puedan crear escenarios educativos para que los estudiantes apropien, apliquen y alcancen los objetivos de formación.

El Aprendizaje Cooperativo permite desarrollar actitudes y capacidades de trabajo en equipo en los estudiantes, preparándolos para su futuro desenvolvimiento profesional, donde, igualmente deberán, ya como trabajadores, establecer equipos de trabajo que muchas veces estarán conformados por personas pertenecientes a diversas culturas, clases sociales, géneros, etc. y así producir resultados superiores, en términos de calidad y tiempo, a los que se producirían si se trabajara individualmente. El trabajo realizado en equipo, es un trabajo sinérgico, el cual gestionado correctamente potencia los resultados de cada uno de sus integrantes en términos de tiempo y productividad (Linares, J. E, 2001), permitiendo encontrar mejores alternativas y resultados a los problemas que se afrontan, haciendo los equipos más productivos gracias a la interacción entre personas al compartir conocimientos y sobre todo, a factores motivacionales asociados la consecución de objetivos comunes.

Por otro lado, en la actualidad la construcción de aplicaciones informáticas debe responder a requerimientos muy complejos y de carácter crítico de las organizaciones. La complejidad inmersa en los proyectos de desarrollo de software está asociada a múltiples fuentes: metodologías utilizadas, tecnologías de apoyo, capacidades de las personas, productividad de los equipos de trabajo, requerimientos de los clientes, entre otras. Muchos de los productos de software deben ser construidos por personas organizadas en equipos de trabajo. Es difícil pensar que una sola persona pueda desarrollar un sistema complejo tratando de responder en términos de calidad y tiempo. En el caso de un desarrollador de software, la productividad se

ve afectada por la cantidad de defectos encontrados y arreglados (Linares, J. E, 2001), lo cual se traduce directamente en tiempos de desarrollo, costos y calidad. La calidad del software depende en gran medida de la calidad del proceso empleado. El proceso incluye actividades tales como la medición, planificación y la estimación entre otras. También debe permitir el mejoramiento continuo con base en indicadores previos, mediciones y datos históricos. Existen diferentes modelos de procesos que contribuyen al desarrollo de habilidades individuales y de equipo para construir software.

El PSP (Personal Software Process) es un proceso de desarrollo de software creado para el trabajo individual (Humphrey, 1995), el cual ayuda a los desarrolladores de software a mejorar su desempeño de una forma disciplinada, permite gestionar la calidad de sus productos en las diferentes actividades del proceso de desarrollo de software y establecer su propio programa de medición. Es un proceso para el desarrollo de software y el mejoramiento continuo a nivel personal, que considera aspectos como estimación de esfuerzo, tamaño y calidad del producto. PSP gracias a dichas características ha sido incorporado como tema o asignatura en planes de estudios de diversas universidades (Bermón-angarita et al., 2009).

El TSP (Team Software Process) es el modelo de proceso complementario al PSP y está orientado hacia equipos de trabajo (Humphrey, 2005), (Humphrey, 2006). Es un proceso que se enfoca en equipos auto dirigidos de desarrollo de software, los cuales planifican y realizan seguimiento de su trabajo, estableciendo metas, además de sus propios procesos y planes, buscando mantener altos niveles de productividad.

Este proyecto de investigación pretende plantear una estrategia de aprendizaje cooperativo para abordar el estudio de prácticas de PSP y TSP, en un curso de Programación de computadores, de estudiantes de Ingeniería de Sistemas y Computación de la Universidad del Quindío buscando de esta forma incrementar las experiencias de trabajo en equipo y permitiendo a los estudiantes el mejoramiento de su plan desarrollo de software, sus estimaciones, la calidad de su trabajo y sus planes de medición.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Diseñar una estrategia de aprendizaje apoyada en técnicas de trabajo cooperativo, que permita la adopción de algunas prácticas de PSP y TSP en un curso de Programación de computadores de la carrera Ingeniería de Sistemas y Computación de la Universidad del Quindío.

2.1. OBJETIVOS ESPECÍFICOS

- Analizar el estado del arte relacionado con la adopción del PSP y TSP en cursos de Programación en Universidades, con el fin de conocer su impacto en la formación de los estudiantes.
- Seleccionar un enfoque de aprendizaje cooperativo, que sirva de soporte al diseño de la estrategia de aprendizaje planteada.
- Diseñar los escenarios y actividades necesarias para confrontar las diversas situaciones de aprendizaje a través de un enfoque cooperativo que propicie en los estudiantes de los cursos de Programación la apropiación y aplicación de prácticas propuestas por PSP y TSP.
- Aplicar la estrategia de aprendizaje diseñada a través de una prueba piloto en un curso básico de Programación de computadores de la Universidad del Quindío.
- Validar los resultados obtenidos de la estrategia de trabajo cooperativo mediante un diseño de investigación cuasi-experimental basada en una técnica de carácter cuantitativo.

3. REFERENTE TEORICO

El presente capítulo tiene como propósito hacer una descripción de los conceptos y en general, de los elementos teóricos necesarios para abordar la investigación que se ha planteado en este proyecto; en él se hace un acercamiento al Personal Software Process (PSP), al Team Software Process (TSP) y al Aprendizaje Cooperativo; además, se analiza la forma en que estos conceptos se relacionan durante el proyecto y se describen algunos trabajos relacionados.

3.1. Fundamentos

3.1.1. Personal Software Process (PSP)

En la mayoría de las profesiones, y particularmente en el ejercicio de la Ingeniería, el trabajo competente requiere la utilización de prácticas preestablecidas, planes y procedimientos que permiten organizar y hacer más efectivo el trabajo a la vez que le facilitan al profesional el concentrarse en generar productos de mayor calidad(Duran & Gamboa, 2010). Con el fin de lograr todo lo anterior, se creó el Personal Software Process. El PSP fue creado por Watts Humphrey luego de crear CMM y con el fin de centrar la atención de la productividad ya no en las organizaciones, ni en los equipos de trabajo, sino en los individuos, ya que el desarrollo de software es una actividad que se realiza en una buena parte con trabajo individual.

Finalizando los años 80, Humphrey realiza una investigación para el desarrollo de pequeños programas de computador analizando un total de 62 proyectos durante los cuales fue madurando los elementos que harían parte de la primera versión de PSP. Entre 1993 y 1994, se impartieron los primeros cursos de PSP en la Universidad de Carnegie Mellon. Basado en estas experiencias, Humphrey publica la primera versión de PSP en 1994.

El PSP fue creado por Watts Humphrey principalmente para ayudar a los ingenieros a organizar y planificar su trabajo, documentar su rendimiento, gestionar los defectos y en general, mejorar su desempeño personal. Según (Duran & Gamboa, 2010), los principales objetivos de PSP son:

- Mejorar las estimaciones
- Mejorar la planeación y acompañamiento de cronogramas
- Proteger contra el exceso de compromisos
- Crear un compromiso personal con la calidad
- Concientizar al desarrollador sobre su compromiso en la mejora continua del proceso de desarrollo
- Aumentar la calidad a través de la reducción en la incidencia de errores
- Lograr una mayor precisión en las estimaciones de tamaño del software y el tiempo de desarrollo.

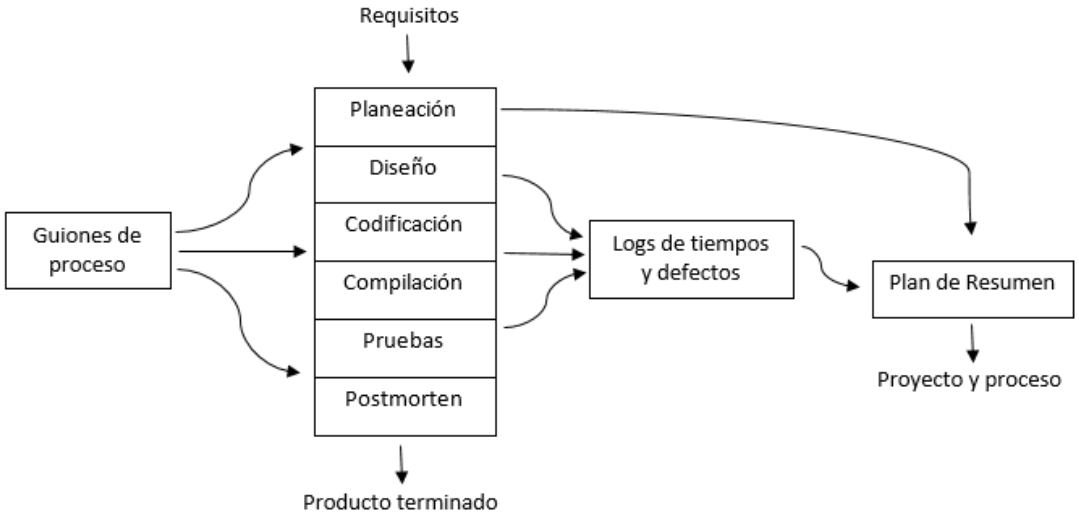
El diseño de PSP se basa en los siguientes principios de planeación y calidad: (Cardona, 2011)

- Cada ingeniero es esencialmente diferente; es decir, los ingenieros deben planear su trabajo y basar sus planes en sus propios datos personales.
- Para mejorar constantemente su funcionamiento, los ingenieros deben utilizar personalmente procesos bien definidos y medidos.
- Para desarrollar productos de calidad, los ingenieros deben sentirse personalmente comprometidos con la calidad de sus productos.
- Para hacer un trabajo de Ingeniería de Software de la manera correcta, los ingenieros deben planear de la mejor manera su trabajo antes de comenzarlo y deben utilizar un proceso bien definido para realizar dicha planeación del trabajo.
- Para que los desarrolladores lleguen a entender su funcionamiento de forma personal, deben medir el tiempo que pasan en cada proceso, los defectos que inyectan y remueven de cada proyecto y finalmente, medir los diferentes tamaños de los productos que llegan a producir

El flujo de procesos que plantea PSP es como sigue (W. S. Humphrey, 2000): En primer lugar

se realiza la planificación de las actividades. A continuación se llevan a cabo las fases de Diseño, Codificación, Compilación y pruebas. A la par que se van llevando a cabo las tareas, se registran los datos de los tiempos y defectos introducidos en las mismas. Luego de terminar el proyecto, se realiza un análisis post-mortem con el fin de completar el resumen del plan del proyecto. En la figura 1, se muestra el flujo de procesos PSP.

Figura 1. Flujo de procesos de PSP (W.S. Humphrey, 2000).



PSP integra una serie de plantillas que se deben diligenciar para obtener la información estadística sobre tiempos y defectos; esto con el fin de facilitar la medición de las actividades durante el flujo de procesos.

PSP es un proceso evolutivo e incremental que se encuentra descrito en un conjunto de niveles (W. S. Humphrey, 2000). Cada nivel se encarga de apoyar y extender el anterior nivel introduciendo al Ingeniero en nuevas prácticas (Tabla1), de tal forma que los niveles superiores de PSP adicionan características a los niveles ya implementados, lo cual minimiza el impacto de los cambios en los hábitos del Ingeniero; este deberá adaptar tan solo las nuevas técnicas a las ya conocidas y aplicadas. En la tabla 1 se muestran los niveles PSP.

Tabla 1. Niveles de PSP**.**

NIVEL	NOMBRE	ACTIVIDAD

PSP 0	Medición Personal	• Registro de tiempos • Registro de defectos
PSP 0.1	Registro de defectos	• Patrón de tipos de defectos • Patrón de codificación • Medida de tamaño • Propuesta de mejoramiento de procesos
PSP 1	Planeamiento Personal	• Estimación del tamaño • Informe de pruebas • Planteamiento de tareas • Cronogramas
PSP 2	Gerenciamiento de la calidad personal	• Revisiones de código • Revisiones de proyecto • Patrones de proyecto
PSP 3	Proceso Personal Cíclico	• Desarrollo cíclico

Niveles de PSP

- *PSP 0 – PSP 0.1: Línea Base:* La línea base de PSP provee una introducción y establece una base inicial para el tamaño, tiempo y defectos. En este nivel, los estudiantes pueden trabajar con sus métodos actuales pero siguiendo los pasos que se enuncian en la Tabla 2.

Tabla 2. Pasos PSP 0 – PSP 0.1

PASO	FASE	DESCRIPCIÓN
1	Planear	Planear el trabajo y documentar el plan
2	Diseñar	Diseñar el programa
3	Codificar	Implementar el diseño
4	Compilar	Compilar el programa y arreglar defectos encontrados
5	Probar	Probar el programa y arreglar los defectos encontrados

6	Postmorten	Registrar el tiempo, defectos y tamaño reales

Se registran tiempo de desarrollo, defectos y tamaño del programa en los formatos que PSP provee para tal fin. En PSP 0.1 se provee un formato adicional para registrar una propuesta de mejora. Este formato le da al Ingeniero una manera adecuada de registrar problemas durante el proceso y proponer soluciones.

- *PSP 1 – PSP 1.1* Gestión de proyectos personales: El nivel 1 se enfoca en las técnicas de gestión de proyectos. Aquí se introducen elementos como la estimación del esfuerzo y de tamaño, programación de la planeación y métodos de seguimiento de la programación.
- *PSP 2 – PSP 2.1 Gestión de la calidad personal:* Este nivel recoge los elementos de los anteriores niveles y agrega métodos de gestión de la calidad, entre ellos: Revisiones personales de diseño y código, notación del diseño, plantillas de diseño, técnica de verificación del diseño y métrica para gestionar la calidad del producto y del proceso. En general, la gestión de la calidad permitirá al Ingeniero hallar y remover todos los defectos antes de entrar a compilar.
- *PSP 3 Proceso cíclico personal:* Este nivel permite escalar PSP a proyectos más grandes sin sacrificar calidad y productividad. En este nivel se introducen elementos como: Diseño de alto nivel, revisión del diseño de alto nivel, planeación cíclica y ciclos de desarrollo basados en los procesos del nivel anterior. Se agregan dos formatos: Tamaño acumulado en cada ciclo y tiempo de desarrollo y defectos por cada ciclo. Igualmente, se agrega una bitácora para registrar aspectos que puedan afectar ciclos futuros.

Artefactos provistos por PSP en cada nivel.

En la tabla 3 se muestran las plantillas, tablas y los materiales usados por los estudiantes por cada nivel del PSP. Las plantillas, tablas y materiales se conocen en PSP como Scripts y varían según sea el nivel de PSP utilizado por el desarrollador.

Tabla 3. Materiales usados en cada nivel de PSP.

Scripts de procesos y resúmenes	PSP0	PSP0.1	PSP1	PSP1.1	PSP2	PSP2.1
Script de proceso PSP	X	X	X	X	X	X
Script de planeación PSP	X	X	X	X	X	X
Script de desarrollo	X	X	X	X	X	X
Script de revisión de diseño					X	X
Script de revisión de código					X	X
Script de Postmorten	X	X	X	X	X	X
Instrucciones y Resumen del plan de proyecto	X	X	X	X	X	X
Script de Estimación			X	X	X	X

Entregables por parte del estudiante en cada nivel PSP.

Los estudiantes de cada nivel deben estar preparados para manejar los diferentes estándares y las plantillas, y diligenciarlas para obtener la información estadística sobre tiempos y defectos, para medición de logros, así como los avances alcanzados en el desarrollo de los diferentes niveles de PSP. De esta forma, se logran desarrollar destrezas que a medida que el alumno pasa al siguiente nivel aprende sobre las prácticas de PSP y asume retos que incrementan la calidad en el desarrollo de software. En la tabla 4 es posible observar los entregables que deben hacer los estudiantes en cada nivel de PSP. Los entregables son elementos como plantillas o formatos que el desarrollador debe utilizar para recoger datos estadísticos y determinar su desempeño dependiendo del nivel de PSP en que se encuentre trabajando.

Tabla 4. Entregables por parte del estudiante en cada nivel PSP

Formatos, Plantillas, estándares e instrucciones	PSP0	PSP0.1	PSP1	PSP1.1	PSP2	PSP2.1

Log de registro de tiempos	X	X	X	X	X	X
Log de registro de defectos	X	X	X	X	X	X
PIP		X	X	X	X	X
Estándar de Codificación		X	X	X	X	X
Plantillas de reporte de pruebas			X	X	X	X
Plantilla de estimación de tamaño			X	X	X	X
Plantilla de planeación de tareas				X	X	X
Plantillas de planeación de cronograma				X	X	X
Lista de chequeo de revisión de diseño					X	X
Lista de chequeo de revisión de código					X	X
Plantilla de especificación de casos de uso						X
Plantilla de especificación funcional						X
Plantilla de especificación de estado						X
Plantilla de especificación lógica						X

3.1.2. Team Software Process (TSP)

El proceso de software en equipo o TSP, por sus siglas en inglés, fue creado con el fin de orientar el trabajo de desarrollo de software en equipo, ya que esta actividad es esencialmente un proceso grupal donde la comunicación y el apoyo entre los miembros de este juegan un papel

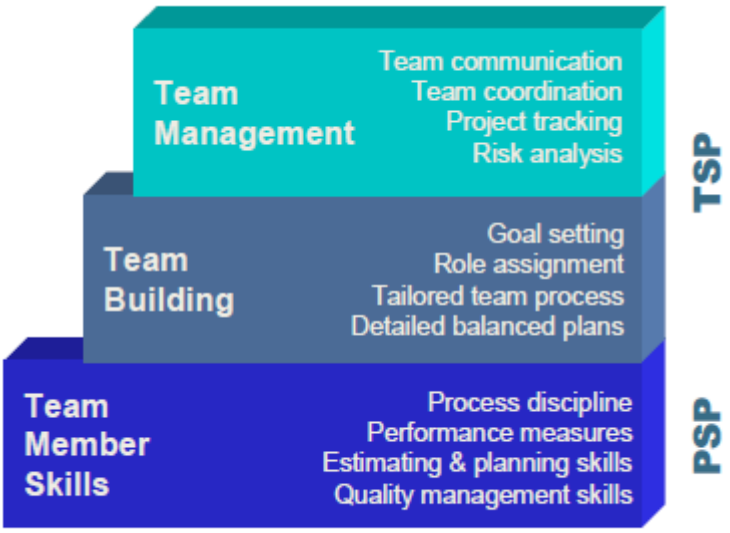
esencial. El objetivo fundamental de TSP es crear un ambiente que le de soporte al trabajo individual disciplinado a la vez que mantener equipos autodirigidos. TSP guía a los equipos y a sus miembros a obtener mejores resultados en términos de costos, cumplimiento de calendario, gestión de la calidad y en otros tantos rasgos en los procesos de desarrollo de software(Noopur Davis Julia Mullaney, 2003)

TSP puede ser usado en todas las fases del proceso de desarrollo de software: En la elicitación de requisitos, durante el diseño, implementación, pruebas e incluso durante el mantenimiento. Puede utilizarse para soportar equipos multidisciplinarios que varían en tamaño desde 2 Ingenieros o estudiantes hasta equipos de cien individuos. Igualmente, puede ser usado para desarrollar diversos tipos de productos, desde sistemas de control embebidos hasta aplicaciones comerciales cliente servidor.

TSP ha sido concebido para trabajar sobre la base de PSP. Dado que este último muestra a los Ingenieros como medir su trabajo y usar esos datos para mejorar su desempeño, PSP guía el trabajo individual, mientras que TSP guía el trabajo en equipo y crea un ambiente en el cual los individuos tienen las metas del equipo claras y el papel que desempeñan dentro de este. PSP y

TSP son herramientas que proveen las habilidades necesarias, disciplina y el compromiso requerido por los proyectos de desarrollo de software exitosos. En la Figura 2 puede verse como se articulan PSP y TSP.

Figura 2. Articulación de PSP y TSP(Noopur, mullaney. 2003).



TSP se encuentra basado en los siguientes principios(W. S. Humphrey, 2000).

- Los Ingenieros conocen casi la totalidad de los aspectos que caracterizan su trabajo y pueden hacer los mejores planes.
- Cuando los Ingenieros planean su propio trabajo, ellos se comprometen con lo planeado
- Un seguimiento preciso de un proyecto requiere planes bien detallados. Únicamente el personal que realiza el trabajo es capaz de recoger con precisión dichos datos.
- Para minimizar el tiempo del proyecto, los Ingenieros deben equilibrar su carga de trabajo
- Para maximizar la productividad, el primer foco de atención debe ser la calidad.

Como TSP pretende orientar el quehacer de los equipos de desarrollo de software, TSP es ideal para atacar los problemas que más frecuentemente se presentan en los equipos de desarrollo de software (W. S. Humphrey, 2000).

- Liderazgo inefectivo

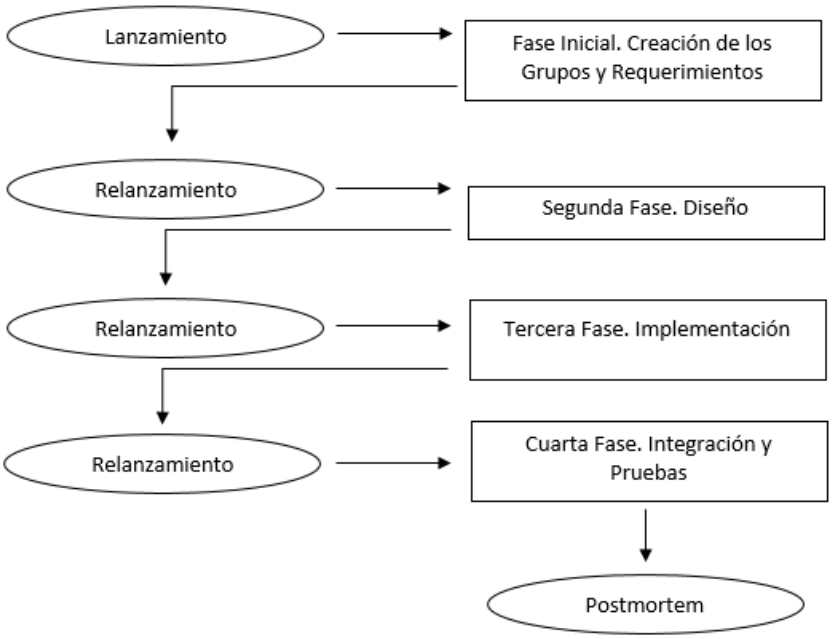
- Incapacidad para la cooperación y el compromiso
- Falta de participación
- Falta de confianza
- Ausencia de calidad
- Inefectiva evaluación entre iguales
- Modificaciones eternas

TSP propone una serie de roles los cuales cumplen unas tareas puntuales en los proyectos y apoyan a los demás miembros del equipo. Estos roles son los siguientes:

- *Líder del Equipo*: Dirige al equipo, se asegura que todos reporten sus datos de los procesos y completen su trabajo tal y como se planeó. Realiza los reportes semanales del avance del equipo.
- *Gestor de desarrollo:* Guía al equipo en el diseño y desarrollo del producto.
- *Gestor de Planificación:* Apoya y guía al equipo en la planificación y seguimiento del trabajo.
- *Gestor de Calidad/Proceso:* Apoya al equipo en definir sus necesidades acerca del proceso, establecer y administrar el plan de calidad. Genera estándares para obtener un trabajo uniforme. Modera las inspecciones y revisa cada artefacto generado.
- *Administrador de Requerimientos/Soporte*: Dirige al equipo en el desarrollo de requerimientos de software y ayuda a dar a conocer la tecnología y en las necesidades de apoyo administrativo. Administra el plan de configuración

Para ejecutar un proyecto, TSP propone la realización de 4 fases que permiten la construcción adecuada del producto (Figura 3). El proyecto debe empezar o terminar con alguna fase, o bien, puede ejecutarse desde el principio hasta el final. Antes de cada fase, el equipo debe planificar y organizar su trabajo a través de una puesta en marcha completa, launch o relaunch. En la Tabla se explica cada una de las fases de TSP.

Figura 3. Flujo de proceso de TSP relacionado con las fases(García. 2011).



En la tabla 5 se hace la descripción de las fases del TSP (W, Humphrey 2000). Se puede observar el proceso o pasos para ejecutar con éxito determinado proyecto. Las fases brindan los parámetros a seguir para el fin u objeto de TSP.

Tabla 5. Fases de TSP (W, Humphrey 2000).

FASE	DESCRIPCIÓN
Lanzamiento	Identifica las necesidades de los clientes, se realiza la conformación de los equipos de trabajo y a cada una de las personas del grupo se le asigna un rol teniendo en cuenta su experiencia, desempeño e intereses dentro del proyecto.
Estrategia	Se establece la forma como se llevará a cabo el desarrollo que se producirá en cada ciclo, teniendo en cuenta los riesgos necesarios para su implementación. Se realizan las primeras estimaciones enfocadas al esfuerzo y a las LOC.
Planeación	Se identifican las tareas a realizarse durante todo el proyecto y las mismas se asignan a cada uno los miembros del equipo. En esta fase

	se realiza la estimación de los diferentes artefactos que se generarán en el proyecto y se define un plan de calidad.
Requisitos	Se refinan las necesidades de los clientes y se especifican los requisitos. También se realiza un primer plan de pruebas para el sistema.
Diseño	Se elabora un diseño de alto nivel en donde se especifica y examina cada parte identificada, de acuerdo con los estándares definidos. Se define también el plan de integración.
Implementación	Se produce el código, de acuerdo con un estándar de codificación definido por el equipo y se realizan pruebas unitarias.
Pruebas	Se integran todos los módulos producidos durante la fase de implementación y se define un plan de pruebas con casos de pruebas, por el grupo de desarrollo.
Postmortem	Se generan las evaluaciones del equipo, lecciones aprendidas y se presenta el estado del proyecto.

3.1.3. El Aprendizaje Cooperativo

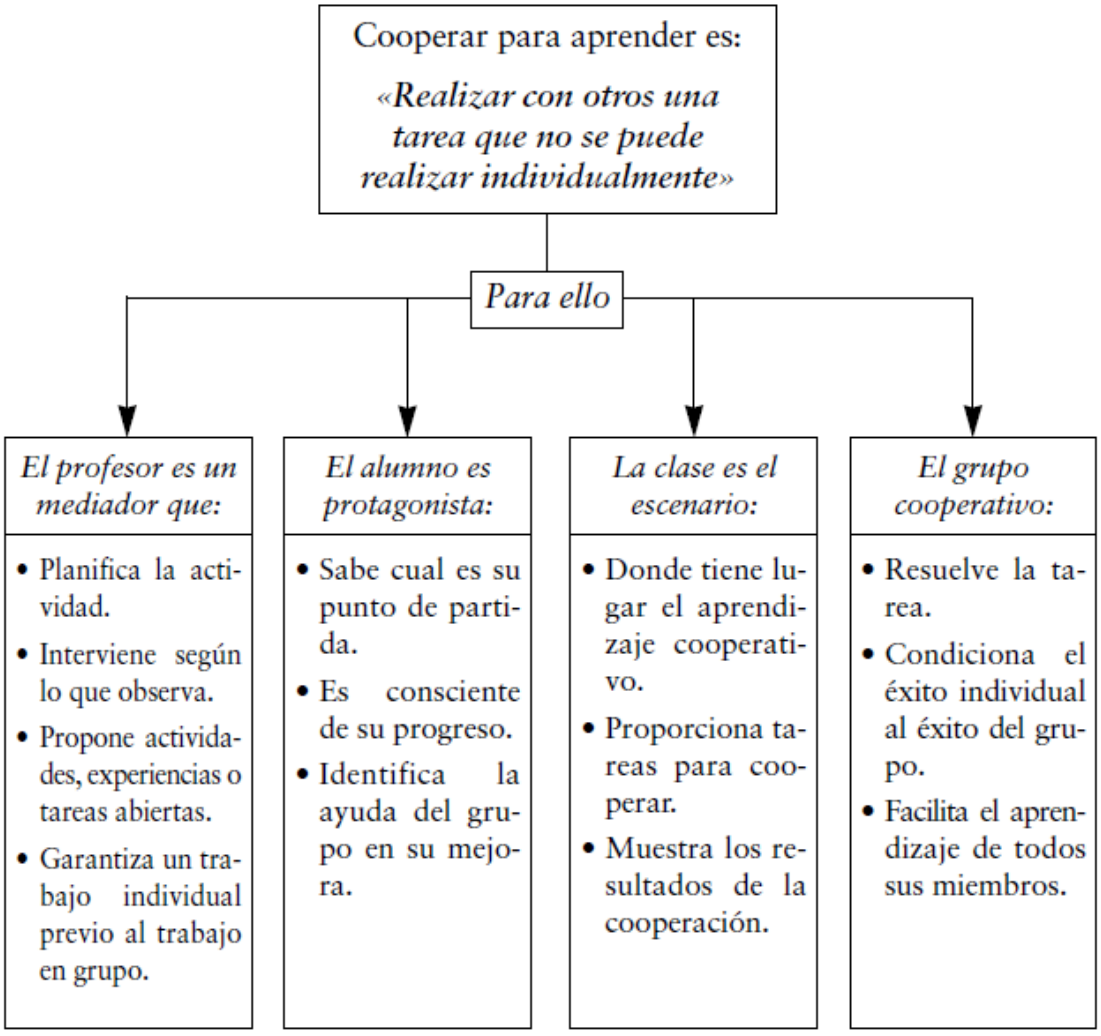
Con la creciente disponibilidad de información que han generado las nuevas tecnologías, se hace necesario crear estrategias que faciliten el acercamiento al conocimiento y en especial al aprendizaje. A esto se suma la exigencia de la sociedad de formar equipos de trabajo que incrementen la productividad de las organizaciones; hasta tal punto se ha llegado, que un individuo corre el riesgo de no incorporarse al mercado laboral si no es mínimamente competente para relacionarse y cooperar con otros.

Estas competencias y habilidades son de uso común en los proyectos de desarrollo de software. Allí, se requiere el trabajo en equipo en un sinnúmero de actividades que abarcan desde la obtención de los requisitos hasta realizarle mantenimiento a un producto software. Es por esta razón que el aprendizaje cooperativo es una excelente herramienta para facilitar el aprendizaje de los alumnos que estudian Ingeniería de Sistemas y Computación, ya que relaciona la necesidad

que tienen de aprender a trabajar en equipo para su desempeño profesional, así como les brinda una herramienta que potencia el aprendizaje.

Según (Linares, 2007), el aprendizaje cooperativo podría considerarse como un sistema de aprendizaje en el que la finalidad del producto académico no es exclusiva, sino que desplaza aquella en busca de la mejora de las propias relaciones sociales, donde para alcanzar tanto objetivos académicos como los relacionales se enfatiza la interacción grupal, potenciando las interacciones positivas. En el siguiente cuadro se resume la esencia de lo que es el aprendizaje cooperativo.

Figura 4. Aprendizaje cooperativo(Uriz Bidegain, 1999).



En el contexto educativo, pueden darse diversas estructuras sociales que es necesario entrar a a definir; entre ellas se encuentran: la cooperación, la competición y la individualización (Dominic, 2007).

1. La cooperación es una situación social en la que los objetivos de las personas se encuentran enlazados de una forma que un individuo solo puede alcanzar su objetivo solo si los demás alcanzan los suyos, y cada persona será recompensada en función del trabajo de los demás miembros del equipo.
2. La competición es una situación social en la que cada persona alcanzara sus objetivos si los

demás no logran los suyos y recibirá la máxima recompensa solo si los demás logran recompensas inferiores

3. La individualización es una situación social en la que el logro de los objetivos por parte de una de las personas es independiente del éxito o fracaso que los demás hayan tenido en el logro de los suyos, por lo que recibirá su recompensa solo en función de su trabajo personal.

Se ha demostrado en estudios como (Johnson, Johnson, & Holubec, 1994), que la situación social cooperativa alumno/alumno posee un valor intrínseco en tres campos específicos:

- La explicita importancia que presentan las relaciones entre iguales para el incremento de las habilidades sociales de los estudiantes y el control de sus impulsos agresivos.
- Posibilita la relativización del punto de vista propio, lo que resulta un elemento esencial para el desarrollo cognitivo y social, ya que se ha demostrado que potencia aquellas capacidades que permiten la presentación y la transmisión de la información, la cooperación y la solución constructiva de los conflictos, la autonomía en los juicios moral y cognitivo.
- La interacción entre iguales tiene una decisiva influencia, tanto sobre el incremento de las aspiraciones de los estudiantes, como la mejora de su rendimiento académico.

Las ideas anteriores, sumadas a una gran cantidad de evidencia empírica han hecho que en los últimos 30 años se valore la contribución en el aprendizaje que pueden hacer las relaciones entre iguales, tanto en los aspectos vinculados al desarrollo cognitivo y social, como a la propia socialización como (Johnson, Johnson, & Holubec, 1994), con lo que empiezan a cobrar importancia un grupo de métodos instruccionales que pretenden ser una respuesta clara y contundente a las deficiencias presentadas por la metodología tradicional usada hasta entonces. Estos nuevos métodos son los que presenta el aprendizaje cooperativo.

Técnicas de Aprendizaje Cooperativo.

1. **Aprendiendo Juntos (Johnson y Johnson, 1975).**

De acuerdo a lo afirmado por (Sáez, 2002), quien ha estudiado a los autores Johnson y Johnson, creadores del método, en el aprendizaje cooperativo se consideran de importancia los siguientes elementos:

a. Interdependencia positiva: Existe interdependencia positiva cuando cada miembro del equipo siente que está vinculado a los demás de modo que no puede lograr el éxito hasta que los demás no lo alcancen también, por lo que coordina sus esfuerzos con los de los demás para la consecución de la tarea.(Johnson y Johnson.,1994, Johnson y Johnson, 2002)

b. La interacción positiva "Frente a Frente": Se entiende como la animación y colaboración reciprocas para conseguir los objetivos comunes y que se manifiesta en que los miembros del grupo: Se prestan ayuda y apoyo sintiendo la necesidad de dar y recibir ayuda; intercambian recursos de información y materiales, estableciendo una comunicación eficaz; verifican a través del feed-back tanto los procedimientos como la satisfacción que se siente al trabajar en equipo; estimulan con opiniones y confrontaciones las creencias, suposiciones y pensamientos ajenos, promoviendo una visión más amplia y global de las cuestiones planteadas; se encuentran motivados por el bien común; se influyen indirectamente a través del control normativo del comportamiento del grupo.

c. La enseñanza-aprendizaje de competencias interpersonales y la formación de pequeños grupos. No basta con juntar estudiantes y decirles que trabajen de forma cooperativa para que lo hagan, es necesario explicarles normas y estrategias para que lleven a cabo el trabajo de una mejor manera.

d. La revisión y control del comportamiento del grupo. La heterogeneidad entre los elementos de cada equipo hacen que se requiera un continuo y constante control del propio comportamiento durante el desarrollo como una vez terminada la tarea común.

2. **Student Team Learning de Slavin.**

Con base en lo expuesto por (Sáez, 2002), quien hace un análisis de esta técnica, una organización escolar hace referencia esencialmente a dos elementos: La estructura didáctica de la tarea y la estructura incentiva del estudiante. La primera hace referencia a las estrategias que puede utilizar el docente para realizar una actividad; la segunda se refiere a los instrumentos usados por el profesor para motivar a los estudiantes. Para este investigador, el aprendizaje cooperativo más que un método didáctico es un conjunto de técnicas específicas, por esto, cuando habla de aprendizaje cooperativo se está refiriendo al modo específico de articular estos dos elementos fundamentales de toda organización didáctica. Para Slavin, según (Sáez, 2002), son tres los elementos característicos y esenciales del método Student Team Learning:

a. La recompensa del grupo: Expresan el reconocimiento al estudiante por los resultados obtenidos

b. La responsabilidad individual: El éxito del equipo depende del nivel de aprendizaje que cada cual está dispuesto a alcanzar.

c. La misma oportunidad de éxito: La condición cooperativa asegura que todos los miembros tengan la posibilidad de conseguir el éxito mejorando resultados anteriores.

3. **Group Investigation de Sharan y Sharan.**

Es una modalidad de aprendizaje cooperativo en la que destacan los aspectos de la interrelación grupal: la investigación, la interacción, la interpretación, la motivación intrínseca (Sáez, 2002). De acuerdo con estos autores, el punto de partida de todo aprendizaje es la organización de una investigación, que suscita una motivación intrínseca, una interacción y una confrontación para poder llevarla a cabo.

a. La investigación: El aprendizaje puede ocurrir de muchas formas y mediante diversas secuencias cognitivas o procedimentales. La investigación es el procedimiento elegido como actividad de aprendizaje y forma el motor que orienta la actividad, las interacciones, las evaluaciones, el aprendizaje y la motivación intrínseca.

b. La interacción: La posibilidad de interactuar de cambiar opiniones, de discutir es un componente fundamental del trabajo en equipo

c. La interpretación: Para que los individuos aprendan, deben desarrollar procesos cognitivos de comprensión, interpretación o integración. Deben confrontar y elaborar el significado de los conocimientos conseguidos, en términos del problema que se está abordando.

d. La motivación intrínseca: Más que una estrategia, es un componente fundamental del proceso de aprendizaje. Este tipo de motivación hace que los individuos se comprometan y se sientan motivados a realizar las tareas.

4. **Structural Approach de Kagan y Kagan.**

A partir de lo expuesto por (Sáez, 2002), quien retoma lo propuesto por Kagan y Kagan, se consideran como aspectos claves de esta técnica, los siguientes elementos:

a. La estructura: Corresponde a la secuencia de acciones que los individuos dentro del grupo llevan a cabo para alcanzar determinado objetivo.

b. Los principios fundamentales: Según estos autores, son 4 los principios fundamentales del aprendizaje cooperativo: la interacción simultánea o, el trabajo simultáneo de 4 o 5 grupos en que se puede dividir una clase. La igualdad de posibilidades de participación de los estudiantes. La interdependencia positiva. La responsabilidad individual.

c. La construcción del grupo y de la clase: El docente debe propiciar las condiciones adecuadas al interior de la clase para que los equipos puedan producir resultados positivos.

d. El equipo: Un equipo es más que un grupo, ya que cohesión entre los individuos que lo componen

e. La conducción de la clase: El docente debe conducir la clase de manera diferente al enfoque tradicional, deberá planificar, animar y hacer seguimiento a los procesos. El estudiante es un sujeto activo que participa e interactúa en las actividades y consecución de los objetivos.

f. Competencias sociales del grupo. El desarrollo de competencias sociales es fundamental para llevar a cabo un aprendizaje cooperativo. Estas competencias debe enseñarse, practicarse y evaluarse.

5. **Complex Instruction de Cohen.**

Este autor parte de la premisa de que los individuos que conforman un equipo son clasificados y clasifican a los demás miembros de acuerdo a un estatus (Sáez, 2002). Este estatus no solo tiene implicaciones sociales, sino también, en el aprendizaje; por tanto, a la hora de preparar un grupo de trabajo, se debe partir de este status, mientras que las estrategias del aprendizaje cooperativo tienden a superar los límites de partida que amenazan los resultados que el método quiere conseguir. Para evitar lo más posible el efecto del estatus es necesario establecer condiciones:

a. Modificar los prejuicios de alumnos y profesores
b. Preparar a los estudiantes a la cooperación por medio de la enseñanza de competencias cooperativas específicas.
c. Organizar tareas complejas. Que le dan la oportunidad a los miembros del grupo de aprender y contribuir al trabajo en equipo.
d. Dar a cada miembro del grupo el rol o tarea a desempeñar
e. Evaluar y optimizar el trabajo de grupo

6. **Collaborative Approach.**

Esta técnica puede ser definida como una orientación a encontrar de modo genérico una colaboración eficaz entre los estudiantes organizados en pequeños grupos para el aprendizaje o el intercambio social o cultural (Sáez, 2002). El collaborative Approach parte de tres aspectos fundamentales:

a. Los estudiantes deben tener experiencias de grupo no basados solo en la amistad, sino en grupos cooperativos.
b. Para formar grupos cooperativos, es necesario enseñar a los estudiantes habilidades de comunicación, de intercambio de información, de trabajo eficaz compartiendo las cargas para alcanzar el objetivo común.
c. La capacidad de saber afrontar y superar los conflictos que puedan surgir como resultado de la interacción entre los individuos.

7. **Peer Tutoring.**

Esta técnica se basa en la colaboración que un estudiante le presta a uno de sus compañeros de grupo que ha solicitado ayuda para aprender un determinado tema. En esta técnica, los grupos de trabajo cooperativo se reducen a un número de dos integrantes, donde uno de ellos cumple el papel de tutor y el otro de alumno. El tutor cumple el rol de maestro enseñando los conceptos de la unidad temática y el alumno aprende dichos conceptos. Esta dinámica entre las parejas de estudiantes deberá ser orientada por el docente.

(Serrano y Calvo: 1994, p-24) consideran que deben de cumplirse las siguientes condiciones para que la Tutoría entre iguales ayude a mejorar el rendimiento de los estudiantes implicados:

- El alumno tutor debe responder a las necesidades de ayuda de su compañero.
- La ayuda que proporcione el tutor a su compañero debe consistir en explicaciones detalladas sobre el proceso de resolución de un problema y nunca debe proporcionarle soluciones ya hechas.

Los pasos de la aplicación de esta técnica son los siguientes:

a) Preparación: Selección de los estudiantes tutores y de los estudiantes aprendices
b) Diseño de las sesiones de tutoría: Estructura, contenidos, recursos, mecanismo de evaluación.
c) Formación de los equipos: Alumno tutor y alumno aprendiz.
d) Formación de los tutores
e) Inicio de las sesiones: Se llevan a cabo bajo la supervisión del profesor.
f) Revisión del proceso: Mediante reuniones con los alumnos tutores.

8. **Jigsaw.**

Esta técnica fue creada por (**Aronson, E. y colaboradores,** 1978). Intenta colocar en una situación extrema de interdependencia positiva, creando las condiciones necesarias para que el trabajo de cada miembro del equipo sobre una parte del material de estudio, sea completamente necesario para que los demás miembros del equipo puedan completar sus conocimientos.

Esta técnica es muy útil en áreas de conocimiento en las que los contenidos pueden ser

fácilmente fragmentados (historia, literatura, ciencias experimentales, entre otros). La técnica Jigsaw consiste de los siguientes pasos:

a) La clase se divide en grupos de 4 o 5 miembros cada uno.
b) El material objeto de estudio se divide en el número de miembros que tienen los grupos, de tal forma que cada miembro de cada grupo recibe una parte del material a estudiar.
c) Cada miembro del equipo prepara su parte basado en la información que le haya dado el profesor o que él mismo haya buscado.
d) Luego, con los integrantes de los demás equipos que hayan estudiado el mismo subtema, forman un grupo de expertos para intercambiar información, profundizan en conceptos clave, clarifican dudas, etc.
e) Por último, cada experto retorna a su equipo y se encarga de explicar a los demás miembros la parte que ha preparado.

De esta forma, todos los estudiantes cooperan, ya que cada uno tiene una parte del "rompecabezas" necesario para culminar con éxito el estudio del tema propuesto.

3.2. Antecedentes

Desde que Watts Humphrey presentó el PSP en su libro "*A Discipline for Software Engineering"*, se han llevado a cabo diversas investigaciones sobre el impacto que genera el uso de PSP en cursos de pregrado y posgrado en diferentes Universidades (Hayes &Over, 1997), (Towhidnejad & Hilburn, 1997), (Hayes W, 1998), (Wesslén, 2000), (Börstler, Carrington, & Hislop,2002), (Runeson, 2001). También se han realizado trabajos experimentales de PSP en cursos de Ingeniería de Software (Venkatasubramanian, Roy, & Dasari, 2001).

Igualmente se tienen reportes de lecciones aprendidas producto de la implementación de PSP en el sector académico con el apoyo de la industria de software (Venkatasubramanian et al., 2001). También se tienen experiencias académicas relacionadas con TSP (Sussy, Calvo-Manzano, Gonzalo, & Tomás, 2008).

Los artículos relacionados con el PSP y con TSP que hacen parte de la revisión de este documento, se buscaron en la Web utilizando las herramientas google schollar, Microsoft Academic, ACM Digital Library y Science Direct. Se hizo un filtrado de aquellos artículos que en su título relacionaran las palabras Personal Software Process, Team Software Process y con las siglas PSP, TSP. También se realizaron búsquedas relacionadas con las experiencias académicas que implementan PSP y TSP, así como los simposios realizados desde el año 2006 hasta el 2012 por el SEI[1].

3.2.1. Experiencias académicas con PSP.

Para el análisis de los trabajos relacionados se retoma lo propuesto por (Börstler et al., 2002), en donde destacan tres factores primarios que influyen en la enseñanza de PSP: entorno de trabajo, nivel de cobertura, y las herramientas de apoyo. El entorno de trabajo hace referencia al público objetivo, el nivel del curso, y el contenido de la asignatura. El nivel de cobertura está asociado con las prácticas del PSP que se aplican. Las herramientas, se encuentran relacionadas con los medios de soporte para el registro de cada una de las actividades que propone el PSP. En este trabajo se aporta un nuevo factor de análisis, asociado con la aplicación o no de una estrategia de aprendizaje.

En la Tabla 6 se describen los resultados obtenidos de la implementación de PSP en diferentes universidades del mundo. Con base en ella se puede establecer que cada una de las experiencias reportadas tiene sus particularidades, dado el contexto y los intereses de formación para sus estudiantes.

Tabla 6. Experiencias académicas de PSP. Adaptado de (Cardona, 2012)

Universidad	**Estudiantes Objetivo**	**Nivel de cobertura**	**Herramientas de soporte de**	**Estrategia de Aprendizaje**

[1] http://www.sei.cmu.edu/tspsymposium/2012/proceedings.cfm

			PSP	
Lund	Pregrado y Posgrado	Full PSP[2]	Hojas de Cálculo	No hace referencia
Zagreb	Pregrado	PSP-Lite[3]	PSP-Tool local	No hace referencia
Purdue	Pregrado	PSP-Lite	Hojas de Cálculo	No hace referencia
Carlos III	Pregrado	PSP-Lite	StudentWorkbook	No hace referencia
Birla	Pregrado	Full PSP	Formularios de TSP	No hace referencia
Tecnológico de Antioquia	Pregrado	PSP-Lite	Hojas de Cálculo	No hace referencia
Embry-RiddleAeronauticalUniversity	Estudiantes de Pregrado	Full PSP	Formularios de PSP	No hace referencia
Universidad del Quindío	Estudiantes de Pregrado	PSP-Lite	Hojas de Cálculo	Implementa una estrategia de aprendizaje
Corporación Universitaria Adventista	Estudiantes de Tecnología	PSP Nivel 0	Hojas de Cálculo	No hace referencia

[2] Hace referencia a la aplicación de todo el cuerpo de conocimiento de PSP.

[3] Hace referencia a una versión simplificada o adaptada de PSP.

Universidad de Yucatan	Estudiantes de pregrado	Full PSP	ProcessDashBoard	No hace referencia
Instituto Tecnológico de Antioquia	Estudiantes de pregrado	PSP Nivel 0	Hojas de calculo	Aprendizaje Significativo

A continuación se detallan las experiencias académicas relacionadas con PSP.

1. En la ***Universidad de Lund*** es realizado un estudio en el que se compara el desempeño de estudiantes de primer año universitario que han tomado un curso de PSP, con estudiantes de posgrado y profesionales de la industria que han realizado cursos de PSP (Runeson, 2003). El contenido del curso se ajustó a lo planteado en el primer libro de Humphrey (W. Humphrey, 1995). Los estudiantes de postgrado que tomaron el curso de PSP tenían experiencia previa en el manejo de lenguajes de programación siendo el más usado por ellos C. Los estudiantes de primer año recibieron un curso inicial de Java y un curso básico de estadística necesario para llevar a cabo tareas asignadas en el curso PSP. El objetivo del estudio fue validar dos hipótesis relacionadas con la mejora del proceso y el desempeño de los participantes. Los resultados encontrados sugieren que los tres grupos de participantes en los cursos PSP mejoran en aspectos relacionados con la calidad como la reducción de defectos y la precisión en la estimación, de tal modo que el estudio concluye que esto se debe más a la naturaleza del proceso, antes que a las diferencias en las personas. No obstante, otra cosa sucede con el desempeño, ya que se notó que sí existe diferencia entre los estudiantes de primer año con respecto al desempeño de los estudiantes de postgrado, lo cual sugiere que la formación previa es importante al momento de realizar un curso de PSP.

2. Investigadores de la ***Universidad de Zagreb*** (Car, 2003) realizaron una investigación con estudiantes de pregrado en la que ofrecen un curso de PSP con el objetivo de fomentar en ellos una actitud profesional hacia la producción de software. El curso fue orientado en

una electiva de ingeniería de software a estudiantes que previamente habían tomado cursos básicos de programación de computadores, análisis y diseño de sistemas, bases de datos, redes y comunicaciones. Solo le fueron presentados a los estudiantes los conceptos básicos del PSP en los niveles 0 y 1 y las tareas principales consistieron en recolectar el tiempo de programación, los defectos y la medida del tamaño del software que construyeron como ejercicio. Finalmente se incorporaron el nivel PSP 1.1 donde los estudiantes debían planificar el cronograma y los tiempos. En cuanto a las herramientas para llevar el registro de los datos, se utilizaron hojas de cálculo. Los resultados obtenidos sugieren que los conceptos básicos de PSP ayudan a los estudiantes a llevar un proceso de desarrollo más organizado y les facilitan la administración del proceso; también se concluye que es posible enseñar en un curso los conceptos generales de un lenguaje de programación además de los conceptos básicos del PSP.

3. Profesores del ***Tecnológico de Antioquia***(Duran & Gamboa, 2010), han estudiado los efectos que producen en los estudiantes de Ingeniería la inclusión de prácticas PSP en cursos de Programación. Según estos autores, “El PSP enseña a ingenieros y futuros ingenieros, cómo administrar la calidad de sus productos y cómo hacer compromisos que ellos puedan cumplir. Puede ser empleado en muchas fases en el ciclo de desarrollo de programas pequeños, definición de requisitos, documentación, pruebas y mantenimiento”. De acuerdo a este trabajo, se debe proporcionar a los estudiantes los conceptos básicos del PSP acompañado de sus prácticas fundamentales desde el primer semestre de carrera, buscando que los estudiantes se familiaricen con el tiempo que se tardan en realizar las tareas de programación, ya que esto les permitirá conocer su productividad y es un paso previo para mejorar su desempeño. Luego, se pretende que los estudiantes reconozcan el valor de registrar los datos de esfuerzo y tamaño y que puedan usar estos datos en la planificación de sus proyectos y analizar su efectividad personal. En las asignaturas de ingeniería de software se implementaron las metodologías a nivel grupal (TSP) y corporativo (CMMI).
4. Investigadores en la ***Universidad de Purdue*** (Lisack, 1997) aplicaron una versión simplificada del PSP apoyados en el libro "Introducción al proceso personal de software"

de Watts Humphrey, para ser enseñada a estudiantes de pregrado en los cursos iniciales de programación. Estos estudiantes habían tomado previamente cursos introductorios de sistemas de información, arquitectura de computadores y bases de datos. El proyecto consistió en que los estudiantes llenaran 3 de los formularios del PSP: El time recording log, Job number log y defect log; estos formularios fueron mantenidos electrónicamente en un procesador de texto y en una hoja electrónica. A medida que les eran presentados los conceptos fundamentales del PSP, los estudiantes recibían la formación en programación de computadores. Se les pidió a los estudiantes que leyeran un capitulo por semana del libro de Humphrey y luego, el profesor resumía los conceptos claves en clase. En el siguiente semestre, se les pidió a los estudiantes que leyeran 2 capítulos por semana y que extractaran los conceptos claves. Entre los hallazgos está que los estudiantes no llenaban correctamente los formatos o, no los llenaban. Al final del estudio, se les preguntó a los estudiantes por el proceso y sobre lo que habían aprendido con relación al uso de los script, la metodología de trabajo y aplicación de PSP en trabajos futuros de programación. En general, afirma el estudio, los estudiantes comprendieron los temas vistos de PSP, fueron capaces de llenar los formatos aunque no se encontraban muy convencidos de ello. La mayoría de los estudiantes admitieron que no utilizaron suficiente tiempo llenando los logs; alrededor de la mitad admitieron que los datos en sus logs no eran lo suficientemente precisos y, en general, no reconocieron el valor del PSP en los proyectos de software.

5. En la ***Universidad Carlos III*** (Bermón-Angarita et al., 2009) se realizó una investigación para conocer el grado de aceptación de las prácticas de PSP en primer semestre de Ingeniería Informática. Allí, se impartió un curso básico de PSP en la materia de Introducción a la Ingeniería de Software que constaba de 8 grupos y un total de 278 estudiantes. El curso se planificó con una duración de 11 semanas en modalidad cuatrimestral. En el curso se impartieron conceptos referentes a los niveles PSP0, PSP0.1, PSP1 y PSP1.1. Durante el curso no se profundizó en aspectos de programación. El curso fue evaluado a través de trabajos prácticos, exámenes y un trabajo final. A partir de las evaluaciones realizadas, más del 50% de los estudiantes consideraron beneficioso

para su formación la incorporación de conceptos de PSP en la carrera. El estudio concluye, que la inclusión de PSP en etapas tempranas de aprendizaje permite a los estudiantes entender mediante una serie de ejercicios prácticos qué es un proceso y la disciplina para realizarlo, así como adquirir el hábito de realizar mediciones. Además, al recolectar datos sobre cómo desarrollan sus programas, se obtiene una línea base que ayudará a planificar trabajos futuros. Igualmente, se encontraron algunos problemas comunes a las otras investigaciones realizadas como: El llenado errado o incompleto de los formularios PSP, incomprensión de las ventajas que trae la metodología PSP en los proyectos de software y algunos problemas de programación.

6. Otra experiencia de la enseñanza de conceptos y prácticas de PSP se puede encontrar en la ***Universidad de Birla***(Venkatasubramanian et al., 2001) en la India. Allí, se orientó un curso de PSP en la materia de Ingeniería de Software durante un semestre a 36 estudiantes que como característica principal tenían conocimientos en programación de computadores en C, C++ y Java. El objetivo de los investigadores, fue exponer a los estudiantes a un proceso de software definido y medido. Durante el semestre, los estudiantes tuvieron 10 lecturas para conocer los métodos y modelos matemáticos para llevar a cabo los ejercicios de PSP. Luego, los estudiantes llenaron los formularios de tiempo y defectos. Se observó que los estudiantes registraron el tiempo y los defectos generados en los ejercicios de programación; también aprendieron a estimar el tiempo que tardan en terminar un ejercicio. Como novedad en esta investigación, se tuvo retroalimentación permanente por parte de los estudiantes; a partir de ella, se pudo concluir que para los estudiantes, llenar los formularios fue un trabajo tedioso aunque necesario. Algunos estudiantes manifestaron no sentirse motivados con el PSP y que no lo utilizarían a no ser estrictamente necesario. Como conclusión, los investigadores consideraron vital mantener a los estudiantes enfocados en los aspectos generales del proceso antes que en los detalles del mismo y que la disciplina y la motivación son fundamentales para aprovechar los conceptos del PSP
7. Otro estudio relacionado con la enseñanza del PSP, fue el realizado en la ***Universidad Embry Riddle Aeronautical***(Towhidnejad, Hilburn, & Beach, 1997). Allí, se impartieron

los conceptos de PSP durante dos semestres; en el primer semestre, se cubrió la primera parte del libro de Humprhey de PSP y los estudiantes trabajaron sobre actividades que les permitieran administrar el tiempo y registrarlo en los formularios durante las once semanas del curso. Para ello, se les definieron cuatro actividades: Las clases, lecturas, programación y tiempo dedicado a llenar los formularios. Al final de cada semana, los estudiantes hacían las comparaciones sobre el tiempo que habían estimado para las tareas con respecto al que realmente utilizaron. Durante el segundo semestre, los estudiantes leyeron la segunda parte del libro, recolectaron datos acerca del esfuerzo dedicado en cada una de las seis fases del proceso de desarrollo de software y se concentraron en la administración de defectos del mismo. Se hizo énfasis sobre dónde se inyectaban los defectos y dónde se depuraban. Como conclusiones, el estudio afirma que se requiere más entrenamiento en los estudiantes para que puedan aprovechar mejor los conceptos de PSP, buscar automatizar el llenado de formularios, ya que allí se tuvieron falencias y seguir ajustando las técnicas de enseñanza para hacer los cursos más accesibles a los estudiantes.

8. En la ***Universidad del Quindío***, (Torres, A, Bermúdez, D, &lopez V, 2012) se llevó a cabo una investigación consistente en la construcción de una estrategia de aprendizaje que utilizara un subconjunto de las mejores prácticas de PSP y TSP que permitiera a los estudiantes de primeros semestres de la carrera de Ingeniería de Sistemas y Computación el mejoramiento de su plan de desarrollo de software personal, de sus estimaciones, de la calidad de su trabajo y sus planes de medición. El proyecto consistió en la realización de una investigación de tipo experimental y exploratoria con dos grupos de un primer curso de Programación de Computadores. Uno de ellos, el grupo de control y el otro, el grupo experimental. Al grupo de control se le aplicó el modelo tradicional mientras que el grupo experimental trabajó con la estrategia de aprendizaje propuesta. Se realizó un pretest y un postest con el fin de determinar si la intervención con la estrategia de aprendizaje fue exitosa. Al final, el estudio destaca la apropiación conceptual y práctica de aspectos como la administración y gestión del tiempo, el manejo y gestión de defectos y la planeación de proyectos mejorando con ello en la aplicación de las prácticas de calidad propuestas por PSP.

9. En la investigación realizada por (Fredy, Manrique, & Anaya, 2012), se aborda un estudio experimental en el que se aplicó PSP Nivel 0 en estudiantes de Tecnología en Sistemas de la ***Corporación Universitaria Adventista***. El trabajo pretendía determinar la viabilidad y pertinencia de aplicar el PSP Nivel 0 en estudiantes de Tecnología con el fin de desarrollar competencias básicas y técnicas de gestión de proyectos y calidad. Durante el proyecto, se diseñó y aplicó una prueba piloto a cuatro grupos de diferente nivel de formación; cada grupo realizó dos prácticas acordes con su nivel de formación dentro de la carrera, de las que se recopilaron datos que evidenciaron el aprendizaje obtenido por los estudiantes. Entre los principales hallazgos del estudio estuvieron: La apertura de los estudiantes de primer año en la aplicación de PSP en contraste con la resistencia de los estudiantes de último semestre; La complejidad de la herramienta de registro de tiempos y defectos que genera más trabajo a los estudiantes y desvían su atención del trabajo técnico; La relación positiva encontrada entre la calidad del producto final y la adopción del PSP Nivel 0.
10. En la ***Universidad Autónoma de Yucatán***(Gómez, 2014) se llevó a cabo un estudio en el programa de licenciatura en Ingeniería de Software. El estudio se desarrolló durante todo un semestre y en él abordaron todos los temas y fases del PSP, así como se llevaron a cabo 8 talleres de programación. La investigación se realizó con un grupo de 19 estudiantes de séptimo semestre, de los cuales 14 lo completaron de manera satisfactoria. Como herramienta de soporte se utilizó Proccess Dashboard. Esta investigación tuvo como propósito obtener indicadores de: precisión en las estimaciones de tamaño, esfuerzo, calidad del producto y productividad. Los resultados obtenidos sugirieron una mejora parcial con respecto a la precisión de las estimaciones de tamaño y esfuerzo, una reducción sustantiva en la densidad de defectos, demostrando que el uso de actividades preventivas como las revisiones de diseño y de código demuestran ser efectivas.
11. En el ***Tecnológico de Antioquia***, (Vargas, 2011) se llevó a cabo una investigación para conocer el impacto de la metodología PSP en uno de los espacios académicos relacionados con el área de programación de computadores de la carrera de Ingeniería de

Sistemas. Allí, los investigadores deseaban saber si el uso de las estrategias que provee PSP en Nivel 0 y 0.1 mejorarían la productividad[4] y asertividad[5] en el desarrollo de software por parte de los estudiantes. Se realizaron talleres de programación con dos grupos: uno de 22 estudiantes y uno de 18. Para el primer grupo, se anotaron los tiempos estimados y ejecutados que los estudiantes tardaron en realizar los talleres, así como los defectos obtenidos. Para el segundo grupo, además de lo del primer grupo, se anotaron las líneas de código obtenidas por cada estudiante. Al analizar los resultados, los investigadores concluyeron que las estrategias proporcionadas por PSP generan un mayor grado de productividad y asertividad en el desarrollo de software. Por un lado, la productividad aumenta a medida que el estudiante se incorpora en el ejercicio de las buenas prácticas proporcionadas por PSP y por el conocimiento que tenga de las herramientas de software. La asertividad se incrementa en la medida que los estudiantes van ganando experiencia en la estimación del tiempo a través de los programas que se van realizando.

En conclusión, los reportes de experiencia en el aula, cuando el PSP es usado en un primer curso de Programación, indican la complejidad de su implementación, debido a que los estudiantes, además de aprender a programar, deben aprender las buenas prácticas de desarrollo de software que propone el PSP (Bermón, Fernandez, Sanchez, Javier, & Seco, 2009).

3.2.2. Experiencias académicas con TSP.

En la tabla 7 se hace referencia a experiencias en el aula de clase en las que se experimentó con TSP en algunas instituciones universitarias de pregrado y postgrado obteniendo diversos resultados.

Tabla 7. Experiencias académicas con TSP

Universidad	Estudiantes Objetivo	Nivel de cobertura	Herramientas de soporte de	Estrategia de Aprendizaje

[4] Métrica que se obtiene a partir de la cantidad de líneas de código escritas sobre el tiempo en horas que se tardó el programador en escribirlas

[5] Comparación entre el tiempo estimado y el tiempo real de ejecución de un proyecto de software

			PSP	
Politécnico Universidad de Madrid	Estudiantes de 4 año de pregrado	TSPi	Formularios de TSP	No hace referencia
Politécnico Universidad de Madrid	Estudiantes de 4 año de pregrado	TSP	Formularios de TSP estándar	No hace referencia
Politécnico Universidad de Madrid	Estudiantes de 4 año de pregrado	TSPi	Formularios de TSP	No hace referencia
Loyola University	Estudiantes de Posgrado	Full TSP	Formularios de TSP	No hace referencia
EmbryRiddleAeronauticalUniversity	Estudiantes de pregrado	TSPi	Hoja de cálculo Excel	No hace referencia

A continuación se detallan algunas experiencias académicas del uso de TSP en cursos universitarios.

1. En el ***Politécnico Universidad de Madrid***(Sussy et al., 2008) se llevó a cabo un estudio en el que se les enseñó TSPi a estudiantes de 4 año de carrera universitaria en grupos de trabajo en la escuela de ciencias de la computación. TSPi es una versión académica de TSP utilizada para enseñar a estudiantes de pregrado y postgrado en las Universidades. Inicialmente, se conformaron equipos de 5 o 6 estudiantes, se les capacitó en trabajo en equipo y TSPi; luego, se asignaron los roles en cada equipo y se les asignó un proyecto de desarrollo para que los equipos aplicaran el TSPi en el desarrollo de dicho proyecto. Los estudiantes tenían conocimientos de programación, pero no conocimientos en gestión

de proyectos, administración de bases de datos o trabajo en equipo. Algunas de las experiencias vividas por los equipos fueron: problemas de liderazgo, escaso compromiso, baja calidad del producto desarrollado, problemas con el cronograma, administración inefectiva del tiempo, problemas con los formularios del TSP y problemas con la estimación del tiempo y el tamaño del producto. Con respecto a las mejoras en el desempeño, el proyecto de desarrollo se llevó a cabo en dos ciclos y los resultados con respecto a estimación del tamaño, esfuerzo, productividad, densidad de defectos y estimación de costos entre los dos ciclos fueron alentadores, mientras se logró mayor precisión en la estimación de tamaño y esfuerzo, se logró la disminución de defectos y costos de un ciclo a otro, lo cual redunda en una mayor productividad del trabajo llevado a cabo por los equipos; todo lo anterior, gracias a la aplicación de las técnicas propuestas por TSPi.

2. Otra investigación realizada en el ***Politécnico Universidad de Madrid***(Manzano & Feliu, 2007a), se desarrolló con 22 equipos de estudiantes de cuarto año de la carrera de informática. El estudio trata sobre el impacto positivo de un correcto entrenamiento en las practicas TSP en los estudiantes, sobre la mejora en la estimación, productividad y calidad en el desarrollo de productos software. El procedimiento seguido en esta investigación fue: Se conformaron equipos de 5 o 6 estudiantes; durante el primer ciclo de desarrollo, los estudiantes recibieron entrenamiento en TSP, recibieron el conjunto de especificaciones del cliente, las instrucciones del proceso a seguir, los formularios TSP a utilizar y se inició el proyecto. Durante el segundo ciclo, los estudiantes recibieron un refuerzo en el entrenamiento TSP y entrenamiento en revisiones e inspecciones. Los equipos registraron los valores estimados de tamaño y esfuerzo para los dos ciclos de desarrollo. A lo largo del proyecto registraron los valores reales de tamaño y esfuerzo, permitiendo así la comparación con los estimados en un principio. Varias fueron las conclusiones del estudio: se pudo determinar que a medida que los equipos ganan experiencia en el proyecto, mejoran la estimación; además, la experiencia ganada en el primer ciclo de desarrollo, incrementa la productividad para el segundo ciclo; igual sucede con la calidad del producto, ya que se pudo observar una reducción significativa de los defectos en el producto del primer al segundo ciclo. Esta reducción de defectos se

tradujo en reducción de trabajo y de retrasos en la entrega del producto al cliente, lo cual llevó a una mayor productividad. Así mismo, observó una reducción en los costos del primero al segundo ciclo.

3. Otro estudio realizado en la ***Politécnica Universidad de Madrid*** (Manzano & Feliu, 2007b), se centró en el impacto que tiene el TSPi en los proyectos de desarrollo de software. Se analizaron variables como estimación del tamaño, del esfuerzo, calidad, productividad y costos. En este estudio, se conformaron equipos de entre 5 y 6 estudiantes y se les asignó un rol dentro de los permitidos por TSPi. Los estudiantes recibieron capacitación en TSPi durante el primer ciclo de desarrollo y durante el segundo ciclo se dedicaron a tomar los datos estadísticos relacionados con el desarrollo del proyecto. Al terminar el proyecto, se realizó la comparación entre los datos estimados y los reales y se llegó a la conclusión que el conocimiento y experiencia previa en las prácticas que proporciona TSPi, son elementos que permiten hacer estimaciones con mayor precisión, incrementan la calidad, la productividad y disminuyen los costos.
4. En la ***Universidad de Loyola***(Honig, 2008), se ha enseñado el TSP en cursos de postgrado por varios años. Los estudiantes que participan de los cursos de Ingeniería de software son heterogéneos, con conocimientos de programación y con poca experiencia en la industria. Se usa una versión personalizada del TSP. El proceso es como sigue: Se arman equipos de entre 7 y 12 estudiantes a quienes se les asignan roles de acuerdo con TSP y la experiencia que hayan tenido previamente. Se definen dos ciclos de desarrollo, en los que se debe generar un producto software para un cliente del mundo real quien especifica los requisitos, se reúne periódicamente con el equipo y hace las sugerencias al proceso desde su punto de vista. Igualmente, el instructor hace reuniones semanales con los equipos para verificar el cumplimiento de los objetivos. Los equipos deben recolectar datos sobre tiempo, esfuerzo, defectos y el trabajo en equipo. Al finalizar el curso, se hace el análisis con los datos recogidos del segundo ciclo de desarrollo definido. Se encontró que la productividad de los equipos de estudiantes medida en líneas de código producidas por hora se asemeja a los datos encontrados en la industria. Los 23 equipos conformados por más de 200 estudiantes reconocieron la importancia del TSP en el incremento de la calidad y la productividad de los proyectos de software y redujeron la

tendencia natural de los estudiantes a escribir código ad hoc.

5. En la ***Embry Riddle Areonautical University*** (Tadayon, 2005.) se llevó a cabo una investigación en la que se abordó el TSP con un proyecto real en una asignatura de Ingeniería de Software para estudiantes de pregrado. Los estudiantes a los cuales se orientó el curso fueron estudiantes de segundo año quienes habían tomado previamente cursos de programación de computadores y tenían conocimientos básicos de PSP. El curso estaba conformado por 20 estudiantes armaron equipos de 4 o 5 integrantes donde cada alumno tomaba un rol de acuerdo con los provistos por TSP. Luego de la asignación de roles, se les hizo una declaración de las necesidades del cliente a todos los equipos por igual; estos requisitos fueron dados por el cliente y el docente. Durante el lanzamiento del proyecto, a los estudiantes se les pidió que escribiesen los objetivos individuales y de equipo en un formato especifico y se les solicitó que sostuvieran reuniones semanales. Luego, se le pidió a los equipos un borrador del diseño conceptual del sistema para usarlo en la estimación de tamaño y tiempo; igualmente, los equipos debieron identificar las funcionalidades a desarrollar durante los dos ciclos en los que se dividió el proyecto. Durante el plan del proyecto, los equipos planificaron el trabajo individual y del equipo para el primer ciclo de desarrollo. Luego, los estudiantes pasaron a elaborar el documento de especificación de requisitos, para el que más adelante realizarían una revisión formal. Durante esta fase, los estudiantes también construyeron el plan de pruebas del sistema y el plan de pruebas de integración. Finalmente, los estudiantes pasaron a construir el software; en esta etapa, los equipos estuvieron compitiendo entre ellos entregando virtualmente todas las funcionalidades requeridas.

En su artículo,(Soto & Reyes, 2010) sugieren buscar estrategias en el aula para incorporar los modelos de calidad como PSP, TSP y CMMI que estructuren los conocimientos de los estudiantes y que surtan efectos en las competencias asociadas a las áreas técnicas, alternando desde los primeros semestres prácticas de programación con prácticas de calidad. Sostienen que TSP le da forma a las prácticas de calidad que les proporciona a los estudiantes el PSP, llevándolos a trabajar en equipos de desarrollo tal como sucede en el ámbito empresarial y siguiendo la dinámica que los proyectos reales imponen en el sector.

3.2.3. Experiencias académicas en aprendizaje cooperativo

A continuación, en la tabla 8 se presentan algunas experiencias académicas en las que se trabajó con técnicas de aprendizaje cooperativo.

Tabla 8. Experiencias académicas en Aprendizaje Cooperativo.

UNIVERSIDAD	METODOLOGIA	AUTOR	CANTIDAD ESTUDIANTES	CANTIDAD DE ESTUDIANTES POR GRUPO
Politécnica de Cataluña	Técnica Puzzle	Eliot Aronson	Aula de clase	5 -6 Alumnos
Politécnica de terraza barcelona[1] Europea de madrid[2]	Aplicación aprendizaje basado AC, ABP	D. Garcia[1], B Amante[2]	Aulas de clase	8 - 12 Alumnos
Murcia (España)	A.C en matemáticas	María Elena González Herrero, y José Manuel Serrano	38 – 37 y 27 alumnos	Grupos A,B Y C Respectivamente
Illinois USA	AC en cursos a distancia tecnología educativa	norma Scagnoli	docentes de maestría	Virtuales
Facultad humanidades U Melilan	AC para la enseñanza de la lengua	Fernando Trujillo Saes	Aulas de clase	No indicado

Enseguida se detallan las experiencias académicas con Aprendizaje Cooperativo que se enunciaron en la tabla 8:

1. Investigadores de la Universidad de Cataluña (Anguas et al., 2005) usaron la técnica del Puzzle propuesta por el profesor Eliot Aronson de la Universidad Austin Texas en 1971. El objetivo era determinar si esta técnica podría servir para que los estudiantes se apropiaran adecuadamente de los conceptos trabajados en las materias en la carrera de programación de computadores. Se utilizó puzzle como herramienta de soporte para asignaturas dedicadas a la enseñanza de programación de ordenadores. El proceso de la investigación fue así: se dividen los estudiantes de la clase en grupos de 6 alumnos, se escoge un moderador, por grupo. Se dividen los conceptos en 5 o 6 partes, se asigna a cada alumno un tema y se da el tiempo necesario para que el alumno lea su tema, luego, se forman grupos temporales de alumnos expertos en cada tema, se reúnen los alumnos nuevamente en los grupos originales. Cada alumno debe explicar el tema a sus compañeros, el profesor debe ir de grupo en grupo para observar el proceso, al final de la clase se debe hacer algún tipo de prueba del material, test, cuestionario. Posteriormente, se propone la aplicación de la técnica Puzzle en otros espacios académicos y se describen algunas ventajas y desventajas de la técnica. Por último, se describe cómo usar el aprendizaje cooperativo como una herramienta de soporte al aprendizaje basado en proyectos, afirmando que la inmensa mayoría de los alumnos que han seguido el plan previsto han conseguido los objetivos de aprendizaje y reconocen que han aprendido.

2. Profesores de la Universidad Europea de Madrid en asocio con profesores de la Universidad de Cataluña (García& Amante, 2004) mostraron algunas experiencias de aplicación del aprendizaje cooperativo y la enseñanza por proyectos en ingeniería. Los Autores organizaron los grupos entre 8 y 12 alumnos, luego organizaron subgrupos, en torno al coordinador de grupo elegido por los alumnos. Se establecieron reuniones periódicas y se hizo seguimiento a los grupos, sobre los proyectos de trabajo, reglas de funcionamiento del grupo, etc. Se dividieron los temas de estudio y se dejó trabajar a cada grupo desarrollando ejercicios en grupos en tiempo limitado, resolviendo uno o dos

ejercicios al inicio, mitad y final de cada una de las clases. Se logró aumentar la asistencia de alumnos en la asignatura de proyectos aplicando el método cooperativo. Se concluye que la aplicación de herramientas de aprendizaje cooperativo y el ABP, es factible en variedad de asignaturas, tanto practicas como de laboratorio, la posibilidad de empleo de entornos de trabajo basados en Web permite minimizar la escaza disponibilidad de trabajo presencial.

3. Investigadores de la Universidad de Murcia (Pons, Elena, José, & Serrano, 2008) utilizaron el aprendizaje cooperativo en matemáticas. Este trabajo estudia los efectos de una metodología cooperativa en el aula de matemáticas, concentrándose en la interacción entre tratamiento y contenido. Los autores parten de los resultados que obtienen de otros investigadores, que defienden la idea de que los métodos de aprendizaje cooperativo son más efectivos en tareas complejas. El estudio plantea dos características fundamentales de las matemáticas para el proceso enseñanza aprendizaje: la primera de ellas, la naturaleza de las matemáticas; la segunda, el papel desempeñado por el rol del alumno. El proceso fue como se describe a continuación: Se obtuvo una muestra formada por 102 estudiantes de tres grupos de una escuela de secundaria, los estudiantes de los grupos, A y C formaron el grupo experimental, los alumnos del grupo B conformaron el grupo control. El grupo experimental fue sometido a un proceso de enseñanza aprendizaje basado en una metodología mixta (cooperativo-individualista). Se fueron exponiendo los temas del curso tanto al grupo experimental como al grupo control, a este último, a través de la metodología tradicional. Los alumnos del grupo experimental a su vez, se dividieron en grupos de 5 o 6 y se les propuso el trabajo cooperativo para que todos los miembros del grupo pudieran aprender las lecciones. Los estudiantes del grupo control trabajaron individualmente en el aprendizaje de las lecciones. Se encontraron diferencias significativas en rendimiento entre los grupos experimental y de control (a favor del experimental), lo que confirmó la hipótesis de trabajo que intuyó el beneficio del aprendizaje cooperativo para el aprendizaje de las lecciones de matemáticas propuestas.

4. En la universidad de Illinois (Scagnoli, 2005) se realizó un estudio que tuvo como propósito describir la manera en que pueden ser utilizados el aprendizaje cooperativo y las diversas herramientas que proporciona Internet para la enseñanza en educación a distancia. La investigación se enfocó en las actividades y estrategias de trabajo propuestas a los estudiantes, que resultaron como incentivos para el aprendizaje cooperativo entre participantes de los cursos. El uso de tales estrategias cumplió un doble fin, por un lado enseñaron a los participantes a explorar, contribuir y aprender participando en equipos virtuales, y por otro lado, los prepararon para que inicien o contribuyan a comunidades de aprendizaje virtuales dentro de sus profesiones o siguiendo sus intereses individuales. El estudio llega a la conclusión de que alentar el trabajo cooperativo potencia en los estudiantes la motivación hacia el aprendizaje y el uso de tecnologías de la información, en particular el Internet, facilitan el acceso a los recursos de aprendizaje y la comunicación entre los estudiantes.

5. En la Universidad de Ceuta (Sáez, 2002) realizan una reflexión sobre el papel del aprendizaje cooperativo en el aprendizaje de la lengua. Consideran allí que el aprendizaje cooperativo resulta muy útil para la didáctica de la lengua así como para desarrollar las competencias interculturales necesarias en los profesionales de hoy, pues para desarrollar estas competencias, es necesaria la "interdependencia positiva" y la "responsabilidad del grupo hacia el individuo y del individuo hacia el grupo" mencionados como rasgos propios del aprendizaje cooperativo. Concluyen, que para que el aprendizaje cooperativo sea una metodología efectiva, se requiere de cambios en la forma tradicional de enseñar la lengua que se encuentra fuertemente apegada al paradigma formal de la educación.

4. PLANTEAMIENTO METODOLÓGICO

4.1. Tipo de investigación.

Debido a que este proyecto se enfoca en hacer uso del aprendizaje cooperativo y abordar con ello el estudio de prácticas de PSP y TSP en un curso de programación de computadores, se realizó una investigación de tipo cuasi-experimental. Un método cuasi-experimental es una derivación de los estudios experimentales, en los cuales la asignación de los participantes no es aleatoria aunque el factor de exposición es manipulado por el investigador, es útil para estudiar problemas en los cuales no se puede tener control absoluto de las situaciones, pero se pretende tener el mayor control posible, aun cuando se estén usando grupos ya formados. Algunas de las técnicas mediante las cuales se puede recopilar información en un estudio cuasi-experimental son las pruebas estandarizadas, las entrevistas, las observaciones, etc. (Segura, 2003). Aquí se hace uso del aprendizaje cooperativo como grupo ya formado y se establece uno de control para luego compararlos. El objetivo con dichos grupos es determinar al final las diferencias en cuanto a los efectos que se producen si se aplica o no dicho aprendizaje.

Por otro lado, hacer uso de un método cuasi-experimental permite que se incluya un grupo de comparación que no reciba la intervención y que se evalúa también, antes y después con el fin de medir otras variables externas que cambien el efecto esperado por razones distintas a la intervención. Por esta razón se recurre a la aplicación de medidas pretest y postest para la recolección de los diferentes datos a analizar en los grupos mencionados.

4.2. Propuesta metodológica.

El desarrollo de esta investigación está orientado a determinar si la implementación de técnicas de aprendizaje cooperativo, contribuye al aprendizaje de prácticas de PSP y TSP en estudiantes de segundo semestre de Ingeniería de Sistemas y Computación de la Universidad del Quindío.

Los pasos a seguir para la actividad investigativa, son los siguientes:

- Definición de las variables de investigación.
- Definición del grupo experimental y del grupo control.
- Descripción de los instrumentos de recolección de información.
- Selección de la técnica de Aprendizaje Cooperativo.
- Aplicar al grupo control y al grupo experimental un pre-test para conocer el nivel de apropiación de prácticas de PSP que ya tenían los estudiantes. Luego, se verifica la homogeneidad de los grupos.
- Desarrollar en el grupo control los temas de la asignatura, siguiendo el modelo tradicional, y desarrollar los contenidos propuestos de PSP y TSP en el grupo experimental a través de la estrategia de aprendizaje cooperativo seleccionada.
- Hacer seguimiento continuo de los avances y logros alcanzados por cada uno de los implicados en el proyecto (profesores, estudiantes).
- Aplicar nuevamente el instrumento (Postest), con el fin de identificar las variaciones en los estudiantes luego de la exposición a los temas estudiados con la técnica de aprendizaje.
- Comparar los resultados obtenidos en la prueba diagnóstica con los resultados finales reportados por los estudiantes en el postest.
- Realizar un análisis de los resultados.

Se pueden observar gráficamente los pasos de la propuesta metodológica en la figura 5.

Figura 5. Metodología de desarrollo del proyecto (Cardona y Rincón 2012).

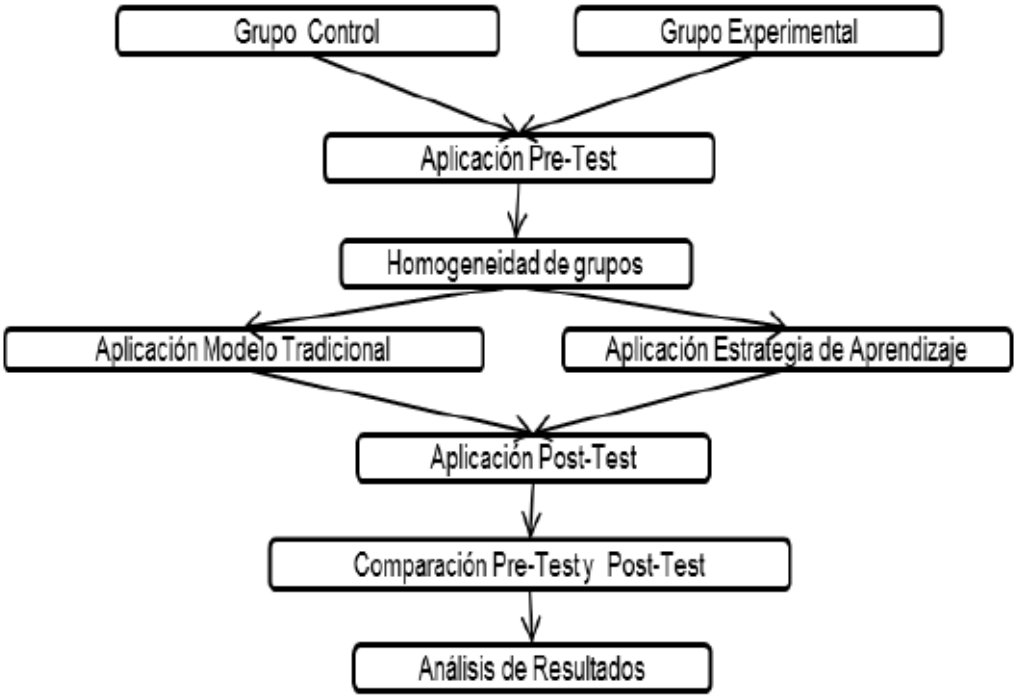


Figura 13. Metodología de desarrollo del proyecto. (Cardona & Rincón, 2012)

4.2.1. Definición de las variables de investigación

En la realización de esta investigación se encontraron dos tipos de variables: independientes y dependientes. Las variables dependientes son las que determinan el objeto de estudio y se definen como aquellas características en los sujetos que se verán modificadas por efecto de la variable independiente (Barbosa, 2010). Las variables independientes son las manipuladas por el investigador y producen un efecto sobre las variables dependientes.

La variable dependiente corresponde al aprendizaje de las practicas de PSP y TSP, la cual es medida a partir de un instrumento de recolección de información (Pretest y PostTest). La variable independiente corresponde a la implementación de técnicas de aprendizaje cooperativo.

Luego de identificar las variables de la investigación, se llevó a cabo el proceso de operacionalización, proceso en el que se detallan las características que permitieron medir las variables. En la tabla 9 se muestran las variables independientes y dependientes con un indicador de medida.

Tabla 9. Variables de la investigación.

Variable	Tipo	Operacionalización
Estrategia de aprendizaje basada de una técnica de aprendizaje cooperativo.	Independiente	Implementación de la técnica de aprendizaje cooperativo JIGSAW en un curso de segundo semestre de Ingeniería de Sistemas y Computación de la Universidad del Quindío
Nivel de aprendizaje de prácticas de PSP y TSP por parte de los estudiantes, basado en Notas y opiniones.	Dependiente	Calificaciones finales obtenidas por los estudiantes. Resultado del Pretest y del Postest hecho a los estudiantes.

4.2.2. Descripción del Grupo experimental y control.

La población objeto de estudio estuvo compuesta por dos grupos de segundo semestre del programa de ingeniería de sistemas y computación de la Universidad del Quindío, que cursaron el espacio académico de Fundamentos de Algoritmia en el área de programación de computadores. La cantidad de estudiantes participantes fueron 13 del grupo experimental y 35 del grupo de control para un total de 48, quienes conformaban el total de la población estudiantil de segundo semestre de la carrera de Ingeniería de Sistemas y Computación en la Universidad del Quindío para el primer semestre del año 2015, el cual fue el momento de la aplicación del instrumento.

De los 48 estudiantes que participaron en la investigación, 34 son de sexo masculino, lo que

equivale al 70,8% y 14 corresponden a estudiantes de sexo femenino que es el 29,2%. Se pudo observar también, que las edades de los alumnos oscilan entre 18 y 26 años, con una media de 19,8 años. La edad que más se repite es de 19 años en 21 estudiantes que equivalen al 43,75%

4.2.3. Descripción de los instrumentos de recolección de información.

Los instrumentos que se utilizaron para la recolección de la información corresponden al Pretest y al Postest, los cuales están conformados por una serie de preguntas cerradas ordenadas por categorías que se aplicaron en momentos diferentes del proceso investigativo: El Pretest al incio y el Postest 8 semanas después. Esto instrumentos fueron analizados por 5 docentes del programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío, quienes hicieron algunas observaciones y recomendaciones tenidas en cuenta para la redacción final del instrumento. Las observaciones hechas por los docentes giraron en torno al orden y redacción de las preguntas. El instrumento utilizado consistió en una encuesta que contenía 29 preguntas en una escala de Likert de 1 a 5.

Categorías e indicadores del Pre y Postest

Tanto las preguntas del Pretest como las del Postest han sido ubicadas en categorías. Para encontrar los indicadores correspondientes a la implementación de PSP y TSP a través de la estrategia de aprendizaje cooperativo, se recurre al uso de preguntas que por medio del Pretest y Postest clasifican a los estudiantes que conforman los dos grupos objeto de observación: grupo experimental y de control. Para ello se proponen 6 categorías de preguntas que agrupan indicadores donde se resalta la presencia de las variables analizadas como lo son el tiempo de desarrollo, el tamaño del programa en líneas de código y la cantidad de errores o defectos encontrados en el desarrollo del mismo. Estas categorías son:

1. Gestión del tiempo usado en un proyecto.
2. Tamaño del proyecto desarrollado.
3. Manejo de defectos

4. Proceso de desarrollo del proyecto de software
5. Trabajo en equipo para el desarrollo de software
6. Uso de técnicas de aprendizaje cooperativo

A continuación se describen los indicadores presentes en cada una de las categorías:

1. Gestión del tiempo usado en un proyecto.

Se sabe que en el desarrollo de todo proyecto el equipo de trabajo y sus integrantes deben gestionar adecuadamente su tiempo, por esta razón se hace uso de los indicadores presentes en la Tabla 10.

Tabla 80. Indicadores para la gestión del tiempo usados en un proyecto.

CATEGORÍA	INDICADOR
1. Gestión del tiempo usado en un proyecto	1.1.Se registra el tiempo que invierte en el desarrollo de un programa de computador.
	1.2.Se registra el tiempo de las interrupciones en el trabajo de desarrollo de software cuando estas se presentan.
	1.3.Se estima el tiempo que debe tardar el desarrollo de un programa de computador.
	1.4.Se calcula el tiempo total que tarda en realizar un programa de computador.

2. Tamaño del proyecto desarrollado.

En la Tabla 11 se pueden apreciar los indicadores usados en esta categoría.

Tabla 11. Indicadores para el tamaño de un proyecto de desarrollo de software.

CATEGORÍA	INDICADOR
2. Tamaño del proyecto	2.1.Se estima el tamaño de un programa medido en la

desarrollado.	cantidad de líneas de código, antes de empezar a construirlo.
	2.2.Se determina el número de líneas de código por hora cuando se escribe un programa de computador.
	2.3.Se planea el número de líneas de código que va a escribir por hora en un programa de computador.

3. **Manejo de defectos.**

Esta categoría pretende caracterizar la correcta identificación y gestión de los defectos que se introducen durante un proceso de desarrollo de software. En la Tabla 12 se pueden apreciar los indicadores usados.

Tabla 12. Indicadores para el manejo de defectos.

CATEGORÍA	INDICADOR
3. Manejo de defectos	3.1.Se registra los defectos que ha introducido al escribir un programa de computador.
	3.2.Se entiende los defectos que ha introducido al escribir un programa de computador.
	3.3.Se clasifican los defectos que va identificando el programador cuando se encuentra desarrollando software.
	3.4.Se cuenta la cantidad de defectos que se encuentran al desarrollar un software.
	3.5.Se sigue un método definido para encontrar y corregir defectos.

4. Proceso de desarrollo en un proyecto de software.

Se pretende identificar la forma como se lleva a cabo el desarrollo de un proyecto de software haciendo uso de los indicadores contenidos en la Tabla 13.

Tabla 93. Indicadores usados en el proceso de desarrollo de software.

CATEGORÍA	INDICADOR
4. Proceso de desarrollo del proyecto de software	4.1. Se planifican las actividades con las que se va a desarrollar un programa de computador.
	4.2. Se conoce y aplica las fases del proceso de desarrollo de software cuando se realiza un programa software.
	4.3. Se utilizan diagramas y/o esquemas para ayudar la etapa de programación del software.
	4.4. Se reúne información de los proyectos de software (como datos de tiempo tardado, el tamaño en líneas de código del programa realizado y la cantidad de errores inyectados y corregidos) ya realizados, para la planeación de proyectos futuros.
	4.5. Se conocen y aplican estándares de codificación.
	4.6. Se documentan los problemas encontrados en el proyecto para corregirlos en futuros trabajos.

5. Trabajo en equipo para el desarrollo de software.

Esta categoría permite definir una serie de indicadores relacionados con el trabajo en equipo evaluando características como la asignación de roles en el desarrollo del proyecto, conocimientos previos de los integrantes y toma de decisiones. En la Tabla 14 se pueden observar estos indicadores.

Tabla 104. Indicadores usados para el trabajo en equipo en el desarrollo de software.

CATEGORÍA	INDICADOR
5. Trabajo en equipo para el desarrollo de software	5.1. Se identifica en los integrantes del equipo conocimientos, aptitudes y habilidades, para asignar las actividades de desarrollo de software.

	5.2. Se definen roles y responsabilidades para cada miembro del equipo.
	5.3. Se observa como las decisiones relacionadas con las tareas de programación, se toman teniendo en cuenta las opiniones de todos los integrantes del equipo.
	5.4. Se planea ordenadamente en equipo la realización de un trabajo o proyecto de desarrollo de software.
	5.5. Se realizan revisiones en equipo sobre el software desarrollado.
	5.6. Se registran los defectos encontrados al trabajar en equipo.

6. Uso de técnicas de aprendizaje Cooperativo.

Esta categoría permite, por medio de diferentes indicadores, medir que tanto se conoce o maneja el concepto de aprendizaje cooperativo en el aprendizaje de algún tema o proyecto. (ver Tabla 15).

Tabla 115. Indicadores para la gestión de técnicas de aprendizaje.

CATEGORÍA	INDICADORES
6. Uso de técnicas de aprendizaje	6.1. Se mide si alguna vez un profesor ha explicado el concepto de aprendizaje cooperativo.
	6.2. Se observa si es posible que el estudiante logre definir lo que es el aprendizaje cooperativo.
	6.3. Se determina si se ha utilizado alguna vez el aprendizaje cooperativo para aprender algún tema.
	6.4. Se establece si se ha explicado alguna vez las diferentes técnicas de aprendizaje cooperativo.
	6.5. Se fija si se ha explicado alguna vez la metodología de trabajo que sigue el aprendizaje cooperativo.

4.2.4. Selección de la técnica de aprendizaje cooperativo.

Para la selección de técnica de aprendizaje cooperativo se tuvieron en cuenta los criterios mencionados por (Jaramillo, 2012), los cuales pretenden que se haga una selección consciente de la técnica de aprendizaje cooperativo que mejor se acomoda a las necesidades de estudiantes y profesor en términos de la actividad que se va a desarrollar. Los criterios buscan facilitar la labor de evaluación y selección de la técnica de aprendizaje cooperativo por parte de los evaluadores.

La calificación fue dada por cinco licenciados expertos en pedagogía, docentes de la universidad del Quindío, quienes a través de una encuesta evaluaron y seleccionaron la técnica de Aprendizaje Cooperativo que mejor se acomoda a las necesidades del proyecto. La encuesta se puede observar en el Anexo al final del documento.

La escala de calificación fue como aparece en la tabla 16:

Tabla 126. Escala de clasificación de técnicas de aprendizaje.

ESCALA DE CALIFICACION	DESCRIPCION DE LA ESCALA DE EVALUACION
5	La técnica de Aprendizaje Cooperativo (A.C.) evaluada cumple plenamente con el criterio de evaluación.
4	La técnica de A.C. cumple en alto grado con el criterio de evaluación.
3	La técnica de A.C evaluada cumple medianamente con el criterio de evaluación.
2	La técnica de A.C cumple pobremente con el criterio de evaluación.
1	La técnica de A.C. no cumple con el criterio de evaluación.

Criterios de Evaluación.

1. Grado de estructuración de la técnica: Se refiere al grado de desarrollo de la técnica y la estructura que la compone. Define si la técnica posee una estructura clara que permita a docente y estudiantes identificar las actividades a realizar y la forma de llevarlas a cabo.
2. Rol del docente: Se refiere a si se encuentra especificado en la técnica el papel que debe cumplir el docente y si este papel es el más adecuado para la actividad a realizar.
3. Rol de los estudiantes: Se refiere a si se encuentra especificado en la técnica el papel que deben cumplir los estudiantes y si este papel es el más adecuado para la actividad a realizar.
4. Número de estudiantes de cada grupo: Se refiere a si la técnica permite conformar grupos de trabajo de estudiantes con la cantidad que requiere la actividad
5. Aporte al estudiante según la taxonomía de Bloom: De acuerdo con (Jaramillo, 2012), cada técnica de aprendizaje cooperativo realiza un aporte mayor o menor al aprendizaje de los estudiantes conforme a lo especificado por Bloom, indicando los conocimientos que los alumnos pueden alcanzar sobre un dominio o tema particular.

Una vez hecha la evaluación de las técnicas de aprendizaje cooperativo, con los criterios de evaluación establecidos por (Jaramillo, 2012) en la tabla 17, se procedió a sumar las calificaciones dadas a cada técnica por parte de los encuestados.

Tabla 137. Evaluación de técnicas de aprendizaje cooperativo.

Técnica de Aprendizaje cooperativo	**Criterios de Evaluación**					**Total**
	Grado de estructuración	**Rol del Docente**	**Rol de los estudiantes**	**Número de estudiantes**	**Aporte según Taxonomía Bloom**	
Aprendiendo juntos	3	3	4	1	2	13
Complex Instruction	3	2	2	1	2	10

Rompecabezas (Jigsaw)	4	4	5	4	3	20

La técnica de aprendizaje cooperativo que cumple la mayoría de criterios plenamente, es la técnica del rompecabezas (Jigsaw), con un puntaje de 20 puntos, superior a las otras, por esta razón se seleccionó para el proceso de investigación.

4.2.5. Definición del contexto.

El programa académico en el que se desarrolla el marco de este proyecto de investigación, es el programa de Ingeniería de Sistemas y Computación perteneciente a la facultad de Ingeniería de la Universidad del Quindío, creado el 12 de agosto de 1996. En promedio el número de estudiantes que ingresa semestralmente al programa es de 80 en la franja diurna y de 80 en la franja nocturna. Para el primer semestre de 2015 se encuentran matriculados 339 estudiantes en la franja diurna y 292 en la nocturna.

En la Tabla 18 se aprecia la información básica del programa de Ingeniería de Sistemas y Computación.

Tabla 18. Información Programa Ingeniería de Sistemas y Computación.

FACULTAD	**INGENIERÍA**
Nombre del programa	Ingeniería de sistemas y computación
Código SNIES	4241
Nivel académico	Pregrado
Nivel de formación	Universitaria
Título	Ingeniero de sistemas y computación
Metodología	Presencial
Franjas	Diurna y nocturna
Número de créditos académicos	178
Departamento	Quindío

Municipio	Armenia
Registro calificado	Resolución 6013 del 1 de junio de 2012 del MEN
Acreditación de alta calidad	Resolución 4626 del 7 de mayo de 2012 del MEN
Año de Iniciación de actividades académicas	1997

El perfil profesional del Ingeniero de Sistemas y Computación de la Universidad del Quindío está centrado en formar profesionales con capacidad para formular, analizar, estructurar y resolver problemas de forma individual e interdisciplinaria, entendiendo las necesidades del entorno y proponiendo soluciones que generen impacto social, mediante la aplicación de las ciencias básicas y las ciencias computacionales.

4.2.6. Aplicación del Aprendizaje Cooperativo en el grupo Experimental.

Una estrategia de aprendizaje contiene los recursos y procedimientos utilizados por el agente de enseñanza para promover aprendizajes significativos (West, Farmer, & Wolff, 1991). Para la experimentación con los estudiantes, se utilizó el aprendizaje cooperativo. Esta estrategia pretendía que los estudiantes cooperaran, a partir de equipos de trabajo, para que cada miembro del equipo aprendiera los contenidos propuestos basándose en el apoyo que sus compañeros le podían prestar. Es importante resaltar que, el profesor buscó hacer responsables a los equipos de trabajo del aprendizaje de cada uno de sus miembros, tratando de afianzar lo que en aprendizaje cooperativo se denomina Interdependencia Positiva; esto para lograr un alto compromiso de los estudiantes hacia el aprendizaje.

El proceso de experimentación se llevó a cabo con el grupo experimental, que estuvo conformado por 13 estudiantes. Las actividades se realizaron durante 8 semanas consecutivas que iniciaron en la quinta semana del primer semestre del año 2015, y terminaron en la semana

12 del mismo semestre. Paralelamente al desarrollo del contenido temático de la asignatura, se fueron incorporando los conceptos y prácticas de los niveles PSP 0 y PSP 0.1 así como los conceptos básicos de TSP. El grupo se dividió en 3 equipos de trabajo de 5, 4 y 4 estudiantes. Los conceptos relacionados con cada uno de los temas se dividieron en el número de integrantes de los equipos de trabajo, conforme a las directrices propuestas por la técnica de aprendizaje cooperativo denominada Rompecabezas o JIGSAW; luego, los estudiantes llevaron a cabo el estudio del tema o temas que le correspondieron de manera individual y después, los alumnos que conformaban los equipos de trabajo se reunieron con sus pares de los otros dos equipos para compartir los conocimientos aprendidos y resolver dudas e inquietudes que se les hubiesen presentado. Posteriormente, cada individuo regresó a su equipo y compartió con sus compañeros los temas de los cuales era responsable. Finalmente, un representante de cada equipo realizó una exposición de lo estudiado por el equipo a todo el grupo de estudiantes que conformaron el grupo experimental y se recibió luego la retroalimentación del profesor; después, los equipos de trabajo se volvían a reunir para hacer una pequeña evaluación a cada uno de los miembros que lo conformaban, con el fin de determinar si se habían alcanzado las metas propuestas en cuanto a la adquisición de los conocimientos respectivos. Esta metodología fue utilizada 3 veces durante las ocho semanas: Una para introducir los conceptos iniciales de PSP, otra para las prácticas habituales de los niveles PSP 0 y PSP 0.1 y la última, para abordar los conceptos principales de TSP.

Para el estudio de los conceptos y prácticas del nivel PSP 0, se diseñó una guía de contenidos, que puede consultarse en el anexo 1, con los fundamentos teóricos necesarios para el aprendizaje y aplicación de las siguientes prácticas:

- Conceptos fundamentales de PSP
- Registro de tiempo de desarrollo medido en minutos que le toma al estudiante realizar el programa propuesto.
- Registro de los defectos y de los tipos de defectos, mediante el estándar de PSP encontrados en las diversas fases del proyecto.
- Resumen del plan de proyecto.

El nivel PSP 0 fue estudiado por los alumnos del curso durante las primeras 2 semanas de la intervención; semanas en las cuales, los estudiantes leyeron y compartieron lo aprendido a través de la técnica JIGSAW.

Igualmente, para el estudio de PSP 0.1, se diseñó una guía que contenía los conceptos fundamentales de este nivel de PSP. Los conceptos de PSP 0.1 fueron abordados durante las semanas 4 y 5. En ellas, los estudiantes volvieron a aplicar la técnica JIGSAW para afrontar los temas de estudio que les propuso la guía. En este nivel, como elemento destacable, los estudiantes debieron utilizar la métrica de líneas de código para determinar el tamaño del programa realizado. En el anexo 2 puede verse la guía de contenidos numero 2.

Los temas finales trabajados con el grupo experimental y relacionados con el proyecto, consistieron en el estudio de las características básicas de TSP, estos se abordaron durante las semanas 7 y 8 de la intervención, a través de una tercera guía con los contenidos planeados; esta guía puede consultarse en el anexo 3. En ella, se abordaron entre otros, los conceptos básicos de lo que es TSP, las practicas más generales de este proceso de desarrollo como las fases, la conformación de los equipos de trabajo y los roles que cumplen los miembros del equipo. Su usaron nuevamente las actividades que propone la técnica de aprendizaje cooperativo Rompecabezas para que los estudiantes interactuaran y aprendieran los temas propuestos en la guía: los temas de la guía fueron divididos para ser estudiados individualmente por los estudiantes; luego, los alumnos se reunían en grupos de trabajo para compartir entre ellos lo aprendido y, por último, un relator de cada grupo exponía a los demás estudiantes del grupo experimental; todo esto, bajo la orientación del profesor.

En la tabla 19 se hace una relación de los conceptos de PSP y TSP abordados, los formatos utilizados y el resultado esperado.

Tabla 19. Conceptos PSP y TSP abordados.

CONTENIDO	FORMATOS PSP INCORPORADOS	ACTIVIDADES DE APRENDIZAJE	RESULTADOS ESPERADOS
- Proceso de Software - Definición de PSP - Principios de PSP - Alcance de PSP - Necesidad de PSP - Etapas de PSP - Gestión de Tiempo. - Gestión de Defectos.	- Log de Registro de tiempos. - Log de registro de defectos. - Resumen del plan del proyecto.	Lecturas individuales. Cooperación con sus pares de otros equipos de trabajo. Socialización de los conceptos aprendidos en cada equipo. Retroalimentación a todo el grupo por parte del profesor. Laboratorio de desarrollo de software	Los estudiantes dominarán los conceptos fundamentales del PSP, sus características y sabrán utilizar los log de manejo de tiempo y defectos.
- Principios de Medición del software. - Estimación del tamaño de un software. - Estimación del tiempo de desarrollo. - Importancia del Diseño. - Conteo de líneas de código.	- Formato PIP. - Estándar de conteo de líneas de código. - Resumen del plan del proyecto.	Lecturas individuales. Cooperación con sus pares de otros equipos de trabajo. Socialización de los conceptos aprendidos en cada equipo. Retroalimentación a todo el grupo por parte del profesor. Laboratorio de desarrollo de software	Los estudiantes aprenderán sobre la estimación de tiempo y tamaño de código, así como diseñarán su propio estándar para conteo de código.
- Descripción del TSP. - La estructura de TSP - La estrategia de desarrollo. - El plan de desarrollo. - Diseño de los equipos.	- Definición de Roles. - Definición de equipos de trabajo.	Lecturas individuales. Cooperación con sus pares de otros equipos de trabajo. Socialización de los conceptos aprendidos en	Los estudiantes se apropiarán de los conceptos fundamentales del TSP: Su estructura, el plan de desarrollo y el diseño de

		cada equipo. Retroalimentación a todo el grupo por parte del profesor. Laboratorio de desarrollo de software	los equipos de trabajo.

En cuanto a las actividades de desarrollo de software, cabe mencionar que se propusieron 2 ejercicios de programación, los cuales fueron resueltos directamente en el laboratorio del curso de manera individual, bajo el seguimiento del profesor. Estos ejercicios de programación se resolvieron individualmente en horas de clase y se utilizó la sala de informática del programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío para la realización de estas actividades de laboratorio. Para los entregables, los estudiantes usaron el log de Registro de tiempos y el log de Registro de defectos para los niveles PSP 0 y PSP 0.1, así como el resumen del plan del proyecto. El formato de manejo de estándares de codificación se usó para el nivel PSP 0.1, el estándar para el conteo de líneas de código y el formato de plan de mejora del proyecto PIP también se usaron en este nivel. Para el proyecto final del curso, se le exigió a los estudiantes la entrega de las plantillas mencionadas. El registro de cada una de las actividades se realizó en las plantillas diseñadas para tal propósito y la retroalimentación de los resultados se hacía en la siguiente clase, resaltando la importancia de las actividades planteadas.

El primer ejercicio de programación consistió en escribir un programa que calculara la media y la desviación estándar de un conjunto de n datos de tipo numérico que debían ser ingresados por el usuario y almacenados en un arreglo de n posiciones. El programa debía ser probado íntegramente con datos proporcionados por el profesor. Este ejercicio se llevó a cabo durante la semana 3 de la intervención al grupo experimental. La primera actividad de Laboratorio tuvo una duración aproximada de 4 horas, que incluyeron el tiempo de programación, pruebas y llenado de los formatos exigidos por PSP para nivel 0.

El segundo ejercicio de programación consistió en la escritura de un programa que almacenara en una matriz o arreglo bidimensional los salarios mensuales pagados a un conjunto de n empleados de una empresa durante un año. El programa debía obtener el salario total pagado en el año a cada trabajador así como el total de la nomina en cada mes del año. El programa debía ser probado con datos proporcionados por el profesor. Este ejercicio se realizó durante la semana 6 de la intervención con el grupo experimental. El segundo laboratorio tuvo una duración de 4 horas que incluyeron el tiempo de programación, pruebas y el llenado de los formatos exigidos por PSP en el nivel 0.1.

En las dos prácticas de laboratorio, los ejercicios de programación propuestos fueron resueltos por los estudiantes utilizando el lenguaje de programación Java, que es el lenguaje visto por los estudiantes en el espacio académico de Fundamentos de Algoritmia en el programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío. Los temas de arreglos uni y bidimensionales ya habían sido tratados con los estudiantes previamente en la materia, por lo cual es pertinente mencionar que los estudiantes ya manejaban los temas de programación necesarios para implementar los programas solicitados. Los estudiantes utilizaron como herramienta para llenar los formatos el PSP Student Workbook, el cual es un aplicativo desarrollado en Microsoft Access que les permitió llenar los registros de tiempo, defectos y el formato de plan de mejora y, automáticamente diligencia el resumen del plan del proyecto.

El plan de evaluación de los aspectos de PSP y TSP estudiados, así como las 2 prácticas de programación tuvo en cuenta aspectos como:

- Observación de las actitudes de los estudiantes.
- Compromiso durante el trabajo en equipo.
- Grado de interés y compromiso con las actividades.
- Tiempo dedicado por los estudiantes.
- Seguimiento al desarrollo de las prácticas que realizan los estudiantes en el laboratorio y durante su trabajo independiente.
- Realización de evaluaciones de forma individual.

Al iniciar la semana 3 de la intervención, los estudiantes fueron evaluados mediante un examen individual de selección múltiple única respuesta que contaba de 10 preguntas. Los temas evaluados en esta actividad fueron los relacionados con la guía # 1. El formato de esta evaluación puede verse en el Anexo 4.

Al principio de la semana 6 de la intervención con el grupo experimental, se llevó a cabo la segunda actividad evaluativa; esta también fue de carácter individual y evaluaba los aspectos relacionados con la guía de trabajo # 2. Al igual que la evaluación # 1, esta constaba de 10 preguntas de selección múltiple única respuesta. El formato de esta evaluación puede verse en el Anexo 5.

Para finalizar el proceso, en la octava semana se llevó a cabo una tercera evaluación individual que pretendía medir el nivel de asimilación de los conceptos trabajados en la guía # 3 por parte de los estudiantes. El formato de esta evaluación puede verse en el Anexo 6.

En total, se obtuvieron 4 notas de cada estudiante, tal como se observa en la tabla 20.

Tabla 20. Actividades de evaluación realizadas.

TEMAS EVALUADOS	ACTIVIDAD EVALUATIVA	DESCRIPCION DE LA EVALUACION
Conceptos de PSP nivel 0	Examen individual	La evaluación constó de 10 preguntas de selección múltiple que se realizó durante los últimos 30 minutos de clase. Cada pregunta tenía un valor de 0.1 para un total de 1.0 que se sumó a la nota del primer parcial de la asignatura de Fundamentos de Algoritmia como bonificación.
Conceptos de PSP nivel 0.1	Examen individual	La evaluación constó de 10 preguntas de selección múltiple que se realizó

		durante los últimos 30 minutos de clase. Cada pregunta tenía un valor de 0.1 para un total de 1.0 que se sumó a la nota del segundo parcial de la asignatura de Fundamentos de Algoritmia como bonificación.
Conceptos de TSP	Examen individual	La evaluación constó de 10 preguntas de selección múltiple que se realizó durante los últimos 30 minutos de clase. Cada pregunta tenía un valor de 0.1 para un total de 1.0 que se sumó a la nota del tercer parcial de la asignatura de Fundamentos de Algoritmia como bonificación.
Laboratorios de Programación	Informes y formatos diligenciados.	La entrega oportuna de los informes de las actividades de laboratorio y los formatos diligenciados permitían obtener 1.0 de bonificación acumulable para el trabajo final del espacio académico de Fundamentos de Algoritmia.

5. ANALISIS DE RESULTADOS.

En esta sección del documento se presentan los resultados obtenidos con la aplicación de los instrumentos de Pretest y Postest. Inicialmente, se hace un análisis descriptivo con los resultados de la aplicación de los instrumentos. Una vez que se ha mostrado la existencia de homogeneidad entre los grupos, se analizaron las 29 preguntas, una a una, revisando las respuestas obtenidas con todos los estudiantes para luego, pasar a analizar el comportamiento de los grupos Control y Experimental durante el Pre y Postest y las diferencias observadas en ambos grupos con la aplicación del Postest, al final del proceso investigativo.

Para determinar la homogeneidad de los grupos de control y experimental se aplicó un instrumento de Pretest, que permitió identificar el nivel de adopción de prácticas individuales de desarrollo por parte de los estudiantes de ambos grupos: Experimental y Control. Al finalizar la intervención con los estudiantes del grupo experimental, se aplicó el Postest con el fin de comparar la información obtenida con estos dos instrumentos. El Postest fue aplicado en la octava semana después de haber aplicado el Pretest, tiempo durante el cual, los alumnos del grupo experimental aplicaron las estrategias del aprendizaje cooperativo en el estudio de las prácticas y métricas de PSP y TSP.

5.1. Análisis del Pretest.

Se realizó el Pretest con los estudiantes tanto del grupo experimental como del grupo control, con el fin de determinar la homogeneidad. Para determinar dicha homogeneidad se utilizó la técnica estadística de Análisis de la Varianza (ANOVA) con la calificación dada por el estudiante a cada pregunta como variable de respuesta y los grupos experimental y de control como factor.

A continuación, en la tabla 21 se puede observar el resultado de homogeneidad para cada pregunta que conformaba el instrumento de Pretest.

Tabla 21. Homogeneidad en el Pretest.

Categoría de Pregunta	Pregunta	Estadístico	P Valor	Se aprueba la Hipótesis de Homogeneidad
1. Grupo de preguntas en **relación** con la **gestión** del tiempo en los proyectos	1.1.	0,00059387	0,980558	Si
	1.2.	0,40222	0,525945	Si
	1.3.	0,165734	0,683932	Si
	1.4.	2,90037	0,0885556	Si
2. Grupo de preguntas en relación con el tamaño (cantidad de líneas de código generadas en la construcción de un programa de software).	2.1.	3,51405	0,0608468	Si
	2.2.	2,25643	0,133057	Si
	2.3.	0,0640174	0,800255	Si
3. Grupo de preguntas en relación con el manejo de defectos(Cualquier cosa que deba de cambiarse en el software fruto de una mala implementación) en los programas de software	3.1.	0,276553	0,598969	Si
	3.2.	2,592	0,107401	Si
	3.3.	0,113643	0,736034	Si
	3.4.	0,179966	0,671402	Si
	3.5.	0,353594	0,552085	Si
4. Grupo de preguntas en relación con el proceso de desarrollo de los proyectos de software	4.1.	2,7616	0,0965477	Si
	4.2.	2,49923	0,113898	Si
	4.3.	3,29532	0,0694737	Si
	4.4.	0,69455	0,40462	Si
	4.5.	0,0188095	0,890914	Si
	4.6.	0,00363529	0,951922	Si
5. Grupo de preguntas en relación a la organización en equipo para llevar a cabo trabajos o proyectos de	5.1.	0,540474	0,462235	Si
	5.2.	0,283956	0,59412	Si
	5.3.	1,28192	0,257539	Si
	5.4.	0,0188095	0,890914	Si

desarrollo de software.	5.5.	0,0184554	0,891939	Si
	5.6.	0,624511	0,429375	Si
6. Grupo de preguntas en relación al uso de técnicas de aprendizaje cooperativo	6.1.	1,00128	0,316999	Si
	6.2.	0,379967	0,53762	Si
	6.3.	2,51546	0,112731	Si
	6.4.	3,0972	0,0784234	Si
	6.5.	0,933054	0,334069	Si

En la tabla anterior se puede observar que no hubo diferencias significativas entre las respuestas dadas a las preguntas por los grupos Control y Experimental. Se puede concluir que existe homogeneidad en los conocimientos entre los estudiantes que respondieron a las preguntas: Los 35 estudiantes del grupo Control y los 13 estudiantes del grupo experimental. Esto es lógico, ya que los alumnos no han estudiado temas relacionados con PSP o TSP en el tiempo que llevan en la universidad y posiblemente tampoco lo han estudiado en otras instancias académicas. Esto permite dar inicio a la intervención con los estudiantes del grupo experimental.

A continuación se describe en la tabla 22 las respuestas obtenidas de los estudiantes durante el Pretest.

Tabla 22. Resultados estadísticos del Pretest.

Pregunta del Pretest	Porcentaje de estudiantes que respondieron						
	1	2	3	4	5	Me.	Des.
1.1. Registra el tiempo que invierte en el desarrollo de un programa de computador?	33.3	35.4	22.9	8.3	0.0	2.06	0.94
1.2. Registra el tiempo de las interrupciones en el trabajo de desarrollo de software cuando estas se presentan?	50	33.3	12.5	2.1	2.1	1.73	0.91

1.3. Estima el tiempo que debe tardar el desarrollo de un programa de computador?	27.1	18.8	31.3	16.7	6.3	2.56	1.22
1.4. Calcula el tiempo total que tardó en realizar un programa de computador?	27.1	18.8	20.8	25.0	8.3	2.69	1.33
2.1. Estima el tamaño de un programa medido en la cantidad de líneas de código, antes de empezar a construirlo?	52.1	25.0	16.7	2.1	4.2	1.81	1.05
2.2. Determina el número de líneas de código por hora cuando escribe un programa de computador?	54.2	33.3	12.5	0.0	0.0	1.58	0.70
2.3. Planea el número de líneas de código que va a escribir por hora en un programa de computador?	60.4	27.1	10.4	2.1	0.0	1.54	0.76
3.1. Registra los defectos que ha introducido al escribir un programa de computador?	22.9	18.8	20.8	29.2	8.3	2.81	1.30
3.2. Entiende los defectos que ha introducido al escribir un programa de computador?	6.3	10.4	29.2	45.8	8.3	3.40	0.99
3.3. Clasifica los defectos que va identificando, cuando se encuentra desarrollando software?	22.9	20.8	35.4	18.8	2.1	2.56	1.10
3.4. Cuenta la cantidad de defectos que encuentra al desarrollar un software?	35.4	27.1	27.1	4.2	6.3	2.19	1.15
3.5. Sigue un método definido para encontrar y corregir defectos?	16.7	25.0	22.9	25.0	10.4	2.88	1.25
4.1. Planifica las actividades con las que va a desarrollar un programa de computador?	12.5	16.7	22.9	37.5	10.4	3.17	1.20
4.2. Conoce y aplica las fases del proceso de desarrollo de software cuando va a realizar un programa software?	4.2	18.8	37.5	31.3	8.3	3.21	0.98

4.3. Utiliza diagramas y/o esquemas para ayudarse en la etapa de programación del software?	18.8	25	20.8	29.2	6.3	2.79	1.22
4.4. Reúne información de los proyectos de software (como datos de tiempo tardado, el tamaño en líneas de código del programa realizado y la cantidad de errores inyectados y corregidos) ya realizados para la planeación de proyectos futuros?	43.8	35.4	12.5	8.3	0.0	1.85	0.94
4.5. Conoce y aplica estándares de codificación?	10.4	43.8	20.8	18.8	6.3	2.67	1.09
4.6. Documenta los problemas encontrados en el proyecto para corregirlos en futuros trabajos?	31.3	31.3	22.9	12.5	2.1	2.23	1.08
5.1. Se identifica en los integrantes del equipo conocimientos, aptitudes y habilidades, para asignar las actividades de desarrollo de software?	14.6	6.3	25.0	31.3	22.9	3.42	1.30
5.2. Se definen roles y responsabilidades para cada miembro del equipo?	4.2	20.8	14.6	47.9	12.5	3.44	1.08
5.3. Las decisiones relacionadas con las tareas de programación, se toman teniendo en cuenta las opiniones de todos los integrantes del equipo?	4.2	12.5	20.8	29.2	33.3	3.75	1.16
5.4. Se planea ordenadamente en equipo la realización de un trabajo o proyecto de desarrollo de software?	4.2	10.4	25.0	43.8	16.7	3.58	1.02
5.5. Se realizan revisiones en equipo sobre el software desarrollado?	8.3	6.3	29.2	37.5	18.8	3.52	1.12
5.6. Cuando trabaja en equipo, se registran los defectos encontrados?	27.1	16.7	31.3	18.8	6.3	2.60	1.24
6.1. Alguna vez un profesor le ha explicado el concepto de aprendizaje cooperativo?	70.8	14.6	12.5	0.0	2.1	1.48	0.87

6.2. Podría usted definir lo que es el aprendizaje cooperativo?	62.5	12.5	16.7	6.3	2.1	1.73	1.08
6.3. Ha utilizado alguna vez el aprendizaje cooperativo para aprender algún tema?	60.4	16.7	16.7	6.3	0.0	1.69	0.96
6.4. Le han explicado alguna vez las diferentes técnicas de aprendizaje cooperativo?	83.3	8.3	6.3	2.1	0.0	1.27	0.67
6.5. Le han explicado alguna vez la metodología de trabajo que sigue el aprendizaje cooperativo?	79.2	12.5	4.2	2.1	2.1	1.35	0.83

De la anterior tabla se puede concluir que:

- La gran mayoría de los estudiantes manifiestan que no gestionan el tiempo usado en un proyecto, tanto el hecho de registrar los tiempos que tardan en actividades como el desarrollo del proyecto o en la fase de implementación del mismo; los estudiantes tampoco realizan estimaciones sobre el tiempo que pueden tardar desarrollando un proyecto de software. Se nota que los estudiantes solo se dedican a escribir el programa sin analizar cuanto podrían tardar en hacerlo o en recoger datos estadísticos del tiempo que tardan, para usarlos en futuros proyectos.
- Los estudiantes no hacen estimaciones del tamaño del proyecto de software a desarrollar ni hacen mediciones una vez han concluido la escritura del programa. Parece que los estudiantes no son conscientes de la métrica de las líneas de código, pues no la utilizan en actividades de estimación ni en la recolección de datos y, mucho menos para conocer su productividad.
- En cuanto a defectos se refiere, solo algunos estudiantes registran permanentemente los defectos que van encontrando al escribir código y los clasifican. Así mismo, son muy pocos (10.4%) los que cuentan el total de defectos producidos en un software y siguen un método definido para encontrar y corregir defectos.
- Son muy pocos los estudiantes que hacen planificación del proyecto que van a enfrentar. Algunos de ellos (8.3%) aplican las fases del proceso de desarrollo; esto quiere decir, que la gran mayoría de los estudiantes solo conoce la etapa de implementación y desconocen las

fases de requisitos, análisis y diseño. Son muy pocos los que documentan el proyecto y solo el 6.3% de los encuestados manifiestan usar estándares de escritura de código constantemente.

- En cuanto al trabajo en equipo, parece haber una mayor consciencia sobre las actividades que se deben desarrollar y las decisiones que se toman, ya que alrededor del 30% manifiestan buenas prácticas de trabajo en equipo.
- En general, los estudiantes manifestaron desconocer lo que es el aprendizaje cooperativo, la forma en que este se aplica en el proceso de aprendizaje y afirman no haberlo utilizado.

5.2. Análisis del grupo Control

Se comparó el comportamiento del grupo Control con el propósito de determinar si los estudiantes que lo conformaron presentaban diferencias significativas en su aprendizaje entre las respuestas dadas en el Pretest y en el Postest. En la tabla 23, que se encuentra a continuación, se puede observar el comportamiento del grupo control.

Tabla 23. Análisis del grupo control.

Categoría de Pregunta	Pregunta	Estadístico	P Valor	Se aprueba la Hipótesis de Homogeneidad
1. Grupo de preguntas en relación con la gestión del tiempo en los proyectos	1.1.	0,187471	0,66503	Si
	1.2.	0,138444	0,709833	Si
	1.3.	0,0119925	0,912798	Si
	1.4.	0,0326937	0,856514	Si
2. Grupo de preguntas en relación con el tamaño (cantidad de líneas de código generadas en	2.1.	0,0113033	0,915331	Si
	2.2.	0,0152451	0,901734	Si
	2.3.	0	1,0	Si

la construcción de un programa de software).				
3. Grupo de preguntas en relación con el manejo de defectos(Cualquier cosa que deba de cambiarse en el software fruto de una mala implementación) en los programas de software	3.1.	0	1,0	Si
	3.2.	0	1,0	Si
	3.3.	0	1,0	Si
	3.4.	0	1,0	Si
	3.5.	0	1,0	Si
4. Grupo de preguntas en relación con el proceso de desarrollo de los proyectos de software	4.1.	0	1,0	Si
	4.2.	0	1.0	Si
	4.3.	1,60609	0,20504	Si
	4.4.	0	1,0	Si
	4.5.	0	1,0	Si
	4.6.	0	1,0	Si
5. Grupo de preguntas en relación a la organización en equipo para llevar a cabo trabajos o proyectos de desarrollo de software.	5.1.	0	1,0	Si
	5.2.	0	1.0	Si
	5.3.	0	1,0	Si
	5.4.	0	1,0	Si
	5.5.	0	1,0	Si
	5.6.	0	1,0	Si
6. Grupo de preguntas en relación al uso de técnicas de aprendizaje cooperativo	6.1.	0	1,0	Si
	6.2.	0	1,0	Si
	6.3.	0	1,0	Si
	6.4.	0	1,0	Si
	6.5.	0	1,0	Si

Como se puede observar de la tabla anterior, todos los valores de P_Valor estuvieron por encima de 0.05, lo cual permite concluir que en el grupo de control sigue existiendo

homogeneidad luego de aplicarse el Postest. La homogeneidad existente significa que los estudiantes no se apropiaron de los conocimientos de PSP y TSP y siguen desconociendo las características fundamentales de estos procesos de desarrollo que se estudiaron con los alumnos del grupo experimental. Esto es comprensible, pues con este grupo no se aplicó ninguna estrategia de aprendizaje ni se presentaron los contenidos propuestos de PSP y TSP, como si se hizo con el grupo experimental.

5.3. Análisis del Postest.

La comparación realizada en el Postest pretende identificar diferencias significativas entre los grupos Control y Experimental, como quiera que este último grupo fue al que se le aplicó la estrategia de aprendizaje cooperativo para que los estudiantes se apropiaran de los conceptos, prácticas y métricas de PSP y TSP. Se utilizó igualmente, el análisis de la varianza ANOVA con el fin de determinar si hubo o no homogeneidad entre los dos grupos. En la tabla 24 es posible observar los resultados del Postest.

Tabla 24. Análisis del Postest.

Categoría de Pregunta	Pregunta	Estadístico	P Valor	Se aprueba la Hipótesis de Homogeneidad
1. Grupo de preguntas en **relación** con la **gestión** del tiempo en los proyectos	1.1.	16,5844	0,0000465338	No
	1.2.	15,3003	0,000091703	No
	1.3.	6,90561	0,00859073	No
	1.4.	7,52637	0,0060787	No
2. Grupo de preguntas en relación con el tamaño (cantidad de líneas de código generadas en la construcción de un programa de software).	2.1.	9,31092	0,00227704	No
	2.2.	15,504	0,0000823335	No
	2.3.	13,427	0,000247846	No

3. Grupo de preguntas en relación con el manejo de defectos(Cualquier cosa que deba de cambiarse en el software fruto de una mala implementación) en los programas de software	3.1.	15,388	0,0000875451	No
	3.2.	17,609	0,0000271306	No
	3.3.	22,0102	0,00000271203	No
	3.4.	9,82177	0,00172385	No
	3.5.	12,5034	0,00040595	No
4. Grupo de preguntas en relación con el proceso de desarrollo de los proyectos de software	4.1.	14,6274	0,000130999	No
	4.2.	17,7326	0,0000254237	No
	4.3.	22,1918	0,0000024672	No
	4.4.	21,4705	0,0000035931	No
	4.5.	25,1329	5,3512E-7	No
	4.6.	12,2239	0,000471513	No
5. Grupo de preguntas en relación a la organización en equipo para llevar a cabo trabajos o proyectos de desarrollo de software.	5.1.	10,9092	0,000956362	No
	5.2.	13,5297	0,000234645	No
	5.3.	14,0645	0,000176531	No
	5.4.	19,2689	0,0000113541	No
	5.5.	21,7346	0,0000031309	No
	5.6.	8,42191	0,00370609	No
6. Grupo de preguntas en relación al uso de técnicas de aprendizaje cooperativo	6.1.	24,0935	9,1772E-7	No
	6.2.	14,1643	0,000167411	No
	6.3.	15,9728	0,0000642604	No
	6.4.	15,504	0,0000823335	No
	6.5.	18,9658	0,0000133083	No

Como se observa en la tabla anterior, en ninguna de las preguntas se obtuvo un P_Valor por encima de 0.05, lo cual permite concluir que, al finalizar la investigación, no había homogeneidad en los conocimientos sobre PSP y TSP entre los grupos Control y Experimental. Lo anterior se puede explicar porque los estudiantes que conformaron el grupo Experimental, a diferencia de los que conformaron el grupo Control, estuvieron expuestos a las actividades y estrategias de aprendizaje cooperativo para adquirir conocimientos en los temas tratados. Es

fundamental recalcar que, el aprendizaje cooperativo con la técnica del ROMPECABEZAS, motiva a los estudiantes a trabajar en equipo y a sentirse responsables del aprendizaje de cada uno de los miembros del equipo. Estos estudiantes, además, debieron presentar evaluaciones que median el grado de apropiación de los conceptos expuestos durante la intervención.

5.4. Análisis del grupo Experimental

Por último, se comparó el grupo experimental en su desempeño durante el Pretest y el Postest. Se pretendía determinar si hubo o no homogeneidad en las respuestas que dieron los estudiantes. Se espera que no exista homogeneidad en las dos comparaciones, ya que los estudiantes tuvieron actividades de aprendizaje de los conceptos, prácticas y métricas del PSP y TSP a través del aprendizaje cooperativo y por tanto, debería existir diferencia.

A continuación, en la tabla 25 se puede observar si existe homogeneidad entre lo respondido por los estudiantes del grupo experimental en el Pretest y el Postest.

Tabla 25. Análisis del grupo experimental.

Categoría de Pregunta	Pregunta	Estadístico	P Valor	Se aprueba la Hipótesis de Homogeneidad
1. Grupo de preguntas en **relación** con la **gestión** del tiempo en los proyectos	1.1.	15,3445	0,0000895839	No
	1.2.	13,3856	0,000253378	No
	1.3.	7,22418	0,00719111	No
	1.4.	8,62843	0,00330848	No
2. Grupo de preguntas en relación con el tamaño (cantidad de líneas de código generadas en la construcción de un programa de software).	2.1.	8,62843	0,00330848	No
	2.2.	11,9847	0,000536068	No
	2.3.	9,99139	0,00157205	No
3. Grupo de preguntas en relación	3.1.	9,99109	0,0015723	No

con el manejo de defectos(Cualquier cosa que deba de cambiarse en el software fruto de una mala implementación) en los programas de software	3.2.	4,4934	0,0340227	No
	3.3.	13,3856	0,000253378	No
	3.4.	7,50361	0,006156	No
	3.5.	9,37643	0,00219708	No
4. Grupo de preguntas en relación con el proceso de desarrollo de los proyectos de software	4.1.	5,16046	0,023104	No
	4.2.	4,13945	0,0418913	No
	4.3.	20,3652	0,00000639827	No
	4.4.	16,0286	0,0000623949	No
	4.5.	5,65388	0,0174143	No
	4.6.	8,63422	0,00329797	No
5. Grupo de preguntas en relación a la organización en equipo para llevar a cabo trabajos o proyectos de desarrollo de software.	5.1.	15,4323	0,0000855143	No
	5.2.	8,0481	0,00455383	No
	5.3.	6,65848	0,00986652	No
	5.4.	16,3585	0,0000524221	No
	5.5.	19,909	0,00000812193	No
	5.6.	7,88602	0,00498045	No
6. Grupo de preguntas en relación al uso de técnicas de aprendizaje cooperativo	6.1.	9,4235	0,00214139	No
	6.2.	6,98853	0,00820154	No
	6.3.	15,9035	0,0000666575	No
	6.4.	12,4301	0,00042219	No
	6.5.	7,26323	0,0070364	No

De la tabla anterior, se observa que en todas las preguntas se obtuvo un P_Valor inferior a 0.05, lo cual permite concluir que no hubo homogeneidad entre el Pretest y el Postest realizado al grupo Experimental. Esto debido a la utilización de la estrategia de aprendizaje cooperativo y la exposición que se hizo con este grupo a los temas relacionados con PSP y TSP, puede verse que los estudiantes que conformaron el grupo experimental asimilaron los conceptos a través de la

utilización de la estrategia de aprendizaje cooperativo.

En la tabla 26 que se muestra a continuación, se puede observar el comportamiento estadístico de lo respondido por los estudiantes del grupo experimental durante el Postest.

Tabla 26. Análisis estadístico del grupo experimental.

Pregunta del Postest	Porcentaje de estudiantes que respondieron						
	1	2	3	4	5	Me.	Des.
1.1. Registra el tiempo que invierte en el desarrollo de un programa de computador?	0.0	0.0	61.0	31.0	8.0	3.46	0.63
1.2. Registra el tiempo de las interrupciones en el trabajo de desarrollo de software cuando estas se presentan?	8.0	8.0	38.0	46.0	0.0	3.23	0.89
1.3. Estima el tiempo que debe tardar el desarrollo de un programa de computador?	0.0	0.0	46.0	54.0	0.0	3.54	0.50
1.4. Calcula el tiempo total que tardó en realizar un programa de computador?	7.0	0.0	23.0	62.0	8.0	3.62	0.92
2.1. Estima el tamaño de un programa medido en la cantidad de líneas de código, antes de empezar a construirlo?	8.0	23.0	23.0	46.0	0.0	3.08	1.00
2.2. Determina el número de líneas de código por hora cuando escribe un programa de computador?	8.0	23.0	38.0	31.0	0.0	2.92	0.92
2.3. Planea el número de líneas de código que va a escribir por hora en un programa de computador?	15.0	15.0	23.0	46.0	1.0	3.00	1.11
3.1. Registra los defectos que ha introducido al escribir un programa de computador?	8.0	0.0	23.0	69.0	0.0	3.54	0.84
3.2. Entiende los defectos que ha introducido al escribir	0.0	8.0	23.0	54.0	15.0	3.77	0.80

un programa de computador?							
3.3. Clasifica los defectos que va identificando, cuando se encuentra desarrollando software?	0.0	15.0	38.0	31.0	15.0	3.46	0.93
3.4. Cuenta la cantidad de defectos que encuentra al desarrollar un software?	8.0	15.0	8.0	46.0	23.0	3.62	1.21
3.5. Sigue un método definido para encontrar y corregir defectos?	15.0	8.0	31.0	38.0	8.0	3.15	1.17
4.1. Planifica las actividades con las que va a desarrollar un programa de computador?	8.0	0.0	15.0	69.0	8.0	3.69	0.91
4.2. Conoce y aplica las fases del proceso de desarrollo de software cuando va a realizar un programa software?	0.0	8.0	31.0	46.0	15.0	3.69	0.82
4.3. Utiliza diagramas y/o esquemas para ayudarse en la etapa de programación del software?	0.0	8.0	0.0	77.0	15.0	4.00	0.68
4.4. Reúne información de los proyectos de software (como datos de tiempo tardado, el tamaño en líneas de código del programa realizado y la cantidad de errores inyectados y corregidos) ya realizados para la planeación de proyectos futuros?	0.0	8.0	8.0	77.0	7.0	3.85	0.66
4.5. Conoce y aplica estándares de codificación?	8.0	0.0	38.0	46.0	8.0	3.46	0.93
4.6. Documenta los problemas encontrados en el proyecto para corregirlos en futuros trabajos?	7.0	0.0	23.0	62.0	8.0	3.62	0.92
5.1. Se identifica en los integrantes del equipo conocimientos, aptitudes y habilidades, para asignar las actividades de desarrollo de software?	0.0	7.0	31.0	54.0	8.0	3.62	0.74
5.2. Se definen roles y responsabilidades para cada miembro del equipo?	0.0	8.0	15.0	69.0	8.0	3.77	0.70

5.3. Las decisiones relacionadas con las tareas de programación, se toman teniendo en cuenta las opiniones de todos los integrantes del equipo?	0.0	0.0	23.0	54.0	23.0	4.00	0.68
5.4. Se planea ordenadamente en equipo la realización de un trabajo o proyecto de desarrollo de software?	0.0	8.0	8.0	38.0	46.0	4.23	0.89
5.5. Se realizan revisiones en equipo sobre el software desarrollado?	0.0	8.0	8.0	31.0	54.0	4.31	0.91
5.6. Cuando trabaja en equipo, se registran los defectos encontrados?	8.0	0.0	15.0	54.0	23.0	3.85	1.03
6.1. Alguna vez un profesor le ha explicado el concepto de aprendizaje cooperativo?	7.0	8.0	31.0	31.0	23.0	3.54	1.15
6.2. Podría usted definir lo que es el aprendizaje cooperativo?	15.0	0.0	31.0	46.0	8.0	3.31	1.14
6.3. Ha utilizado alguna vez el aprendizaje cooperativo para aprender algún tema?	7.0	8.0	15.0	62.0	8.0	3.54	1.01
6.4. Le han explicado alguna vez las diferentes técnicas de aprendizaje cooperativo?	15.0	8.0	31.0	38.0	8.0	3.15	1.17
6.5. Le han explicado alguna vez la metodología de trabajo que sigue el aprendizaje cooperativo?	23.0	0.0	23.0	46.0	8.0	3.15	1.29

De la anterior tabla, se pueden hacer las siguientes consideraciones:

- La mayoría de los estudiantes se ha apropiado de las prácticas de PSP relacionadas con el registro de los tiempos de desarrollo de software y con la estimación del tiempo que pueden tardar los proyectos. El 70% de los estudiantes encuestados manifiestan calcular el tiempo que tardan en escribir un programa de computador.
- En lo que tiene que ver con las líneas de código, hay una leve mejoría con respecto a los datos arrojados en el Pretest; sin embargo, se ve que todavía no existe consciencia sobre

la importancia de estimar y medir la cantidad de líneas de código que se van a escribir o recoger datos sobre las que ya se han escrito para usarlas en proyectos posteriores.

- Los estudiantes le han dado mayor importancia a lo relacionado con los defectos: la mayoría de ellos registran los defectos que van generando en la escritura de programas de computador, también los clasifican y los cuentan, para tenerlos en cuenta en futuros proyectos.
- Los estudiantes mostraron mejoría en aspectos relacionados con la planeación de los proyectos de software, manifestaron registrar datos estadísticos para utilizarlos en futuros proyectos, documentan los problemas encontrados y dan mayor importancia a los estándares para la escritura de código.
- También es palpable la mejoría en los aspectos de trabajo en equipo. Los estudiantes afirman definir roles y responsabilidades, así como asignar las tareas de acuerdo con las habilidades de los miembros del equipo. También afirman realizar revisiones en equipos de trabajo sobre el software que están desarrollando y registrar los defectos que van encontrando, de la misma manera.
- La mayoría de los estudiantes reconocen las características principales del aprendizaje cooperativo y lo han utilizado para estudiar los temas de PSP y TSP propuestos durante la intervención.

5.4.1. Análisis de las Notas obtenidas con el grupo Experimental

Como se mencionó en la sección 4.2.6, se sacaron 4 notas con los estudiantes del grupo experimental relacionadas con los temas vistos durante la intervención sobre PSP y TSP. En este apartado, se hace un análisis de dichas notas obtenidas.

La primera nota obtenida fue una bonificación de 1.0 sacada con un examen de 10 preguntas selección múltiple única respuesta. A continuación, en la tabla 27 se tienen los resultados:

Tabla 27. Primera Evaluación realizada

Numero de Estudiante	Bonificación Obtenida
1	0.7
2	0.6
3	0.8
4	1.0
5	0.7
6	0.8
7	0.9
8	0.8
9	1.0
10	0.8
11	0.6
12	0.9
13	0.7

En promedio, los 13 estudiantes del grupo experimental bonificaron 0,79 unidades para el primer parcial, lo cual corresponde a aproximadamente 8 respuestas correctas de las 10 que se les presentaron. Tuvieron una bonificación máxima de 1.0 y una mínima de 0.6, obteniendo una desviación estándar de 0.13.

La segunda nota obtenida fue una bonificación de 1.0 sacada con un examen de 10 preguntas selección múltiple única respuesta. Esta bonificación se sumó a la nota del segundo parcial de la Materia. A continuación, en la tabla 28 se tienen los resultados:

Tabla 28. Segunda Evaluación realizada

Numero de Estudiante	Bonificación Obtenida
1	0.9
2	0.8
3	0.7
4	1.0
5	0.8
6	1.0
7	1.0
8	0.7
9	1.0
10	0.9
11	0.8
12	0.8
13	0.8

Para esta segunda evaluación, los estudiantes obtuvieron en promedio una bonificación de 0.9, lo que corresponde a 9 de 10 preguntas correctas de la evaluación. 3 estudiantes lograron la máxima puntuación obtenida de 1.0 unidad, mientras de la bonificación mínima obtenida por un estudiante fue de 0.7. Lo anterior, permite concluir que los estudiantes, en general, se apropiaron de los conocimientos tratados en la segunda guía didáctica y aprovecharon la bonificación que se les dio para mejorar la nota del segundo examen parcial de la materia.

La tercera nota obtenida fue una bonificación de 1.0 sacada con un examen de 10 preguntas selección múltiple única respuesta. Esta bonificación se sumó a la nota del tercer parcial de la Materia. A continuación, en la tabla 29 se tienen los resultados:

Tabla 29. Tercera Evaluación realizada

Numero de Estudiante	Bonificación Obtenida
1	0.9
2	1.0
3	1.0
4	1.0
5	0.9
6	1.0
7	1.0
8	0.8
9	1.0
10	1.0
11	0.9
12	0.8
13	0.9

En el examen de la tercera bonificación, los estudiantes obtuvieron en promedio 0.9 unidades, nota que permite concluir que los conceptos tratados durante esta fase de la intervención fueron apropiados por los estudiantes. Se tuvo una nota mínima de 0.8, lo cual dice que el estudiante que tuvo el mas mal desempeño acertó a 8 de 10 preguntas que se le presentaron en la evaluación. Siete estudiantes obtuvieron la máxima calificación; esto corresponde a más del 50% del grupo.

La cuarta y última nota obtenida fue una bonificación de 1.0 sacada con la presentación de los formatos diligenciados en las actividades de laboratorio. Esta bonificación se sumó a la nota del

trabajo final de la Materia. A continuación, en la tabla 30 se tienen los resultados:

Tabla 30. Cuarta Evaluación realizada.

Numero de Estudiante	Bonificación Obtenida
1	1.0
2	1.0
3	1.0
4	1.0
5	1.0
6	1.0
7	1.0
8	0.7
9	1.0
10	1.0
11	1.0
12	1.0
13	0.8

En la última actividad de evaluación bonificable para el trabajo final de la asignatura, los estudiantes obtuvieron en promedio 1.0 unidad; esto muestra que los estudiantes aprendieron a utilizar los formatos de registro de tiempos, de defectos y el formato de mejoras al programa; también crearon su propio estándar para el conteo de líneas de código. Todo esto les permitió obtener una excelente nota para sumar a la nota del proyecto final de la materia.

En cuanto a las notas finales de la materia ***Fundamentos de Algoritmia*** de los grupos

experimental y de control, puede verse un resumen de las mismas en la tabla 31.

Tabla 31. Comparación de Notas entre grupos

Característica a Comparar	Nota Grupo Experimental	Nota Grupo Control
Nota Final promedio	4.6	3.5
Nota Máxima Obtenida	5.0	4.5
Nota Mínima Obtenida	4.2	1.5
Desviación Estándar	0.2	0.7

En la anterior tabla, se puede observar que los estudiantes del grupo experimental obtuvieron mejores notas finales, dadas las bonificaciones que tuvieron para los tres exámenes parciales de la asignatura y el trabajo final. Puede observarse también, un comportamiento mucho más homogéneo entre las notas obtenidas por los estudiantes del grupo experimental, esto se puede atribuir a la alta motivación que tuvieron los estudiantes al utilizar las técnicas de aprendizaje cooperativo y sobretodo, al realizar las actividades de aprendizaje a través de equipos de trabajo, donde los estudiantes se sentían responsables por el aprendizaje de cada uno de sus compañeros.

6. CONCLUSIONES.

Luego de haber realizado esta tesis de grado, son varios los aprendizajes y experiencias que quedan y que se mencionan a continuación.

Es fundamental diseñar adecuadamente los instrumentos de recolección de información, ya que a partir de estos documentos se obtiene la información necesaria a partir de la población o de la muestra (en este caso, de los estudiantes) que se tiene. Fue necesario un proceso de diseño y rediseño de los instrumentos de Prestest y Postest, en los que se añadían y se suprimían preguntas o, se reescribían las mismas, hasta sentir que estos instrumentos se encontraban ya lo suficientemente maduros para ser aplicados a los estudiantes.

El proceso de recolección de los datos fue un proceso clave y dispendioso. Clave porque la información que los estudiantes depositaron en los instrumentos de recolección de información, fueron el insumo principal para el análisis que se llevó a cabo en la etapa final del proyecto, donde se contrastaron las respuestas dadas por los estudiantes de los grupos de control y experimental en el Pretest y el Postest. Inicialmente, los estudiantes vieron como un proceso engorroso el tener que responder a una serie de preguntas, que involucraban, la mayoría una serie de conceptos que ellos aún no habían visto durante la carrera; sin embargo, se notó con el grupo experimental durante el Postest, que los estudiantes valoraron los conceptos y prácticas que aprendieron durante la intervención aplicando la técnica de aprendizaje cooperativo.

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

Los estudiantes del grupo experimental manifestaron al inicio del semestre que PSP implica tiempo adicional en su proceso de desarrollo, percibiéndolo como un llenado de formatos sin valor agregado. Esta percepción está en correspondencia con resultados de investigaciones previas. A pesar de ello, en la medida en que los estudiantes fueron realizando las actividades planteadas, las mismas se incorporaron incrementalmente, llegando a ser realizada de forma natural por un número importante de los estudiantes del grupo experimental. El registro de los defectos resultó difícil para los estudiantes debido a que muchos de ellos no identificaban el tipo de defecto y se encontró que estos siempre eran ubicados en las mismas categorías. No obstante, al finalizar el proceso, los estudiantes reconocieron la importancia de estimar y registrar los tiempos de desarrollo de software y los defectos, así como la medida de Líneas de Código, pues son base para el futuro trabajo de programación, tanto en asignaturas posteriores del área de programación y otras de la carrera, así como también en su futuro desempeño profesional.

Con base en los resultados de la intervención al grupo experimental mediante la estrategia de aprendizaje cooperativo, puede concluirse que esta fue satisfactoria en lo que refiere a la gestión del tiempo y de los defectos, así como en la apropiación de los conceptos de PSP y estudiados. El aprendizaje cooperativo mostró ser una herramienta útil para ser aplicada en las asignaturas del área de programación de computadores, donde los estudiantes deben analizar problemas, aprender conceptos y utilizar lo aprendido en la construcción de programas de computador. El desarrollo de este trabajo mostró una serie de desafíos, pues desde lo pedagógico se puede afirmar que el éxito de estas experiencias académicas está asociado con la madurez de los estudiantes, para que reconozcan el valor de una disciplina aplicada a un proceso de programación (cuestión que todavía no han experimentado en su formación). (noopur david, 2003)(Bidegain, 1999)(Garcia, 2001)(colaboradores., 1999).

7. BIBLIOGRAFIA.

Anguas, J., Díaz, L., Gallego, I., Lavado, C., Reyes, A., Rodríguez, E., Valero, M. (2006). La técnica del Puzzle al servicio del aprendizaje de la programación de ordenadores.

Aronson y colaboradores., (1999). *the jigssaw classroom.*

Bermón, L., Fernandez, A., Sanchez, M., Javier, G.-, & Seco, A. (2009). Experiencias Docentes en la Aplicación del Proceso Software Personal en Primero de Grado de Ingeniería Informática. In *Fomento e Innovación con Nuevas Tecnologías en la Docencia de la Ingeniería* (pp. 107–114). Vigo.

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, 107–114.

Bidegain, U. (1999). *Aprendizaje cooperativo.*

Blvd, S. C. M., & Fl, D. B. (2007). SOFTWARE ENGINEERING BASED ON THE TEAM SOFTWARE PROCESS WITH A REAL WORLD PROJECT *, 133–142.

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

Car, Z. (2004). A METHOD FOR TEACHING A SOFTWARE PROCESS BASED ON THE PERSONAL SOFTWARE PROCESS.

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 Universidad del Quindío, 5–6.

Documento, C. E. (2001). El aprendizaje cooperativo: aprender a cooperar, aprendiendo cooperando.

Duran, D. E. S., & Gamboa, A. X. R. (2010). Recibido: 24 de agosto de 2009 Aceptado: 05 de octubre de 2009, 1–5.

Fredy, J., Manrique, N., & Anaya, R. (2012). TRANSVERSAL DE COMPETENCIAS DE GESTION , EN ESTUDIANTES DE UN PROGRAMA DE TECNOLOGÍA EN SISTEMAS Jhon Fredy Niño Manrique Octubre 2012.

Garcia. (2001). *Flujo de procesos de TSP relacionado con las fases.*

Gómez, O. S. (2014). Estudio del Proceso Software Personal (PSP) en un entorno académico, (2).

Honig, W. L. (2008). Teaching Successful “Real-World” Software Engineering to the “Net” Generation: Process and Quality Win! *2008 21st Conference on Software Engineering Education and Training*, 25–32. doi:10.1109/CSEET.2008.38

Humphrey, W. (1995). *A discipline for software engineering* (p. 789). Addison-Wesley.

Humphrey, W. S. (2000). The Personal Software Process (PSP) The Personal Software Process SM (PSP SM). *Engineering*, (November).

Jaramillo, S. G. (2012). Modelo para la selección de técnicas de aprendizaje colaborativo.

Lisack, S. K. (1997). The Personal Software Process in the Classroom : Student Reactions (An Experience Report).

Manzano, C., & Feliu, S. (2007a). TEAM SOFTWARE PROCESS (TSP): MEJORAS EN LA ESTIMACIÓN , CALIDAD Y, *4*, 9–17.

Manzano, C., & Feliu, S. (2007b). Impact of TSPi on Software Projects, 706–711. doi:10.1109/CERMA.2007.92

noopur david, m. l. (2003). *the team software process TSP in practice.*

Noopur Davis Julia Mullaney. (2003). The Team Software ProcessSM (TSPSM) in Practice: A Summary of Recent Results, (September).

Pons, R. M., Elena, M., José, G., & Serrano, M. (2008). Aprendizaje cooperativo en matemáticas : Un estudio intracontenido, *24*, 253–261.

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

Sáez, F. T. (2002). No Title.

Scagnoli, N. I. (2005). Estrategias para Motivar el Aprendizaje Colaborativo en Cursos a Distancia ., 1–15.

Segura, A. M. (2003). Diseños cuasiexperimentales. *Diseños cuasiexperimentales*, 1–4.

Sussy, B. O., Calvo-Manzano, J. a., Gonzalo, C., & Tomás, S. F. (2008). Teaching Team Software Process in Graduate Courses to Increase Productivity and Improve Software Quality. *2008 32nd Annual IEEE International Computer Software and Applications Conference*, 440–446. doi:10.1109/COMPSAC.2008.135

Torres, C., Augusto, S., Bermúdez, R., David, R., & Viviana, J. L. (2012). Implementación del Proceso Personal Software en un primer curso de Programación de Computadores, 90–102.

Towhidnejad, M., & Hilburn, T. (1997). Integrating the Personal Software Process (PSP) across the Undergraduate Curriculum. In *Frontiers in Education Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change* (pp. 162–168). Pittsburgh.

Towhidnejad, M., Hilburn, T., & Beach, D. (1997). Integrating the Personal Software Process (PSP) across the Un, 162–168.

Vargas, F. A. (2011). Productividad y asertividad. Metodología aplicada desde el aula 1, 104–112.

Venkatasubramanian, K., Roy, S. B. T., & Dasari, M. V. (2001). Teaching and Using PSP in a Software Engineering course : An Experience Report.

West, C., Farmer, J., & Wolff, P. (1991). *Instructional desing. Implication from cognitive science.*Needham : Allyn and Bacon.