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

SERGIO AUGUSTO CARDONA TORRES

Tesis para acceder al título de
Magíster en Ingeniería

Asesor: MSc. RAFAEL DAVID RINCÓN BERMÚDEZ

MEDELLÍN
UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS
MAESTRÍA EN INGENIERÍA
2012

Nota de aceptación

_____

_____

Presidente del jurado

_____

Jurado

_____

Jurado

_____

Medellín Octubre de 2012

DEDICATORIA

A mi esposa y a mis padres, por su amor y apoyo incondicional.

AGRADECIMIENTOS

Al Profesor Rafael David Rincón Bermúdez, por su apoyo y orientación; gracias MAESTRO.

A la Profesora María Dolly García, por compartir su tiempo y conocimientos.

Al Decano de la Facultad de Ingeniería José Fernando Echeverri Murillo, por sus consejos.

A la Universidad del Quindío y al Consejo de Facultad de la Facultad de Ingeniería.

A los estudiantes del Programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío.

GLOSARIO

TÉRMINO 1: Estrategias de aprendizaje

TÉRMINO 2: Competencias

TÉRMINO 3: Proceso de desarrollo de software

TÉRMINO 4: PSP (Personal Software Process)

TÉRMINO 5: TSP (Team Software Process)

TÉRMINO 6: Modelos de procesos

TÉRMINO 7: Atributos de calidad

TÉRMINO 8: Desarrollo de software

TÉRMINO 9: Mejores prácticas

TÉRMINO 10: Enseñanza de la programación

TÉRMINO 11: Métricas de calidad

TÉRMINO 12: Ambiente Virtual de aprendizaje

TÉRMINO 13: Estrategia de enseñanza

TÉRMINO 14: Teorías de aprendizaje

CONTENIDO

RESUMEN

En este artículo, se muestran aspectos relacionados con el desarrollo de la tesis de maestría titulada "Diseño de una estrategia de aprendizaje para implementar prácticas de PSP (Personal Software Process) y TSP (Team Software Process) en cursos básicos de Programación". En esta tesis, se presenta una propuesta metodológica de enseñanza que promueve la adopción de las prácticas de PSP y TSP en un curso de Fundamentos de Programación, que hace parte del currículo del Programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío en Armenia (Colombia). Los resultados obtenidos en esta investigación, permiten realizar un diagnóstico y una serie de reflexiones acerca de la importancia de implementar enfoques orientados a procesos que incorporen buenas prácticas y conceptos de calidad en el desarrollo de software desde etapas tempranas en la formación de los estudiantes. Con la introducción de PSP y TSP, los estudiantes van apropiando buenas prácticas de desarrollo de software y adquieren disciplina para planear y gestionar sus proyectos.

Palabras clave - calidad de software, estrategia de enseñanza, proceso personal de software, planeación, administración del tiempo, gestión de defectos.

1 INTRODUCCIÓN

Actualmente la construcción de aplicaciones informáticas debe responder a requerimientos muy complejos y de carácter crítico de las organizaciones. El software es considerado uno de los pilares estratégicos de estas organizaciones y de la sociedad en general, debido a que muchos de sus procesos, productos y servicios dependen en un alto grado del correcto funcionamiento de las aplicaciones de software que las apoyan. La complejidad inmersa en los proyectos de desarrollo de software está asociada a múltiples fuentes: metodologías utilizadas, tecnologías de apoyo, capacidad y competencias de las personas, productividad de los equipos de trabajo, requerimientos cambiantes de los clientes, presupuesto disponible, entre otras.

Es por ello que 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 ese contexto, el capital humano presente en el desarrollo de software es fundamental, tanto en su aporte individual como de su participación de trabajo en equipo. En el caso de un Ingeniero de Sistemas o afín, la productividad se ve afectada por la cantidad de defectos encontrados y arreglados (Ferguson, Humphrey, Khajenoori, & Macke, 1997), lo cual se traduce directamente en tiempos de desarrollo, costos y calidad. Y a pesar que en la actualidad se cuenta con librerías de componentes reutilizables y marcos de trabajo especificados, la responsabilidad recae de forma directa en quien desarrolla el software y en la cantidad de defectos que introduce.
La calidad del software depende en gran medida de la calidad del proceso empleado. El proceso incluye la metodología de desarrollo y otras actividades tales como la medición, planificación y la estimación, entre otras. El proceso debe permitir el mejoramiento continuo con base en indicadores calculados previamente, mediciones y datos históricos.

El TSP (*Team Software Process*) 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. PSP (*Personal Software Process*), por su parte, 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.

El desarrollo de habilidades individuales es necesario para potenciar el trabajo en equipo; en este sentido, el PSP (*Personal Software Process*) 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. El aprendizaje de PSP es considerado requisito para el uso del proceso de desarrollo de software en equipo.

En el contexto académico de la Ingeniería de Sistemas y afines, se pueden establecer estrategias de aprendizaje que permitan desarrollar en los estudiantes habilidades que potencien sus prácticas individuales y en equipo, para que al momento de insertarse en el mundo productivo puedan entender, explicar y aplicar en contextos reales las mejores prácticas en los proyectos de desarrollo.

El presente proyecto de tesis pretende utilizar el marco de procesos de PSP y TSP, y un subconjunto de sus mejores prácticas, para construir una estrategia de aprendizaje que permita a los estudiantes del programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío el mejoramiento de su plan desarrollo de software, de sus estimaciones, la calidad de su trabajo y sus planes de medición.

2 OBJETIVOS

OBJETIVO GENERAL

Diseñar un estrategia de aprendizaje para implementar prácticas de PSP y TSP en cursos básicos de Programación y Algoritmia, dirigido a estudiantes de Ingeniería de Sistemas y Computación de la Universidad del Quindío, que les permita desarrollar habilidades de desarrollo de software tanto individuales como de equipo, para una efectiva estimación de tamaño, costos y tiempos, manejo de estándares, prevención y remoción de defectos, de acuerdo con las necesidades de la industria de software nacional.

OBJETIVOS ESPECÍFICOS

A. Analizar el estado del arte y los resultados de las experiencias más significativas a nivel mundial del uso de PSP y TSP en la academia, para identificar su impacto en el proceso de formación de los estudiantes de pregrado de Ingeniería de Sistemas y carreras afines, y que puedan servir de referencia para tener un sustento teórico en la realización de esta propuesta.

B. Realizar un diagnóstico sobre el uso de buenas prácticas de desarrollo de software por parte de los estudiantes del programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío, relacionados con el área de Programación y Algoritmia, para definir una línea base de esta investigación.

C. Seleccionar una teoría pedagógica que permita la implementación de la estrategia de aprendizaje para el desarrollo de esta investigación.

D. Identificar y analizar las mejores prácticas de PSP y TSP, para seleccionar las métricas en la estructuración de la estrategia de aprendizaje.

E. Diseñar y estructurar los escenarios, actividades y recursos que permitan confrontar las diferentes situaciones de aprendizaje, por medio de una estrategia que facilite a los estudiantes de los cursos de Programación y Algoritmia, la apropiación y aplicación de las mejores prácticas de PSP y TSP durante su proceso de formación en lo relacionado con el desarrollo de software.

F. Realizar una prueba piloto en un curso de Programación de Ingeniería de Sistemas y Computación, con el fin de verificar y valorar que las estrategias de aprendizaje propuestas aportan al desarrollo de las competencias esperadas en los estudiantes.

3 JUSTIFICACIÓN

Teniendo en cuenta que Colombia se está proyectando como un país con capacidad para desarrollar software de alta calidad, se hace necesario y prioritario que los profesionales y estudiantes inmersos en esta disciplina tengan diferentes tipos de competencias requeridas, que respondan a las necesidades de cambio en el ámbito del desarrollo de software. Por ello, desde las instituciones de educación superior se debe promover el desarrollo de competencias que estén asociadas con esa dinámica de cambio, teniendo en cuenta que el recurso humano es fundamental en esta industria. La constante evolución de la Ingeniería de Sistemas y/o afines, nos invita necesariamente a pensar en los roles de desempeño de los profesionales y egresados, en los que se plantean requerimientos de TI más complejos, siempre girando en torno a los proyectos, en donde la gestión de los mismos, de los equipos de trabajo y de las personas, es fundamental para la consecución de los objetivos.

Se hace necesario entonces desde la academia analizar la forma como actualmente se están formando los estudiantes de los programas relacionados con el desarrollo de software, para que puedan gestionar adecuadamente su proceso de desarrollo personal, de acuerdo con las necesidades que actualmente presenta la industria del desarrollo de software. Esta gestión del trabajo personal y en equipo del desarrollo de software permitiría al ingeniero administrar un proceso definido, medido y controlado con criterios de calidad. Los estudiantes deben aplicar las mejores prácticas de desarrollo de software tanto en el trabajo individual como de equipo en temas como: gestión, estimación de tamaño, costos y tiempos, manejo de estándares, remoción y prevención de defectos. Es necesario que los estudiantes en su proceso de formación asuman roles y funciones, se expongan a ambientes de trabajo similares a la industria, en donde se consideren aspectos que incorporen diferentes prácticas como una forma de ser o un hábito normal del ingeniero.

Nos encontramos en la actualidad frente a estudiantes que poseen características de adaptación rápida a los cambios, muy visuales en sus procesos de aprendizaje, con apoyo constante de herramientas tecnológicas en su proceso de apropiación y que aprenden haciendo interacción constante y permanente con otros. Se hace entonces necesario desarrollar estrategias de aprendizaje que potencialicen todas estas características. En la actualidad existe gran cantidad de información para documentar y gestionar todo el proceso de software personal y en equipo, pero no se tiene una estrategia de aprendizaje que permita evidenciar el desarrollo de competencias esperadas en este contexto. Si se desea incorporar elementos diferenciadores de formación en los estudiantes de pregrado, ellos deben ser capaces de planificar su tiempo y administrar su trabajo,

lo que les permitiría adaptarse a la dinámica de los proyectos de desarrollo de software en el mundo real.

El impacto del desarrollo de este proyecto se podría evidenciar en las habilidades de los estudiantes en los espacios académicos del área de Programación y de Ingeniería de Software, tales como: (paradigma orientado a objetos, fundamentos de algoritmia, lenguaje de programación, estructuras de datos y análisis de algoritmos e Ingeniería de Software), pues en estos espacios académicos los estudiantes ya tienen conocimientos de programación, lo cual es un requisito indispensable para trabajar con PSP.

También se plantean interrogantes en esta investigación para que a través del quehacer docente permita desde la Universidad del Quindío la valoración del impacto de la enseñanza de la Programación, implementando procesos de PSP y TSP en la formación de los estudiantes, a partir del estudio de casos. El poder implementar y validar este proyecto de tesis puede servir como aporte a una reforma curricular en el programa de Ingeniería de Sistemas y Computación, teniendo en cuenta que se está iniciando un proceso de análisis en cuanto a los perfiles y competencias que se desea de los egresados de este programa.

4 PLANTEAMIENTO DEL PROBLEMA

La industria del desarrollo de software y de las tecnologías de la información en general necesita contar con profesionales que posean una alta capacidad para producir soluciones a las necesidades mundiales; capaces de incorporar las mejores prácticas del proceso de desarrollo de software y aplicarlas a los requerimientos de información cada vez más complejos de las organizaciones y en general de la sociedad. De acuerdo con los estudios del Standish Group, en el reporte Chaos Report 2009[1] se indica que sólo el 32% de los proyectos de software fueron exitosos y el 68% de estos proyectos fracasaron o no cumplieron con criterios y requisitos de calidad. Por ello se puede afirmar que existe aún una brecha muy amplia entre las buenas prácticas y las prácticas artesanales que aún son utilizadas para la construcción de software.

Para impulsar la industria Colombiana de software es necesario contar con personal competente que esté en los niveles de los mejores del mundo y con la capacidad de utilizar en su trabajo estándares internacionales. Actualmente las empresas desarrolladoras de software consideran que los mayores problemas que afectan el crecimiento de su empresa es la falta de aplicación de estándares de calidad internacionales, el desconocimiento de la logística de comercialización y la escasez de recurso humano capacitado, entre otros[2] . Esto implica que las empresas deben invertir gran cantidad de tiempo y costos de inducción en el nuevo personal de desarrollo.
Históricamente los currículos de la Ingeniería de Sistemas y afines se centran en el desarrollo de habilidades y capacidades técnicas para que los estudiantes desarrollen software. Se enseñan diferentes metodologías que permiten al estudiante apropiarse de técnicas y métodos de desarrollo de software, pero el concepto de calidad no se encuentra plenamente caracterizado, y por lo tanto no se posee una cultura alrededor de este concepto.

Las prácticas de PSP y TSP se han utilizado de forma muy general en cursos de últimos semestres, como en el caso del ITESM, Instituto Tecnológico y de Estudios Superiores de Monterrey, y de la UAZ, Universidad Autónoma de Zacatecas, en México, en cuyas instituciones se ha estado ofreciendo un curso de TSP como materia optativa[3]; sus principios pueden ser aplicados por los estudiantes desde cursos iníciales de programación y a través de los cuales pueden descubrir cómo planear y en general gestionar su trabajo. De otra parte, en un reporte de

[1] http://www.standishgroup.com/newsroom/chaos_2009.php
[2] Introduciendo PSP en el Aula. MSc. Dario E. Soto Duran, MSc. Adriana X. Reyes Gamboa. Revista Colombiana de Tecnologías de Avanzada. Octubre 2009.
[3] Villa Cisneros Juan Luis; De León Sigg María; Solís Recéndez Blanca. Software Process (PSP) a nivel Licenciatura.

experiencias en el aula[4] se muestra que dada la metodología propuesta en PSP, puede resultar abrumadora para un estudiante que, además de estar aprendiendo a programar, debe también completar gran variedad de registros. Ello muestra la necesidad de implementar estrategias de aprendizaje que motiven a los estudiantes a apropiar de forma adecuada, buenas prácticas de desarrollo de software durante su proceso de formación.

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

De acuerdo con las anteriores consideraciones, se identifica la necesidad de proponer estrategias de aprendizaje para incorporar prácticas de PSP y TSP en cursos básicos de programación y del área de Ingeniería de Software, las cuales a partir de un análisis de las políticas de formación, la aplicación de teorías de aprendizaje, el diseño de escenarios de interacción y colaboración y la instrumentación didáctica, provean a los estudiantes elementos que permitan el desarrollo de competencias específicas en su proceso de formación.

[4] Susan K. Lisack. The Personal Software Process in the Classroom: Student Reactions (An Experience Report), Purdue University.

5 MARCO TEÓRICO

Este capítulo contiene un contexto general de la industria del software en Colombia y el marco de referencia con el cual se fundamenta la realización de este trabajo.

En una investigación nunca se parte de cero, sino que se considera una base teórica y conceptual determinada. Ella guía todo el proceso, y con base en ella llegamos nuevamente al objetivo de toda investigación: generar un conocimiento válido y generalizable (Tamayo, 1999). El marco de referencia que se presenta en esta tesis consta del marco teórico y los antecedentes. El marco teórico es el resultado de una investigación preliminar sobre teorías, consolidadas en la literatura, que se están siguiendo como modelo de la realidad y, que se utiliza para tomar decisiones en el diseño y para ordenar y estructurar la investigación (Giraldo, 2010). A partir del marco teórico se identifican los elementos conceptuales fundamentales para clasificar los trabajos que provienen del marco de antecedentes.

Este marco contiene los antecedentes que son la fuente de información y las cuales se encuentran en diferentes tipos de publicaciones. Para esto, se realizará un análisis de los resultados más significativos a nivel mundial del uso de PSP y TSP en la academia, con el cual se pretende identificar la incidencia en el proceso de formación de estudiantes con diferente nivel de formación en el área de la Ingeniería de Software y en general de la Informática. También se busca conocer las lecciones aprendidas de experiencias pedagógicas en diferentes Universidades a nivel mundial.

Finalmente, se presenta una sección en donde se analizan cada una de las investigaciones realizadas y a partir de ellas se identifican los elementos diferenciadores para el desarrollo de esta tesis.

5.1 Contexto de la Industria del Software en Colombia

Muchos gobiernos de países desarrollados y emergentes han tratado de avanzar durante los últimos años hacia un modelo de desarrollo que se ha denominado la Sociedad del Conocimiento. Este modelo se apoya en el uso adecuado y en la apropiación de las TIC para lograr el crecimiento productivo y el progreso económico y social. Para ello, han desplegado diferentes planes y estrategias para impulsarlo (Ministerio de comunicaciones, 2008).

En ese sentido el Gobierno Nacional desde el 2004 ha liderado el proceso de construcción de la Agenda Interna para la Productividad y la Competitividad, la cual es un acuerdo de voluntades y decisiones entre el Gobierno Nacional, las entidades territoriales, el sector privado, los representantes políticos y la sociedad civil, sobre las acciones estratégicas que debe realizar el país para mejorar su productividad y competitividad. Teniendo en cuenta la heterogeneidad regional del país, la Agenda Interna se sustenta en la necesidad de contar con estrategias de desarrollo económico diferenciadas, que reconozcan y atiendan las particularidades de cada región (Departamento Nacional de Planeación, 2007).

Dentro de las apuestas productivas que se identifican en las diferentes regiones del país, está el sector de los servicios, en donde se encuentran el sector de la Informática, telecomunicaciones y el desarrollo de Software. Se evidencia entonces el interés por desarrollar acciones y estrategias de consolidación e internacionalización del sector de las TIC y la industria de software en Colombia en general, teniendo en cuenta que en los últimos diez años esta industria creció muy por encima de otros países.

En el Plan Nacional de Tecnologías de la Información y las Comunicaciones del Ministerio de Comunicaciones se reconoce el impacto de estas tecnologías en la competitividad, su potencial para apoyar su inserción en la economía globalizada e impulsar el desarrollo económico y social de los países. Estos beneficios sólo pueden convertirse en resultados concretos a medida que la sociedad se apropie de estas tecnologías y las haga parte de su desempeño cotidiano. Es decir, con usuarios preparados que utilicen las TIC se puede lograr una verdadera transformación económica y social. En la figura 1 se muestra la evolución en ventas del mercado de TI en América Latina entre los años 2008 y 2011, en la cual se muestra una tendencia creciente para los segmentos de software, servicios y hardware.

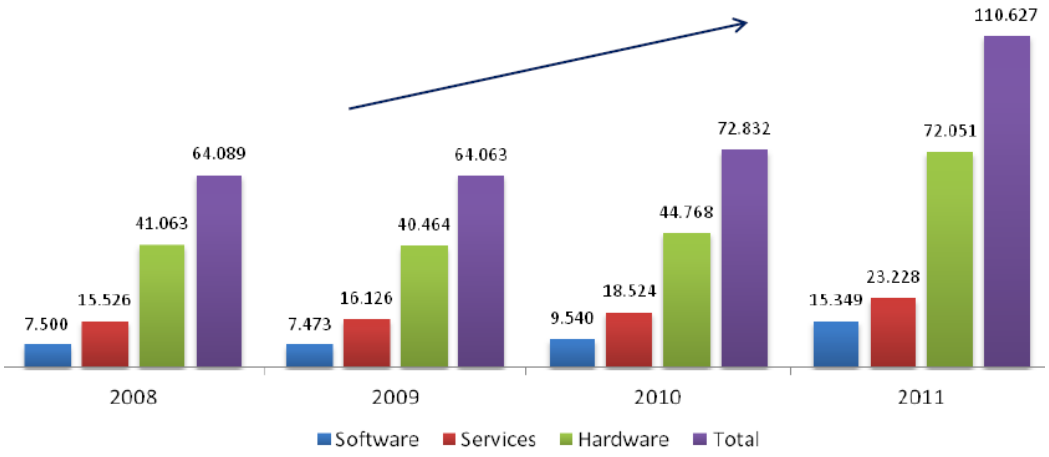


Figura 1. Evolución del mercado de TI en America Latina (Proexport, 2012)

Colombia actualmente es un país con un gran auge en los mercados de las Tecnologías de la Información (TI), especialmente en los sectores de servicios de tecnologías y de software. Según Bussines Monitor International (tomado de (ProExport, 2011)) el país es el tercero a 2009 en participación en la industria de TI en América Latina, con un 1.03% del Producto Interno Bruto. El valor total en el mercado del sector de las TI en Colombia es de US$ 2976 millones, y el crecimiento esperado entre el 2010 y el 2014 es del 11%.

Según (Fedesoft, 2011) , las ventas de software en millones de dólares en el año 2011 fueron de aproximadamente US$ 602,9 millones, y se proyecta que para los años 2012 y 2013 esta cifra estará en US$ 680 millones y US$ 760 respectivamente. En cuanto a las exportaciones de software se reporta que en el año 2011 la suma estuvo alrededor de los US$ 40 millones, y se proyecta que para el año 2012 la cifra estará en US$ 54 millones y para el 2012 en US$ 75 millones.

5.1.1 La industria del Software en Colombia

En Colombia, el sector de software está principalmente orientado hacia el mercado interno, con un bajo nivel de exportaciones, con el agravante de que el mercado interno no está muy desarrollado y se encuentra poco extendido el Outsourcing en el sector productivo colombiano. Así las cosas, se requiere de algún apoyo estatal para romper este círculo vicioso de bajo crecimiento y poco desarrollo de este tipo de servicios (Ministerio de comunicaciones, 2008).

Actualmente, la industria colombiana de TI es poco especializada, orientada al mercado doméstico y enfocada principalmente en:

1. Comercialización y soporte de software empaquetado;
2. Desarrollo de software a la medida; y
3. Consultoría e integración de sistemas.

Superando grandes brechas en la disponibilidad de recurso humano y madurez de la industria, Colombia podría convertirse en un jugador relevante en TI, gracias a su ventaja relativa en costos, infraestructura y ambiente de negocios (Ministerio de Comercio, Industria y Comercio, 2008).

Muchas empresas de software colombianas están tratando de cruzar las fronteras en busca de nuevas oportunidades. Sin embargo, el terreno es muy difícil. De acuerdo con (Fedesoft, 2009), menos del 10% de estas empresas tiene o ha tenido incursiones en el mercado internacional, mientras las exportaciones del sector en su conjunto han crecido a razón de 6% anual. Este resultado es superior al promedio de la región, 4,5%, pero es todavía muy bajo si se compara con Israel, China e India, países que han alcanzado crecimientos anuales superiores al 100% en los últimos 4 años (Instituto Español de Comercio Exterior, 2006).

Las compañías colombianas son reconocidas en el mercado internacional como buenos "artesanos de software" y se han concentrado en el desarrollo y comercialización de lo que se conoce como aplicaciones a la medida, es decir, aplicaciones desarrolladas específicamente para los requerimientos del usuario. Esto se explica porque el mercado del software empaquetado es ampliamente dominado por compañías estadounidenses y europeas, en un segmento en el cual es casi imposible competir (Revista Dinero, 2002).

La calidad se mide hoy con estándares internacionales muy claros, que deben ser cumplidos por cualquier empresa que aspire a hacer negocios. En muchas ocasiones, estos estándares son tan elevados que pocas empresas colombianas pueden cumplirlos. Buena parte de las empresas de software colombianas que han incursionado con éxito en mercados internacionales han basado su estrategia en alianzas con compañías multinacionales, para poder llegar a las grandes corporaciones que exigen estos estándares de calidad, especialmente en Estados Unidos y Europa (Instituto Español de Comercio Exterior, 2006).

Ante las oportunidades que tienen estas empresas de desarrollo de software para incursionar en mercados internacionales, se debe trabajar entre otros en: trabajar con estándares de calidad internacional, capacitación de las personas que trabajan en esta industria y mejorar los esquemas de comercialización de los productos y servicios. Para impulsar la industria Colombiana de software es necesario contar con personal competente que esté en los niveles de los mejores del mundo y con la capacidad de utilizar en su trabajo estándares internacionales. Actualmente las empresas desarrolladoras de software consideran que los mayores problemas que afectan el crecimiento de su empresa es la falta de aplicación de estándares de calidad internacionales, el desconocimiento de la logística de comercialización y la escasez de recurso humano capacitado, entre otros (Soto & Reyes, 2010). Esto implica que las empresas deben invertir gran cantidad de tiempo y costos de inducción en el nuevo personal de desarrollo.

Dadas estas situaciones, las empresas de desarrollo de software están adoptando modelos internacionales de calidad que permitan definir e institucionalizar procesos, buenas prácticas y un mejoramiento continuo. Actualmente algunas de las principales compañías de software del país ya han sido valoradas en el modelo CMMI (*Capability Maturity Model Integration*); para el año 2009 se tenían 15 empresas valoradas, y en el año 2010 ya se contaba con 40 empresas. Actualmente se está desarrollando un programa promovido por el SENA, Colciencias y Proexport apoyando a cerca de 60 compañías a obtener la valoración CMMI.

Fedesoft, en conjunto con Intersoftware, SEONTI (SEI Partner), y el apoyo económico del SENA, realizó capacitación a más de mil profesionales del sector Software & TI en la implementación de prácticas TSP (*Team Software Process*) y PSP (*Personal Software Process*), procesos personales y de equipo para desarrollo de software, por el cual los profesionales recibieron certificados otorgados por el Instituto de Ingeniería de Software (SEI) de la Universidad Carnegie Mellon de Estados Unidos, como base para lograr que la industria colombiana del software sea altamente competitiva a nivel internacional, basada en altos niveles de calidad.

En el 2011 se capacitaron 193 directivos, 233 líderes de proyecto, 394 desarrolladores y 216 personas de equipos de apoyo. Para el 2012 se continuará con el proceso de capacitaciones, certificaciones e implementaciones, con el fin de proporcionar un diferenciador importante a las empresas colombianas, ya que estas prácticas apoyan la creación y administración de equipos de desarrollo de software (Fedesoft, 2011).

5.1.2 El recurso humano como factor de desarrollo

Según (Ministerio de Comercio, Industria y Comercio, 2008), en Colombia cada año egresan 7.282 personas entre técnicos, tecnólogos, ingenieros de sistemas y electrónicos para trabajar en la Industria de TI.

En cuanto a la disponibilidad de recurso humano altamente calificado, según el IMD (*Institute for Management Development*) tomado de (ProExport, 2011), Colombia es el segundo país con índice más alto en América Latina, en cuanto a la disponibilidad de mano de obra calificada. En la siguiente figura se muestra la situación de Colombia, considerando que 10 corresponde a alta disponibilidad de mano de obra.

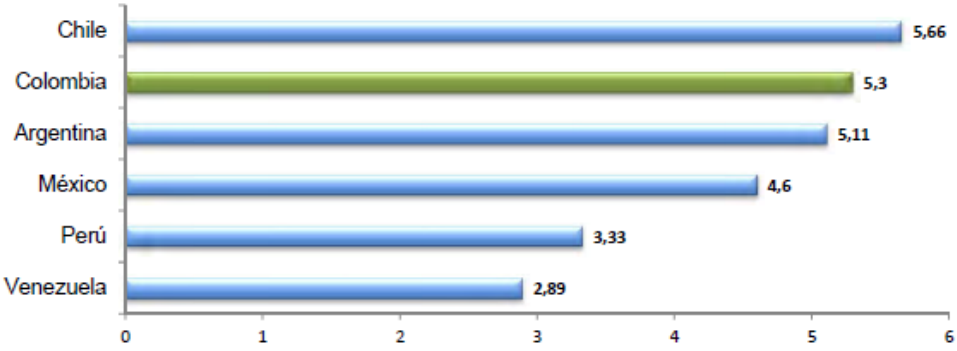


Figura 2. Disponibilidad de trabajadores altamente calificados. (ProExport, 2011)

Actualmente desarrollar un trabajo competente requiere la aplicación de buenas prácticas, planes y procedimientos que lleven de forma ordenada y eficiente las actividades y que permita a las personas concentrarse para construir productos de la más alta calidad. En este sentido, para desarrollar la industria Colombiana de Software es necesario contar con recurso humano altamente competitivo, entrenado en una disciplina personal y de trabajo en equipo.

En cuanto el sector de la educación, la Ingeniería de Sistemas y sus afines, responde a la necesidad actual del país para la formación de capital humano necesario para el desarrollo económico de las regiones y del país.

Sin embargo, se observa que aún sigue existiendo una brecha entre las necesidades de la industria del software y la academia, se tienen reportes ampliamente documentados sobre esta situación, tales como (Raymond, 1993), (Tucker A. , 1991).Un informe del Instituto de Ingeniería de Software, SEI, reporta que: "Los estudiantes no están preparados para dar el salto entre la Ingeniería de Software en la Universidad a la Ingeniería de Software en el mundo profesional". Una razón de la brecha existente entre lo que los estudiantes aprenden en las universidades y las habilidades requeridas por la industria del software es que son pocos los programas académicos que desarrollan las competencias que se esperan de los Ingenieros de Software (Towhidnejad & Hilburn, 1997).

5.2 Fundamentos Teóricos

En esta sección se presentan algunas descripciones teóricas sobre temas que son relevantes en el desarrollo de la investigación. En este sentido, se considera que son pilares para establecer la

coherencia y consistencia del marco de referencia. Los fundamentos teóricos proporcionan, principalmente, la terminología que es utilizada para definir la propuesta de investigación.

5.2.1 *Personal Software Process* (PSP)

Los proyectos de desarrollo de software están relacionados con problemas asociados con factores como calidad, sobrecostos, y los cronogramas exceden los tiempos establecidos. En parte, estos problemas tienen su raíz en el nivel individual (Zhong, Madhavji, & El Emam, 2000).

El *Personal Software Process* (PSP) fue creado por Watts Humphrey para hacer frente a la necesidad que los Ingenieros de Software adquieran un enfoque disciplinado y eficaz para la construcción de programas. La filosofía detrás de PSP es que la capacidad de una organización para construir sistemas de software a gran escala depende de la capacidad de sus Ingenieros de Software para el desarrollo de alta calidad, de una forma efectiva y disciplinada.

El PSP está diseñado para ayudar a los ingenieros a organizar y planificar su trabajo, realizar el seguimiento de su rendimiento, la gestión de defectos de software, y analizar y mejorar su proceso de desarrollo personal (Towhidnejad & Hilburn, 1997). Presenta formas con las cuales los ingenieros pueden gestionar la calidad de los productos de una forma planificada.

PSP muestra a los ingenieros la forma de gestionar la calidad de sus productos, proporciona información para que las personas mejoren su capacidad de estimación y planificación, con miras a optimizar su proceso de desarrollo personal.

El diseño de PSP se basa en los siguientes principios de planeación y de calidad (Humphrey W. , 1995)

- 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 de la mejor manera la planeación del trabajo.
- Para que los desarrolladores lleguen a entender su funcionamiento de manera 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 de PSP se muestra adaptado de (Humphrey W. , The Personal Software Process, 2000) en la figura 3. Primero se planifica el trabajo. Se continúa con las fases de Diseño, Codificación, Compilación y Pruebas. A medida que el trabajo se realiza, se registran datos de tiempos y defectos. Al finalizar el proyecto, se realiza un análisis post-mortem para completar el resumen del plan de proyecto (Humphrey W. , 1996).

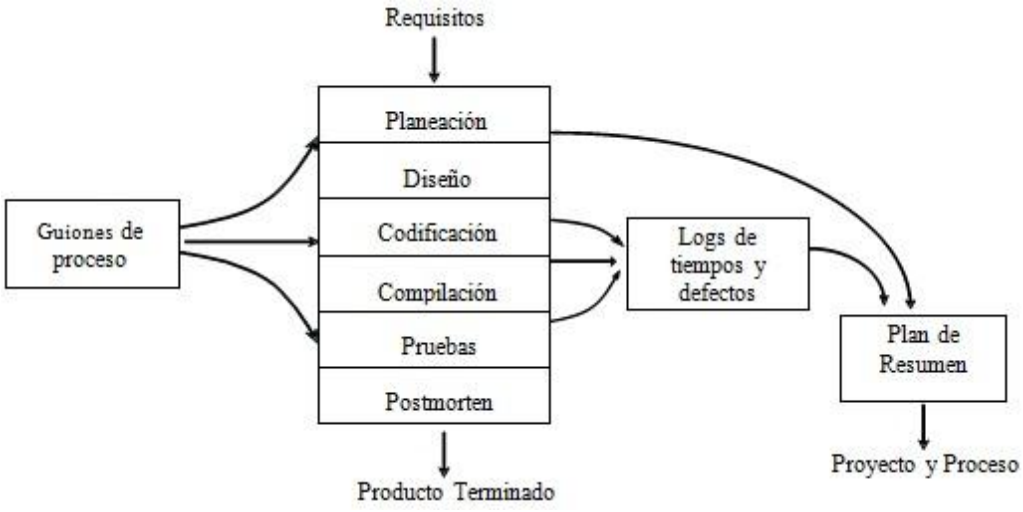


Figura 3. Flujo de procesos de PSP (Humphrey W. , The Personal Software Process, 2000)

El PSP presenta un conjunto de formularios que se debe registrar y unos estándares para cada una de las etapas del proceso de desarrollo, cuyo objetivo fundamental es ofrecer métodos para incrementar la productividad y la calidad del Ingeniero de Software.

PSP es un proceso evolutivo e incremental. Los niveles de procesos PSP son (Humphrey W. , 1996). PSP utiliza un proceso de madurez, por medio del cual el ingeniero de forma incremental va aplicando buenas prácticas de desarrollo de software. Para ello se diseño un marco de referencia

por niveles, cada uno de los cuales se apoya y extiende en el anterior introduciendo al ingeniero en nuevas practicas. Son siente los niveles establecidos para el proceso: PSP0, PSP0.1, PSP1, PSPS1.1, PSP2, PSP 2.1 y PSP3. En la medida en que el modelo avanza progresivamente, se van obteniendo resultados de forma que los mismos puedan se analizados y como pueden ser mejorados.

La figura 4, muestra el modelo de madures definido en (Humphrey W. , 1995). El nivel de medición personal se compone del PSP0 y el PSP0.1. El nivel PSP0 permite a los ingenieros identificar las practicas de desarrollo de software comúnmente utilizadas por la persona, estas prácticas tienen asociadas una serie de métricas que permiten realizar una medición del desempeño de las personas en aspectos como el registro de tiempo y la gestión de defectos. En PSP0 se pretende que el ingeniero conozca su rendimiento mediante tres fases: planificación, desarrollo y postmorten.

El PSP0.1 extiende el PSP0 incorporando aspectos relacionados con estándares de codificación, técnicas de medición de tamaño del producto y una propuesta de mejora del proceso. En PSP0.1 se realiza una medición del tamaño del programa en líneas de código (LOC). Adicionalmente en este nivel es necesario el registro de los problemas detectados durante la construcción del programa denominado el PIP (*Process Improvement Proposa*l).

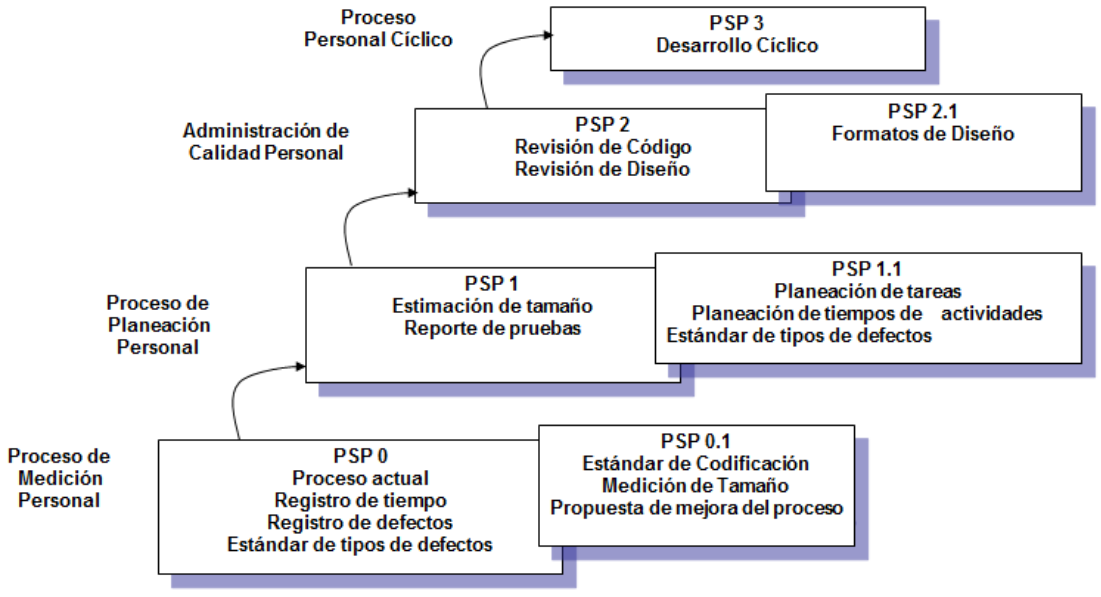


Figura 4. Modelo de Madurez de PSP

En el nivel de planeación personal, tiene como objetivo fundamental identificar es esfuerzo de trabajo mediante la relación del tiempo necesario para la realización del proyecto y su tamaño. Este nivel consta de dos versiones: PSP1 y PSP1.1.

En PSP1 se define y aplica un método para realizar estimaciones del tamaño del programa, denominado PROBE (*PROxy Based Estimating*) (Humphrey W. , 1995). Este método utiliza regresión lineal sobre datos históricos de tamaños de programa, recolectados en desarrollos similares determinando también la exactitud de dicha estimación. También en este nivel se introduce el concepto de reporte de pruebas para registrar datos sobre las pruebas realizadas (Pérez, 2009). En el PSP1.1 se introduce el planifican las actividades a realizar durante el proyecto mediante la elaboración de un cronograma, para determinar fechas claves durante la realización del proyecto.

El nivel de administración personal de la calidad se enfoca en la calidad del producto construido. Este nivel se compone de dos versiones: PSP2 y PSP2.1.

El PSP2 introduce se enfoca en la gestión de defectos como medio para asegurar la calidad del producto desarrollado. Se realizan revisiones tempranas de diseño y codificación, las cuales se realizan mediante listas de chequeo, las cuales tienen como objetivo reflejar los errores más comunes durante el proceso de desarrollo de software. El PSP2.1 se enfoca en el diseño de software, en donde se proporciona al ingeniero una serie de plantillas que permiten documentar el diseño desde diferentes perspectivas.

El último nivel de PSP es el proceso cíclico. En el nivel PSP 3se introducen técnicas para el desarrollo de proyectos a gran escala y en el cual el ingeniero debe tener la capacidad para desarrollar programa de miles de líneas de código.

5.2.2 *Team Software Process* (TSP)

Una definición aceptada de un equipo, es la dada por (Dyer, 1984), en la cual establece que un equipo consiste en al menos dos personas que trabajan para lograr una meta/ objetivo/ misión común, donde cada persona tiene asignado un rol específico o funciones específicas que desarrollar, y para completar la misión se requiere alguna forma de dependencia entre los miembros del equipo.

Según (Hogan & Thomas, 2005), el trabajo en equipo es una de las principales competencias que deberían tener los egresados de las carreras de desarrollo de software. En especial se mencionan las competencias de comunicación y manejo de tiempo.

El *Software Engineering Institute* (SEI) ha desarrollado el TSP para ayudar a equipos de Ingenieros de Software a elaborar productos de software de manera eficiente. TSP extiende y refina los métodos CMMI y PSP, para guiar a los miembros de los equipos en el trabajo de desarrollo y mantenimiento. TSP muestra cómo construir un equipo auto dirigido y cómo ser un miembro efectivo del equipo. También les enseña cómo dirigir y soportar estos equipos y mantener un medio para obtener un alto nivel de desarrollo. TSP define cinco funciones específicas: Gerente de Soporte, Calidad / *Process Manager*, Gerente de Planificación, Gerente de Desarrollo y Jefe de Equipo (Honig, 2008). TSP hace énfasis en el cliente, el equipo, y el individuo de desarrollo, y define los procesos de equipo, los roles, así como una guía para el desarrollador, tanto individual como del equipo.

A diferencia de otras metodologías, el TSP requiere el entrenamiento de un alto porcentaje de las personas que trabajan en la organización y requiere del trabajo cercano de *coaches* certificados que entrenan y guían al equipo en el correcto desarrollo del proyecto. El TSP requiere de cambios profundos en las prácticas individuales y grupales de los desarrolladores de software, pero los resultados que se obtienen son sorprendentes. Es por eso muy importante que esta metodología se implemente con gran cuidado, sin reducir su alcance y con una muy cercana supervisión.

TSP conduce al equipo a través de cuatro fases dentro de un mismo proyecto. Estos proyectos deben empezar o terminar con alguna fase o pueden ejecutarse desde el principio al final. Antes de cada fase, el equipo planifica y organiza su trabajo mediante una puesta en marcha completa (launch) o relaunch (García, 2011).

En las empresas donde se ha implementado el TSP en forma correcta, se han logrado obtener beneficios como los siguientes (Alva & Valdez, 2011):

- Un aumento en la productividad de hasta un 70%
- Una reducción del error en la estimación de tiempo y costo de los proyectos en un ±5%
- Una reducción de la cantidad de los defectos entregados al cliente a un equivalente a 5 sigma (60 defectos por cada millón de líneas de código)

- Una reducción del tiempo para certificarse en CMMI a menso de la mitad

El TSP requiere la recolección semanal de los datos que incluye el tiempo trabajado en actividades específicas: el tamaño de los documentos y el código producido, los problemas, errores o defectos encontrados. Los equipos deben registrar cada fallo o defecto encontrado (en todas las entregas del equipo, incluidos los documentos), el informe y un resumen de estos defectos de forma semanal.

En la figura 5 se muestra el flujo de proceso para TSP la interacción entre cada una de estas fases dentro del proceso de desarrollo.

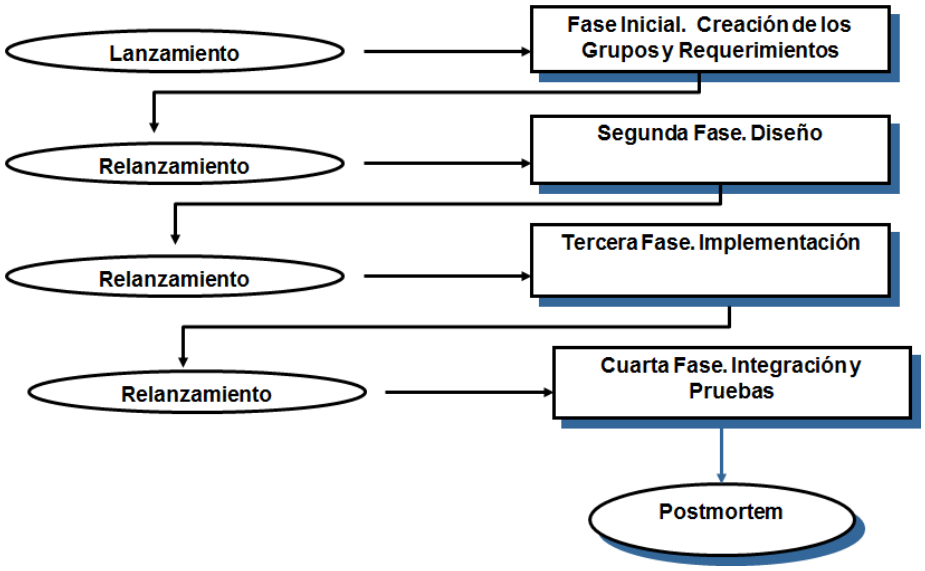


Figura 5. Fases del ciclo de vida TSP (García, 2011)

En la tabla 1 se realiza la descripción adaptada de (Humphrey W. , Introduction to the Team Software Process, 2000) para cada una de las fases de TSP.

Tabla 1. Descricpión de fases de TSP (Humphrey W. , Introduction to the Team Software Process, 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.
Postmorten	Se generan las evaluaciones del equipo, lecciones aprendidas y se presenta el estado del proyecto.

La correcta implementación de TSP en una organización empieza con la apropiada capacitación de los ejecutivos y gerentes de la misma, para proceder a seleccionar uno o dos equipos de ingenieros que realizarán proyectos piloto usando TSP. A estos ingenieros se les capacita primero en las prácticas individuales que deben de dominar, a través del PSP (Alva & Valdez, 2011).

5.2.3 Estrategia de Aprendizaje

El desarrollo de este trabajo está relacionado con el diseño de una estrategia de aprendizaje, la cual está orientada a la consecución de una serie de capacidades y habilidades de los estudiantes.

Para (Weinstein & Mayer, 1986) las estrategias de aprendizaje pueden ser definidas como conductas y pensamientos que un aprendiz utiliza durante el aprendizaje con la intención de influir en su proceso de codificación. Para (Monereo, 1994), las estrategias de aprendizaje son procesos de toma de decisiones (conscientes e intencionales) en los cuales el alumno elige y recupera, de manera coordinada, los conocimientos que necesita para complementar una determinada demanda u objetivo, dependiendo de las características de la situación educativa en que se produce la acción.

Para otros autores (Schmeck, 1998) (Schunk, 1991), las estrategias de aprendizaje son secuencias de procedimientos o planes orientados hacia la consecución de metas de aprendizaje, mientras que los procedimientos específicos dentro de esa secuencia se denominan tácticas de aprendizaje.

La estrategia de aprendizaje es un procedimiento que un alumno adquiere y emplea de forma intencional como instrumento flexible para aprender significativamente y solucionar problemas y demandas académicas (Díaz & Lule, Efectos de las estrategias preinstruccionales en alumnos de secundaria de diferentes niveles socieconómicos, 1978). Los objetivos particulares de cualquier estrategia de aprendizaje pueden consistir en afectar la forma en que se selecciona, adquiere, organiza o integra en nuevo conocimiento, o incluso la modificación del estado afectivo o motivacional del aprendiz, para que este aprende con mayor eficacia los contenidos curriculares que se le presentan (Dansereau, 1985),

Según (Pozo & Postigo, 1993) las características de las estrategias de aprendizaje consideran aspectos, como:

a. Su aplicación no es automática sino controlada. Precisan planificación y control de la ejecución y están relacionadas con el conocimiento sobre los propios procesos mentales.
b. Implican un uso selectivo de los propios recursos y capacidades disponibles. Para que un estudiante pueda poner en marcha una estrategia debe disponer de recursos alternativos, entre los que decide utilizar, en función de las demandas de la tarea, aquellos que él cree más adecuados.
c. Las estrategias están constituidas de otros elementos más simples, que son las técnicas o tácticas de aprendizaje y las destrezas o habilidades. De hecho, el uso eficaz de una estrategia depende en buena medida de las técnicas que la componen. En todo caso, el dominio de las estrategias de aprendizaje requiere, además de destreza en el dominio de ciertas técnicas, de una reflexión profunda sobre el modo de utilizarlas o, en otras palabras, un uso reflexivo y no sólo mecánico o automático de las mismas.

Teniendo en cuenta los conceptos de los autores citados anteriormente, se pueden destacar elementos fundamentales del proceso de aprendizaje, dentro del cual se hace referencia a una serie de actividades orientadas a la consecución de metas de aprendizaje orientadas a los objetivos de formación que se pretenden alcanzar. El concepto de estrategia está relacionado directamente con acciones enfocadas en el estudiante, de una forma estructurada, y que son, en muchas ocasiones, gestionadas por el estudiante.

En lo que respecta a los diferentes tipos de estrategias, diferentes autores han definido una serie de clasificaciones. Según (Valle, Barca, González, & Núñez, 1999), aun considerando la amplia diversidad de aportes al momento de categorizar las estrategias de aprendizaje, es posible advertir algunas coincidencias respecto a la definición de tres grandes clases de estrategias: estrategias cognitivas, metacognitivas o de control de la comprensión y estrategias de manejo de recursos.

Para (Biggs, Approaches to learning Nature and measurement of, 1994), el aprendizaje resulta de la interrelación de tres elementos clave: la intención (motivación) de quien aprende, el proceso que utiliza (estrategia) y los logros que obtiene (rendimiento). Las estrategias de aprendizaje son procedimientos internos, no observables, de carácter generalmente cognitivo, que ponen en juego los sujetos cuando aprenden y que tienen como fin lograr un plan, un objetivo o una meta. El autor propone un conjunto de categorías que se corresponden con diferentes tipos de estrategias: cognitivas, metacognitivas o de apoyo. Las estrategias cognitivas son procesos por medio de los cuales se obtiene conocimiento. Las estrategias metacognitivas son conocimiento sobre los procesos de cognición u auto administración del aprendizaje por medio de planeamiento, monitoreo y evaluación. Por ejemplo, el estudiante planea su aprendizaje seleccionando y dando prioridad a ciertos aspectos de la matemática para fijarse sus metas. Las estrategias de apoyo permiten al estudiante exponerse a la asignatura que estudian y practicarla, "conversar" la asignatura, explicarse y explicar, intercambiar ideas (Biggs, Learning strategies and learning styles, 1988).

En la tabla 2 se muestran algunas estrategias de aprendizaje cognitivas definidas en (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000).

Tabla 2. Estrategias de aprendizaje cognitivas (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000)

Estrategia de aprendizaje	Descripción
Clarificación/ verificación	Las usa el estudiante para confirmar su comprensión de los temas
Predicción/ inferencia Inductiva	Uso de conocimientos previos, por ejemplo, conceptos, símbolos, lenguajes matemáticos, las representaciones gráficas. Inferir significados en gráficos, ecuaciones, problemas, etc. Se revisan aspectos como ¿qué significado tiene?, ¿Dónde lo usé antes?, ¿cómo se escribe?, ¿con qué se relaciona?
Razonamiento Deductivo	Estrategia de solución de problemas. El alumno busca y usa reglas generales, patrones y organización para construir, entender, resolver. Usa: analogías, síntesis, generalizaciones, procedimientos.

Práctica y memorización	Contribuyen al almacenamiento y retención de los conceptos tratados. El foco de atención es la exactitud en el uso de las ecuaciones, gráficos, algoritmos, procesos de resolución. Se usa: repetición, ensayo y error, experimentación, imitación
Monitoreo	El propio alumno revisa que su aprendizaje se esté llevando a cabo eficaz y eficientemente.
Toma de notas	Se refiere a colocar los contenidos que se desea aprender en una secuencia que tenga sentido. Escribir las definiciones, ideas principales, puntos centrales, un esquema o un resumen de información que se presentó oralmente o por escrito.
Agrupamiento	Clasificar u ordenar material para aprender con base en sus atributos en común.

Las estrategias metacognitivas son conocimiento sobre los procesos de cognición o auto administración del aprendizaje por medio de planeamiento, monitoreo y evaluación. Por ejemplo, el estudiante planea su aprendizaje seleccionando y dando prioridad a ciertos aspectos de la matemática para fijarse sus metas. Las estrategias metacognitivas o de control de la comprensión corresponden a la planificación, control y evaluación de la cognición por parte de los propios estudiantes (Silvestri, 2006). Para González y Tourón, citado en (Silvestri, 2006), son aquellas que posibilitan tanto el conocimiento de los procesos mentales como su control y regulación, con el objetivo de cumplimentar ciertas metas de aprendizaje.

En la tabla 3 se presenta un resumen de algunas estrategias metacognitivas.

Tabla 3. Estrategias de aprendizaje metacognitivas. (Austin, 2011).

Estrategia de Aprendizaje	Descripción
Organizadores previos	Hacer una revisión anticipada del material por aprender en preparación de una actividad de aprendizaje.
Atención dirigida	Decidir por adelantado atender una tarea de aprendizaje en general e ignorar detalles.
Atención selectiva	Decidir por adelantado atender detalles específicos que nos permitan retener el objetivo de la tarea.
Autoadministración	Detectar las condiciones que nos ayudan a aprender y procurar su presencia.
Autoevaluación	Verificar el éxito de nuestro aprendizaje según nuestros propios parámetros, de acuerdo con nuestro nivel.

Las estrategias de apoyo son mecanismos o procedimientos que facilitan el estudio. Sensibilizar hacia el aprendizaje. Optimizar las tareas de estudio y aprendizaje. Las estrategias de apoyo permiten al estudiante exponerse a la asignatura que estudia y practicarla, "conversar" la asignatura, explicarse y explicar, intercambiar ideas. Según (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000) las estrategias de apoyo ejercen un impacto directo sobre la información que se va a aprender, y su papel es mejorar el nivel de funcionamiento cognitivo del alumno, habilitando una disposición afectiva favorable. En la tabla 4 se muestran algunas estrategias de apoyo al proceso de aprendizaje.

Tabla 4. Estrategias de apoyo. (Austin, 2011).

Estrategia de apoyo	Descripción
Cooperación	Trabajar con uno o más compañeros para obtener retroalimentación.
Aclarar dudas	Preguntar o discutir significados con los compañeros o con el profesor.
Logro	Querer ser premiado por su desempeño. Obtener la mejor nota. Querer ser reconocido como el mejor en algún aspecto

5.2.4 Estrategia de Enseñanza

La enseñanza puede ser descrita como un proceso continuo de negociación de significados, de establecimiento de contextos mentales compartidos, fruto y plataforma a la vez de este proceso de negociación (Coll & Solé, La interacción profesor/alumno en el proceso de enseñanza aprendizaje, 1990). Se pretende que el rol de docente sea el de mediador o intermediario entre los contenidos del aprendizaje y la actividad constructiva que despliegan los alumnos para alcanzarlos.

De acuerdo con (Coll, Significado y sentido del aprendizaje escolar, 1990) el profesor gradúa la dificultad de las tareas y proporciona al alumno los apoyos necesarios para afrontarlas, pero esto solo es posible porque el alumno en sus reacciones indica constantemente al profesor sus necesidades y comprensión de la situación. Esto significa que tanto profesores como alumnos gestionan de manera conjunta la enseñanza y el aprendizaje en un proceso de participación guiada.

Teniendo en cuenta el propósito de formación de esta estrategia de enseñanza, se deben tener en cuenta aspectos motivacionales que influyen en el aprendizaje. Se tiene la necesidad de promover en el estudiante el interés por aprender. A continuación en la tabla 5 se relacionan algunos principios motivacionales y de enseñanza (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000).

Tabla 5. Descripción de principios motivaciones y de enseñanza. (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000)

Principio motivacional y enseñanza	**Descripción**
En relación con la forma de presentar y estructurar la tarea	• Activar la curiosidad y el interés del alumno en el contenido del tema a tratar o la tarea a realizar • Mostrar la relevancia del contenido o la tarea para el alumno.
En relación con la forma de realizar la actividad en el contexto de la clase	• Organizar la actividad en grupos cooperativos; la evaluación individual dependerá de los resultados grupales. • Dar el máximo de opciones posibles de actuación para facilitar la percepción de autonomía.
En relación con el modelado que el profesor puede hacer de la forma de afrontar las tareas y valorar los resultados.	• Organizar las evaluaciones a lo largo del curso, de forma que se evite la comparación de unos con otros y se acentúe la propia comparación para maximizar la constatación de los avances.

En un trabajo de Hoostein (1995, citado por Brophy), se pedía a los profesores de historia a nivel secundaria que listaran las estrategias que empleaban para motivar a sus alumnos. Las diez estrategias que se mencionaron con más frecuencia, fueron:

1. Trabajar con simulaciones históricas (dramatizaciones*, role playing*) en las que los estudiantes representan personajes históricos.
2. Organizar proyectos que se traducen en la creación de productos concretos.
3. Realizar juegos con los estudiantes, como una manera de revisar material contenido en las pruebas.
4. Relacionar la historia con eventos actuales o con la vida de los alumnos.
5. Solicitar a los alumnos que lean novedades históricas.
6. Formular preguntas que provoquen la reflexión.
7. Invitar como conferencistas a personas de la comunidad.

8. Proyectar videos y películas históricas.
9. Organizar actividades de aprendizaje cooperativo.
10. Proporcionar experiencias de participación activa y manipulaciones

La realización de actividades académicas cooperativas permite que los individuos establezcan metas que son benéficas para sí mismo y para los demás miembros del grupo, buscando así maximizar tanto su aprendizaje como el de los otros. El equipo trabaja junto hasta cuando todos los miembros del grupo han entendido y completado la actividad con éxito (Marquez, 2001).

El docente puede utilizar el enfoque del aprendizaje cooperativo para promover que sus estudiantes:

- Se sientan involucrados en relaciones con compañeros que se preocupan por ellos y los apoyan.
- Sean capaces de influir en las personas con quienes están involucrados.
- Disfruten el aprendizaje.

Se puede definir las estrategias de enseñanza como los recursos o procedimientos utilizados por el agente de enseñanza para promover aprendizajes significativos (West, Farmer, & Wolff, 1991). Las estrategias de enseñanza promueven los aprendizajes significativos a partir de contenidos curriculares en donde se enfatiza la planeación, el diseño y la elaboración de los contenidos para que los estudiantes aprendan.

En la tabla 6 se presentan algunas de las estrategias de enseñanza que se emplean con frecuencia para facilitar el aprendizaje significativo. Estas estrategias han demostrado en diversas investigaciones (Díaz & Lule, Efectos de las estrategias preinstruccionales en alumnos de secundaria de diferentes niveles socieconómicos, 1978), (Mayer, 1984), (West, Farmer, & Wolff, 1991) su efectividad al ser introducidas en la dinámica de la enseñanza. Las principales estrategias de enseñanza, son:

Tabla 6. Estrategias de Enseñanza. (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000)

Estrategias de enseñanza	Descripción
Objetivos o intenciones	Son enunciados que describen las actividades de aprendizaje de determinados contenidos curriculares, así como los efectos esperados que se pretenden

	conseguir en los alumnos.
Ilustraciones	Comunican conceptos de tipo visual o espacial, eventos que ocurren de manera simultánea, y también para ilustrar procedimientos o instrucciones procedimentales (Hartley, 1985).
Resúmenes	Es una versión breve del contenido que habrá de aprenderse, donde se enfatizan los puntos sobresalientes de la información.
Preguntas intercaladas	Se le plantean al alumno a lo largo de la situación de enseñanza. También son conocidas como preguntas adjuntas o insertadas (Rickards & Denner, 1978) , que se van insertando en partes importantes del texto.
Analogías	Es una proposición que indica que una cosa o evento es semejante a otro (Curtis & Reigeluth, 1984). Se emplea cuando la información que se ha de aprender se presta para relacionarla con conocimientos antes aprendidos.
Mapas conceptuales	Jerarquía de diferentes niveles de generalidad o inclusividad conceptual, estructurada por varias proposiciones conceptuales (Novak & Gowin, 1988). Estos permiten la negociación de significados entre el profesor y alumnos, se pueden precisar y profundizar los significados referidos a los contenidos curriculares.

De acuerdo con el momento de uso y presentación, las estrategias pueden clasificarse, en:

- Preinstruccionales: por lo general preparan y alertan al estudiante en relación a qué y cómo va a aprender, y le permiten ubicarse en el contexto del aprendizaje pertinente. Por ejemplo: Objetivos y el organizador previo.
- Coinstruccionales: apoyan los contenidos curriculares durante el proceso mismo de la enseñanza. Cubren funciones como: detección de la información principal; conceptualización de contenidos; delimitación de la organización, entre otras.
- Posinstruccionales: Se presentan después del contenido que se ha de aprender, y permiten al alumno formar una visión sintética, integradora e incluso crítica del material.

De acuerdo con (Barrios, 1992), algunos prototipos de actividades de entrenamiento que el docente puede emplear, son:

- Exposición y actividades guiadas
- Discusión y trabajo en equipos cooperativos
- Ilustración y análisis de casos concretos

- Revisión y criticas de textos
- Resolución, autoevaluación y análisis individual / grupal de ejercicios, cuestionarios, trabajos.
- Elaboración de materiales y planes de clase apoyados con estrategias de enseñanza y aprendizaje.

5.2.5 Aprendizaje Significativo

Este proyecto pretende generar una serie de procesos activos en la construcción de las capacidades de los estudiantes, es necesario un sustento que permita una función por parte del profesor para orientar las actividades que permitan a los estudiantes la adquisición de formas que resulten esenciales para fortalecer su competencia académica, profesional o personal.

Desde ese punto de vista, este trabajo se apoya en la postura constructivista, en la cual según (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000), se postula la existencia y prevalencia de procesos activos en la construcción del conocimiento: habla de un sujeto cognitivo aportante que claramente rebasa a través de su labor constructiva lo que le ofrece su entorno.

La concepción constructivista del aprendizaje se sustenta en la idea de que la finalidad de la educación que se imparte en las instituciones educativas es promover los procesos de crecimiento personal del alumno en el marco de la cultura del grupo al que pertenece. Estos aprendizajes no se producirán de manera satisfactoria, a no ser que se suministre una ayuda específica a través de la participación del alumno en actividades intencionales, planificadas y sistemáticas, que logren propiciar en éste una actividad mental constructiva (Coll, Sicología y Curriculum, 1988). El aprendizaje significativo depende de las motivaciones, intereses y predisposición del aprendiz.

De acuerdo con (Coll, Significado y sentido del aprendizaje escolar, 1990) la concepción constructivista se organiza en torno a tres ideas fundamentales:

- El alumno es el responsable último de su propio proceso de aprendizaje.
- La actividad mental constructiva del alumno se aplican a contenidos que poseen ya un grado considerable de elaboración.
- La función docente es unir los procesos de construcción del alumno con el saber colectivo culturalmente organizado.

En opinión de (Resnick, 1987), la forma como la institución escolar busca fomentar el conocimiento con frecuencia contradice la forma como se aprende fuera de ella. El conocimiento fomentado en la escuela es individual, fuera es compartido; en la escuela se manipulan símbolos libres de contexto, mientras que en el mundo real se trabaja y razona sobre contextos complejos. En ese sentido, en los planteles educativos se debe formar de acuerdo con las necesidades de la vida real, en las cuales las prácticas deben ser contextualizadas y significativas.

El origen de la Teoría del Aprendizaje Significativo está en el interés por conocer y explicar las condiciones y propiedades del aprendizaje, que se pueden relacionar con formas efectivas y eficaces de provocar de manera deliberada cambios cognitivos estables, susceptibles de dotar de significado individual y social (Ausubel, Psicología educativa. Un punto de vista cognoscitivo, 1976). Pone el énfasis en lo que ocurre en el aula cuando los estudiantes aprenden, en la naturaleza de ese aprendizaje, en las condiciones que se requieren para que éste se produzca, en sus resultados y, consecuentemente, en su evaluación (Ausubel, Psicología educativa. Un punto de vista cognoscitivo, 1976). La Teoría del Aprendizaje Significativo aborda todos y cada uno de los elementos, factores, condiciones y tipos que garantizan la adquisición, la asimilación y la retención del contenido que la escuela ofrece al alumnado, de modo que adquiera significado para el mismo.

Es crucial también que el que aprende sea crítico con su proceso cognitivo, de manera que manifieste su disposición a analizar desde distintas perspectivas los materiales que se le presentan, a enfrentarse a ellos desde diferentes puntos de vista, a trabajar activamente por atribuir los significados y no simplemente a manejar el lenguaje con apariencia de conocimiento (Ausubel, Adquisición y retención del conocimiento. Una perspectiva cognitiva, 2002).
Aprendizaje significativo es el proceso que se genera en la mente humana cuando subsume nuevas informaciones de manera no arbitraria y sustantiva y que requiere como condiciones: predisposición para aprender y material potencialmente significativo que, a su vez, implica significatividad lógica de dicho material y la presencia de ideas de anclaje en la estructura cognitiva del que aprende. Durante el aprendizaje significativo el alumno relaciona de manera no arbitraria y sustancial la nueva información con los conocimientos y experiencias previas que ya posee en su estructura de conocimientos o cognitiva (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000).

En (Shuell, 1990), se afirma que el aprendizaje significativo ocurre en una serie de fases, que dan cuenta de una complejidad y profundidad progresiva y distingue tres fases:

- Fase inicial de aprendizaje
- Fase intermedia de aprendizaje
- Fase terminal del aprendizaje

Con base en lo anterior, para el propósito de este trabajo se presentan los elementos más simples y fundamentales de los contenidos y después se trabaja con información más detallada y cada vez más compleja. Según (Coll & Rochera, Estructuración y organización de la enseñanza: Las secuencias de aprendizaje, 1990), esto proporciona un aprendizaje en espiral, puesto que cada vez que se elabora uno de los elementos iniciales, se vuelve al punto de partida con el fin de enriquecer y ampliar el plano del conjunto.

En la tabla 7 se muestran los contenidos que se enseñan en los currículos de todos los niveles educativos y los cuales pueden agruparse en tres áreas (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000):

- Los contenidos declarativos.
- Los contenidos procedimentales.
- Los contenidos actitudinales.

Tabla 7. Niveles de contenidos educativos. (Díaz & Hernandez, Estrategias Docentes para un Aprendizaje Significativo, 2000)

Contenido	Descripción
El saber qué o conocimiento declarativo	Hace referencia al conocimiento de datos, hechos, conceptos y principios. Estos proporcionan información verbal y que los alumnos deben aprender de forma literal.
El saber hacer o saber procedimental	Se refiere a la ejecución de procedimientos, estrategias, técnicas, métodos. Es de tipo práctico, porque está basado en la realización de varias acciones u operaciones. Ejemplos son la elaboración de resúmenes, ensayos o gráficas estadísticas, el uso de algoritmos u operaciones matemáticas, la elaboración de mapas conceptuales o el uso de alguna herramienta informática. La enseñanza de procedimientos se transfiere a través de la participación guiada y con la asistencia continua, pero paulatinamente decreciente del profesor, la cual ocurre al mismo tiempo que se genera la creciente mejora en el manejo de procedimientos por parte del alumno.

Los contenidos actitudinales	Conjunto de constructos que median nuestras acciones y que se encuentran compuestas de tres elementos básicos: un componente cognitivo, un componente afectivo, y un componente conductual (Bednar & Levie, 1999). Técnicas eficaces para trabajar con procesos actitudinales, son las técnicas participativas (*role-playing*), las discusiones y técnicas de estudio activo, las exposiciones y explicaciones de carácter persuasivo (conferencista de prestigio o influencia) e involucrar alumnos en la toma de decisiones.

5.3 Antecedentes

Desde el momento cuando Watts Humphrey presentó el PSP en su libro *A Discipline for Software Engineering*, se han llevado a cabo diversos estudios relacionados con PSP.

Se han realizado investigaciones y se presentan informes en los cuales se describen las experiencias de su implementación en diversos contextos de aplicación (Ferguson, Humphrey, Khajenoori, & Macke, 1997), (Humphrey W. , 1995). Además, se han realizado algunos estudios e investigaciones del efecto que genera la implementación de PSP durante en algunos cursos de pregrado y posgrado en diversas instituciones (Hayes & Over, 1997), (Hayes W. ”., 1998), (Wesslén, 2000), así como los intentos de evaluar su impacto posterior (Rosca, Li, Moore, Stephan, & Weiner, 2001).

También se tienen informes sobre el impacto de la aplicación de PSP en la industria [13], y numerosos estudios relacionados con el uso del PSP en la academia y en la enseñanza, tales como los de (Börstler, Carrington, & Hislop, 2002) (Runeson P. , 2001). También se propone utilizar PSP como contexto para los experimentos de Ingeniería de Software (Honig, 2008).

A continuación se describen los resultados obtenidos de la implementación de PSP en diferentes universidades del mundo.

5.3.1 Lund University - Suecia

Lund University
Resumen Se comparó el desempeño de estudiantes de primer año con poca experiencia en programación, con estudiantes de posgrado y con personas que trabajan en la industria del software, con experiencia en programación. La hipótesis se centró en que existen diferencias entre el

desempeño de estudiantes de posgrado y las personas que trabajan en la industria, y por otro lado que existen diferencias de desempeño entre los estudiantes de posgrado y los de primer año. Se investigó si la precisión de la estimación, la densidad de defectos y la productividad, mejoran en la medida en que los estudiantes aplican progresivamente las técnicas de PSP. También, se analizan las diferencias entre el consumo de tiempo, la productividad y el número de defectos.

Resultados

En cuanto al rendimiento de los estudiantes, se identificaron diferencias en su desempeño; a nivel del tiempo se encontró que los estudiantes de primer año desarrollan sus trabajos en un 47% más de tiempo que los estudiantes de posgrado. En cuanto al número de defectos insertados, esta cantidad no difiere entre los dos grupos. La densidad de los defectos es significativamente mayor para estudiantes de primer año en 4 de los 9 ejercicios propuestos. En cuanto al número de defectos insertados por hora es bajo. Esto significa que la diferencia más significativa está asociada con la métrica del tiempo. En cuanto al tamaño del producto, se encuentra que los estudiantes de posgrado se reporta que estos, son más eficientes en términos de líneas de código en aproximadamente 19% por encima que los estudiantes de primer año. Una razón de esta situación se encuentra asociada a que los requisitos funcionales para los estudiantes de posgrado son más complejos.

Los estudiantes de primer año tienden a realizar más preguntas sobre los problemas de programación, mientras que los estudiantes de postgrado se centraron en las partes del proceso. Un aspecto que se identificó es que el número de estudiantes varió en ambos grupos, el grupo de graduados se redujo en un 14% y el grupo de estudiantes de primer año se redujo a un 45%, las razones fundamentalmente fueron de desmotivación en los temas de PSP fueron los que incidieron en la deserción. En cuanto a las estimaciones del tamaño y del esfuerzo, a medida que se avanza en los niveles de PSP, se encuentra que son cada vez mejores estas estimaciones. Así mismo, se encuentra que la productividad se incrementa para cada nivel de PSP.

Observaciones

No se hace referencia a una metodología de enseñanza durante la impartición de los temas de PSP. Los estudiantes deben diligenciar las formas y registros definidos en el proceso. No se hace énfasis en una herramienta en particular, ni tampoco a una estrategia de enseñanza.

5.3.2 Zagreb University – Croacia

Zagreb University
Resumen El objetivo de este proyecto era propiciar a los estudiantes de pregrado una experiencia práctica para definir y medir su proceso personal de desarrollo de software. Los resultados se presentan teniendo en cuenta que los estudiantes son programadores sin experiencia. Los aspectos conceptuales más avanzados de PSP no fueron tenidos en cuenta para este trabajo, mientras que los aspectos de análisis, diseño y revisiones fueron mantenidos. Los estudiantes deben aplicar el proceso de referencia PSP0 y el proceso de planificación PSP1. Esto supone el registro de tiempo y defectos, el tamaño de la medición, y la producción de informes de resumen. Para la enseñanza de PSP se han adaptado dos registros (registro de tiempo y de defectos), también tienen que diligenciar el resumen del plan de proyecto. El objetivo es que el estudiante aprenda la forma de trabajar con un proceso bien definido y cómo analizar los datos de medición para mejorar la estimación, la prevención y la eliminación de defectos. También, proporcionar una idea de la utilización de procesos de desarrollo de software y cómo puede ayudar a predecir, medir, analizar y mejorar su propio proceso. Los estudiantes analizan individualmente lo registrado en sus datos y llegar así a ciertas conclusiones sobre sus propios procesos. Deben comentar cómo se adaptó PSP para conseguir conocimiento sobre su propio proceso de desarrollo y si es posible un mejoramiento después del curso.
Resultados Con relación a los resultados cualitativos, se evidenció una disminución en la tasa de defectos durante las fases de compilación y pruebas. En cuanto al tiempo invertido en cada una de las fases del proceso de desarrollo, se observa que en los ejercicios iniciales los estudiantes estuvieron la mayor cantidad de tiempo en las fases de compilación y pruebas. En los ejercicios finales se evidencia que dedicaron mayor cantidad de tiempo en la fase de diseño y disminuyeron sustancialmente su tiempo en la fase de pruebas. Los resultados no cuantificables se obtienen al entrevistar al estudiante, los cuales expresan la gran ventaja de la implementación del método. Se afirma que el método ayuda a entender la idea y que la experiencia práctica, de acuerdo con un proceso bien definido, permiten entender cómo recoger y analizar los datos de medición, y cómo estos afectan a su propio proceso. Según su experiencia, el método es simple como para permitir a los estudiantes aprender paralelamente a programar, al mismo tiempo que se adapta PSP en los siguientes niveles.

Observaciones

La metodología del curso incluye presentaciones orales de los conceptos básicos de PSP, la introducción de los formularios y registros necesarios y la presentación de la herramienta de apoyo para PSP que tienen que ser utilizadas en el proceso. No se hace referencia a una estrategia en particular de enseñanza y aprendizaje.

5.3.3 Purdue University –Indiana, USA

Purdue University
Resumen Los estudiantes que tomaron el curso de PSP son estudiantes que han completado cursos básicos de programación. Para este ensayo inicial de la enseñanza del PSP se consideraron los registros, de tiempo, trabajo y defectos, para que los estudiantes diligencien los formularios durante las tareas de programación. Estas formas son proporcionadas en medios electrónicos.
Resultados En la evaluación de los registros presentados por los estudiantes, sólo el 40% de los alumnos de la muestra presentaron datos utilizables. En el registro de tiempo, algunos estudiantes no calcularon el tiempo total gastado en el programa, o no registraron sus tiempos de interrupción. El registro de las líneas de código no se realizaba correctamente en muchos casos, pues los espacios para registrarse eran con frecuencia vacíos o estaban incorrectos. Varios estudiantes informaron los tiempos en horas en lugar de minutos, a pesar del hecho de que se les informaba en la parte superior del formulario, que lo diligenciaran en minutos. Después de aplicar una encuesta a los estudiantes, estos consideraron que entienden los temas PSP y fueron capaces de completar los registros, a pesar de que no estaban muy convencidos de éstos. También indican que los datos de sus registros no son exactos. Hay una negativa mucho más alta que indica que los estudiantes no reconocen el valor de PSP en su formación. Un problema con la incorporación de PSP en un curso, es que los temas de programación tienen que ser cubiertos en la misma profundidad, y cualquier tiempo utilizado en la clase en la explicación de PSP quita tiempo antes dedicado a temas exclusivamente de programación. En el trabajo desarrollado se identificó que algunos estudiantes no estaban diligenciando correcta y

completamente los registros, pues consideran que los mismo implican demasiado tiempo como para realizarse con la seriedad esperada.

Observaciones

No se tiene definida una estrategia de aprendizaje concreta, y el estudiante debe mantener un registro de cada una de las actividades propuestas en PSP y con base en ello se hace una serie de análisis estadístico sobre su desempeño en cada una de las prácticas propuestas.

5.3.4 Loyola University –Chicago, USA

Loyola University
Resumen Este trabajo informa sobre un experimento de TSP con un grupo de posgrado de Ingeniería de Software. Los estudiantes tienen experiencia en construcción de software y conocen los conceptos básicos de desarrollo de software en equipo. Los equipos están conformados por personas con culturas y procedencias diferentes, y un papel que se asigna de acuerdo con los roles de TSP; los productos se deben entregar a un cliente real. Cada estudiante debe trabajar como programador. El equipo debe completar dos ciclos de desarrollo de acuerdo con un cronograma. Deben empezar a trabajar con una breve declaración del problema y seleccionar las funciones que se ejecutarán en cada ciclo; completan una o más fases del TSP cada semana y todas las fases se realizan de forma secuencial en el desarrollo de cada ciclo. La dinámica de trabajo del equipo es definida por agendas de las reuniones que requieren el registro del tiempo en minutos y el registro las actividades realizadas para realizar el seguimiento al desarrollo del proyecto. Los scripts de TSP dan estructura a las actividades semanales del equipo, en los cuales se deben plasmar los resultados obtenidos.
Resultados Los defectos se suman para todo el ciclo como una medida de alto nivel de la calidad del producto. La calidad del producto al final del ciclo dos, es de unos veinte errores por cada mil líneas de código. Se observa una gran variación en la tasa de los defectos de los diferentes

equipos, esto puede estar influenciado tanto por el tiempo que tienen los equipos de trabajo para realizar las pruebas, como por la poca cultura de los equipos para registrar errores y así obtener resultados completos. En términos generales, la calidad del producto es razonable en comparación con los equipos del mundo real.

Cuando se utiliza TSP con sólo unas pocas semanas de entrenamiento, los equipos de estudiantes lograr una productividad en desarrollo de software similar a los del mundo real de las organizaciones.
Los resultados del tiempo en cada una de las fases de TSP durante todo el ciclo de vida, se observa que un 15% está dedicado a las pruebas y que las fases que más consumen tiempo son las de Desarrollo y la de administración, la cual incluye las reuniones de equipo; un 42% está dedicado a estas dos fases. A continuación se muestran los tiempos de cada fase. Administración en cada fase del proyecto 19,3%, Lanzamiento 3,3%, Estrategia y planeación 9,3%, Requerimientos 8,2, Diseño 16,5%, Implementación 22,8%, Pruebas 15,1% y Postmorten 5,5%.

Los estudiantes que completan el curso desarrollan un aprecio por el proceso y aceptan el análisis de datos como un elemento central y fundamental de desarrollo de software. Ellos adquirieron una comprensión de las métricas e informaron de sus avances semanales, la superación de algunas de sus tendencias a trabajar de forma rápida, sin tener en cuenta las consecuencias y los resultados no necesariamente óptimos.

Observaciones
En este trabajo, los equipos de trabajo recopilaron la información de su proceso de desarrollo y posteriormente se analizaba semanalmente su desempeño. Cada equipo debe gestionar los formularios definidos en TSP. No se evidencia una estrategia de aprendizaje en el curso.

5.3.5 Embry-Riddle Aeronautical University

Embry-Riddle University
Resumen En esta investigación se aplicaron los conceptos de proceso de desarrollo de software en dos cursos básicos de programación. En el primer curso se trabajó con la gestión del tiempo y la planificación de las actividades. En el segundo curso de programación se presentaron oficialmente las actividades de PSP, los estudiantes llevaron a cabo la planificación para cada proyecto, el uso

de datos históricos para la estimación del tamaño, el esfuerzo y los defectos. También utilizaron formularios para la planificación, el registro de tiempo y los horarios de trabajo.

Este trabajo incluye una descripción de los objetivos del proyecto, un análisis de las actividades realizadas por los estudiantes, una explicación de cómo las actividades se integraron en el plan de estudios, una descripción de los datos obtenidos y una evaluación del impacto del proyecto. El experimento se llamó: *Doing Quality Work* – DQW/PSP y pretendía que los estudiantes desde el principio de su formación, entendieran e incorporaran el concepto de calidad, por medio de los elementos fundamentales de PSP. En el primer curso de programación se trabajó con la gestión del tiempo y la planificación, que se combinaron con el material de fundamentos de programación. Los estudiantes debían utilizar varios tipos de documentos para el registro de tiempo, horarios de trabajo y documentar el resumen del plan. También realizaron tareas de estimación del esfuerzo (tiempo). Para las tareas de Programación, los estudiantes registraron datos estimados y datos reales, tanto para el tamaño del esfuerzo (tiempo en minutos) y del programa (líneas de código).

En el segundo curso de programación se aplicaron las actividades de PSP, los estudiantes realizaron la planeación de sus proyectos, también se incluye el uso de datos históricos para realizar estimaciones del tamaño, el esfuerzo y los defectos.

Resultados

La mayoría de los estudiantes tuvieron problemas para el registro del tiempo dedicado en cada fase del proceso de desarrollo. Se concluyó que los estudiantes de primer año no disfrutaban el estar realizando registros de forma intensiva. Los estudiantes indicaron alguna resistencia a la inclusión de PSP en el primer curso de programación.

Como resultado de este proyecto se encontró que los estudiantes de primer año no disfrutaban el estar realizando registros de forma intensiva. Muchos de ellos expresaron su preocupación acerca de sacrificar el tiempo necesario para estudiar programación y para llevar a cabo las actividades de PSP. También se muestra que los estudiantes tenían problemas de distinguir entre los tiempos dedicados a cada etapa del ciclo de vida.

Observaciones

Este trabajo presenta una estrategia de enseñanza en la cual se pretendía que los estudiantes, de forma incremental, se apropiaran de los conceptos de PSP, así como también que conocieran la importancia de aplicar técnicas disciplinadas en el proceso de desarrollo de software. No se hace

referencia a una herramienta de apoyo en particular.

5.3.6 Birla Institute of Technology and Science – India

Birla Institute of Technology and Science
Resumen Este trabajo describe la experiencia de la enseñanza de PSP en un curso de Ingeniería de Software. El objetivo era crear en los estudiantes buenos hábitos de desarrollo de software y motivarlos para que el desarrollo de software sea percibido como una disciplina sistemática, más que una actividad de prueba y error. Los estudiantes registraron los datos durante el desarrollo de sus programas, presentaron informes y los programas de software. A los estudiantes se les dieron conferencias para explicar los métodos y modelos matemáticos que se utilizan para realizar los ejercicios de PSP. También registraron datos durante el desarrollo de estos programas y presentaron sus informes, junto con sus programas.
Resultados Se muestra en esta experiencia que, aunque a la mayoría de los estudiantes no les gustaba llenar todas los formas y recoger todos los datos, admitieron que valía la pena hacerlo, especialmente el registro de tiempo y de defectos; algunos manifestaron que no estaban motivados y que nunca quisieron usar PSP. Se aprendieron lecciones en dos aspectos: mantener la motivación de los estudiantes y centrarlos en las cosas importantes del proceso. Los resultados muestran que en cuanto a las líneas de código, la tendencia es a incrementar la productividad en cantidad de líneas. En cuanto al tiempo se observó que los estudiantes también mejoran en este aspecto, teniendo en cuenta que los problemas propuestos son progresivamente más complejos. En cuanto la densidad de defectos, el número total de defectos encontrados durante el desarrollo por cada 1000 líneas de código (KLOC) disminuyó significativamente con el tiempo. Para más de la mitad de los estudiantes resultó que la experiencia de PSP puede ser muy útil en la comprensión de la mejora del proceso de software. Sentían que realmente aprendían acerca de la forma como tenían que trabajar, y se podía ver una mejora significativa en su productividad. Aunque a la mayoría de ellos no les gustaba llenar todos las formas y recoger todos los datos, admitieron que valía la pena hacerlo, especialmente los registros de tiempo y los registros de

defectos. Hubo otros que no estaban motivados y que en realidad nunca quisieron usar PSP

Observaciones

Los estudiantes deben diligenciar los formularios básicos de PSP durante el desarrollo del curso. No se evidencia una estrategia de aprendizaje que de soporte al proceso de formación de los estudiantes. En cuanto a las herramientas, no se hace referencia al uso de alguna en particular.
Se aprendieron lecciones importantes en dos aspectos: mantener la motivación de los estudiantes y mantenerlos centrados en las cosas importantes.

5.3.7 Universidad Carlos III – Madrid

Universidad Carlos III
Resumen Esta investigación se realizó con alumnos de primer año de Ingeniería Informática. Por medio de la revisión de conceptos teóricos de PSP y el desarrollo de prácticas basadas en una evaluación continua, los estudiantes entienden qué es un proceso de software, adquieren habilidades para conocer cómo desarrollan personalmente sus programas en forma disciplinada y recolectan métricas que pueden incorporarse como mejoras. El curso se planificó a través de una clase teórica y una práctica a la semana. Se impartieron conceptos de PSP referentes a los niveles PSP0, PSP0.1, PSP1 y PSP1.1. Durante el curso no se profundizó en aspectos de programación. Para la realización de algunas prácticas se entregó material de aprendizaje adicional para completarlas con éxito. El trabajo realizado se centró en desarrollar habilidades concretas en Ingeniería del Software que involucran el registro y estimación de tamaños y tiempos a medida que se elaboran las prácticas semanales. El curso se enfocó en el desarrollo de competencias para entender el proceso de software y los estudiantes perciben que la tarea de desarrollar software es un proceso complejo, con un conjunto de actividades de ingeniería que requiere técnicas para construir software, que van más allá de sus competencias iniciales sobre codificación de programas.
Resultados Con la introducción de PSP en el primer año de grado, los estudiantes van adquiriendo disciplina en el proceso de gestión de su tiempo y logran gestionar sus trabajos prácticos, aplicando las

tareas básicas de Ingeniería de Software desde etapas tempranas de formación (Bermon, Fernández, & Sanchéz, 2009) . El promedio final de las evaluaciones presentó porcentajes altos en la realización correcta del registro de tiempos y defectos que reflejan que sus elementos fueron comprendidos. Los conceptos de estimaciones planificación y seguimiento presentan valores medios, debido a su complejidad y a los diferentes errores cometidos en la realización de los programas (Bermon, Fernández, & Sanchéz, 2009).

El promedio final de evaluaciones presenta porcentaje altos en la realización correcta del registro de tiempos y defectos que reflejan que sus conceptos fueron comprendidos. Los conceptos de estimaciones, y planificación y seguimiento presentan valores medios debido a su complejidad y los diferentes errores cometidos en su realización. El análisis realizado manifiesta porcentajes superiores al 50% en los aspectos evaluados que hacen favorable la incorporación de conceptos de PSP en nuestro curso de primero de grado (Bermon, Fernández, & Sanchéz, 2009).

Observaciones

La realización de este curso presenta la estructuración de una estrategia enseñanza en la cual se planearon cada una de las sesiones del curso, así como también la elaboración de material de apoyo. Como herramienta de soporte se utiliza la herramienta desarrollada por el SEI.

5.3.8 Universidad Politécnica de Madrid

Universidad Politécnica de Madrid
Resumen Este trabajo presenta un estudio de caso de la enseñanza de TSPi para alumnos de Cuarto año en la Escuela de Ciencias de la Computación. Es un caso de estudio relacionado con la evolución de la estimación, la calidad y la productividad de las mejoras de estos equipos. Se seleccionaron como métricas para analizar el impacto en los estudiantes: el tamaño, el esfuerzo, productividad, los costos y la densidad de defectos. Estas métricas fueron elegidas para analizar la evolución del rendimiento de los equipos a través del desarrollo del proyecto. Los estudiantes fueron capacitados antes de aplicar el TSPi en los siguientes temas, como: el trabajo en equipo, factores humanos, la toma de decisiones técnicas, definición de las funciones y actividades. Se enseñaron los conceptos fundamentales de TSPi como la descripción de las actividades del proceso de desarrollo, las técnicas de estimación y de recolección de datos.

Los equipos realizaron durante el curso dos ejercicios. El primero estuvo relacionado con factores humanos, cada equipo estableció los roles necesarios para desarrollar un proyecto de software. El segundo ejercicio consistió en el desarrollo de un proyecto de software utilizando TSPI. Cada equipo desarrolló el mismo proyecto bajo las siguientes situaciones: el mismo tiempo, dos ciclos de desarrollo y los mismos instructores. Durante el segundo ciclo, los estudiantes tienen la visión de un producto orientado a procesos y produjo una versión más amplia del producto.

Resultados

En cuanto a las mejoras de habilidades y experiencia de los equipos, se encontró que durante el proceso de desarrollo el tiempo de esfuerzo de las fases de Administración de la configuración y de Planeación son mayores en el primer ciclo; las actividades de Requerimientos, Diseño y Requerimientos son similares para ambas fases; sólo en la fase de Implementación el tiempo es mayor en el segundo ciclo, pues ya los equipos tienen la experiencia del primer ciclo.

En aspectos de calidad, se encuentra deficiencia en cuanto a los datos recolectados; sólo se analizó la información del 68% de los equipos. Se identificó que los equipos mejoran para el segundo ciclo en las estimaciones de tiempo, esfuerzo, tamaño del producto y la densidad de los defectos. Para el segundo ciclo se identificaron como beneficios reales del equipo, una mayor motivación, compromiso, nivel de rendimiento y disminuyeron los tiempos de desarrollo.

Este estudio muestra que la formación en TSPi tiene un impacto positivo para conseguir mejores estimaciones, la reducción de costos, la mejora de la productividad y la disminución de la densidad de defectos. Llega a la conclusión que los estudiantes necesitan aprender sobre los mecanismos de participación efectiva en un equipo. Esto es importante, porque muchos de los problemas relacionados con el trabajo en equipo se relacionan con habilidades de comunicación.

Observaciones

La metodología del curso estaba orientada al diligenciamiento de cada uno de los registros por parte de los equipos. No se tiene definida una estrategia de aprendizaje.

5.3.9 Universidad Nacional de la Matanza – Argentina

Universidad Nacional de la Matanza

Resumen

El trabajo se realizó en el curso de Programación Avanzada a estudiantes de pregrado. Se adopta una metodológica de trabajo que le permita al alumno, de forma temprana, ser capaz de medir sus habilidades para producir software de calidad y estimar su tiempo de desarrollo. Además de mostrar la relación de los contenidos de formación con las demandas laborales del medio.

Se explican técnicas PSP junto a temas como: metodologías de desarrollo del software, métodos ágiles, XP y *Scrum*. Todos estos temas están relacionados con las técnicas PSP y se explican desde la perspectiva de un curso de Programación y no la de un curso de Ingeniería de Software. El profesor mide todo lo que los alumnos hacen. Esta medición es comparada con las propias mediciones que los alumnos realizan y se controla su exactitud. El registro final es una valoración de ambas fuentes. En el aula se usa la programación de a pares, que permite medir la evolución de la competencia general de los trabajos en grupo. Los alumnos usan las técnicas PSP en cada uno de los ejercicios de las guías de trabajos prácticos.

Resultados

El uso de PSP permitió evaluar la calidad individual del egresado o de cada alumno en un curso particular. Además, es útil para medir el rendimiento desde el inicio de un curso hasta su finalización, viendo como mejoraron sus registros. También sirve para incentivar al alumno para su mejora continua.

El uso de PSP permitió a los alumnos: Aprender a administrar el tiempo entre las distintas fases del desarrollo de software, entender la relación entre el tamaño de los programas que escriben y el tiempo que les toma desarrollarlos, aprender a tomar compromisos que puedan cumplir, preparar un plan ordenado para realizar su trabajo, y establecer una base para realizar seguimiento de un trabajo (Aubin, Blauzic, & Dejean, 2011). El alumno lleva sus propios registros individuales y de a pares. Este acopio de datos le sirve para realizar su autoevaluación y tener evidencias claras de cómo va mejorando su rendimiento a lo largo del curso.

Se registraron aumentos de más del 300% entre las LOC/Horas producidas en las primeras semanas de clase a las obtenidas finalizando el curso utilizando técnicas de PSP. En la primera etapa de la aplicación de esta metodología, los estudiantes tienden a verla como una limitación, como algo tedioso y hasta como una pérdida de tiempo. Luego de asumirla como un marco de

trabajo para aprender y mejorar personalmente, comienzan a ver los resultados positivos.

Observaciones

El trabajo define una metodología de trabajo para el desarrollo de cada una de las actividades propuestas en el curso; sin embargo no define una estrategia para el proceso de formación.

5.3.10 Umea University

Umea University
Resumen El uso de PSP se utiliza para ejemplificar el desarrollo de software de forma disciplinada. Se aplica en dos cursos: el primero en un curso de Programación de segundo año y el segundo en un curso de Ingeniería de Software. Para el primer curo se desarrolló una versión extra lite de PSP para ofrecer los elementos esenciales de PSP. También se proporcionó una herramienta para apoyar la recolección de los datos y minimizar los errores. En el curso de Ingeniería de Software, se intentó un enfoque diferente para familiarizar a los estudiantes con el PSP. Este curso combina la teoría del curso con un proyecto de grupo y tareas. Se mostraron estudios de casos publicados ampliamente para mostrar a los estudiantes que se ha realizado correctamente y ha servido para el mejoramiento de sus prácticas.
Resultados Teniendo en cuenta la planeación del curso, se encuentra que dados los ejercicios planteados y el tiempo de entrega definidos para los mismos, no fue suficiente el tiempo para recoger datos históricos que permitieran realizar un análisis de las tendencias de los resultados. Considerando que el uso de la herramienta para la recolección de los datos de PSP era opcional, sólo 6 de los 78 estudiantes la utilizó a lo largo del curso. Para el resto de los estudiantes, el principal motivo de abandono de la herramienta de PSP era la sensación de que se estaba imponiendo un proceso excesivamente estricto en ellos y que era un trabajo extra en el cual no se percibía un beneficio. Los resultados de los exámenes realizados a los estudiantes mostraron que

este método de enseñanza les ayudó a entender los problemas que PSP puede ayudar a mejorar en cuanto al desarrollo de software.

Observaciones

Como estrategia de enseñanza dentro del curso, se han introducido conferencias magistrales para la impartición de los conceptos. No se reportó una estrategia de aprendizaje para la implementación de cada uno de los temas propuestos.

5.3.11 Utah University

Utah University
Resumen Este trabajo se realiza con estudiantes de primer año, se utiliza una versión adaptada de PSP para enseñar los conceptos fundamentales. Estos conceptos se imparten en los primeros dos cursos de programación básica. Como metodología de trabajo, se dan conferencias a los estudiantes sobre la materia. En el primer curso de programación los estudiantes trabajan en la parte de gestión de defectos y se les muestra la incidencia de los mismos dentro del proceso de desarrollo de software. Ellos analizan los defectos y se les enseñan técnicas para mejorar en estas prácticas. En el segundo curso, los estudiantes realizan actividades de estimación y seguimiento a su desempeño y al tiempo que invierten en la realización de sus programas; en este punto continúan analizando sus defectos. Los profesores proporcionan a los estudiantes informes con las estadísticas de la clase y se les pregunta sobre sus propios datos de PSP en los exámenes.
Resultados Se encontró que los estudiantes no encuentran la integración de este material de PSP con el tema de la programación. Debido a los resultados obtenidos, el curso de Ingeniería de Software se pasó para el segundo año y ahora se basa en los conceptos de PSP para su enseñanza. En cuanto a los resultados cuantitativos, se reportó que el 15% de los estudiantes gastó aproximadamente la mitad del tiempo en la fase de pruebas debido a la aplicación de PSP. Adicionalmente, los estudiantes que trabajaron por pares se divirtieron más, tenía mayor confianza en su trabajo, y cada uno alentó a otros para seguir las prácticas de PSP.

Observaciones

No se reporta una estrategia de aprendizaje para la impartición de los conceptos de PSP.

5.4 Análisis del marco de referencia

En esta sección se presenta un análisis de la información recopilada durante la revisión de los fundamentos teóricos y los antecedentes. El objetivo principal de este análisis es determinar las debilidades y las fortalezas de las propuestas estudiadas, y así identificar una serie de requerimientos potenciales que orienten el desarrollo de la propuesta de investigación. Afortunadamente, en la actualidad ya existen muy buenos estudios comparativos que han permitido identificar bondades y carencias en las distintas áreas de interés que trata esta trabajo. Para facilitar este proceso de análisis, se utilizan los estudios comparativos que están en capacidad de aportar información concluyente dentro de las líneas de esta investigación.

Dependiendo del entorno, existen muchas maneras de enseñar PSP. El material relacionado con PSP es bastante extenso, por lo que en ocasiones los instructores deben adaptar y personalizar lo definido en este proceso de desarrollo para satisfacer las necesidades de formación de estudiantes. Se han identificado tres factores principales que influyen en la enseñanza de PSP: el entorno de trabajo, el nivel de cobertura, y las herramientas de apoyo (Börstler, Carrington, & Hislop, 2002). Para el contexto de este trabajo estos factores también se harán extensivos a TSP. En cuanto al entorno de un curso de PSP, se hace referencia a que el mismo depende del público objetivo, el nivel del curso, y el contenido de la asignatura. El nivel de cobertura está asociado con las prácticas de PSP y TSP que se aplican y en relación con las herramientas; estas se encuentran relacionadas con los medios de soporte para el registro de cada una de las actividades que proponen tanto PSP como TSP.

Adicional a estos factores se realizan comentarios relacionados con la aplicación de estrategias de enseñanza y aprendizaje para cada uno de los trabajos analizados.

En la tabla 8 se muestra un resumen de las experiencias académicas reportadas, en las cuales se muestra la implementación de PSP y TSP en diversos contextos de educación a nivel mundial.

Tabla 8. Análisis de las experiencias en Universidades a nivel mundial

Universidad	Entorno de trabajo	Nivel de cobertura	Herramientas de soporte	Estrategia de enseñanza – aprendizaje
Lund University	Estudiantes de Pregrado Estudiantes de Posgrado Personas de la industria	Full PSP[1]	Hojas de Cálculo	No se hace referencia
Zagreb University	Estudiantes de Pregrado	PSP-Lite[2]	PSP-Tool se desarrollo localmente	No se hace referencia
Purdue University	Estudiantes de Pregrado	PSP-Lite	Hojas de Cálculo	No se hace referencia
Loyola University	Estudiantes de Posgrado	Full TSP	Formularios de TSP	No se hace referencia
Embry-Riddle Aeronautical University	Estudiantes de Pregrado	Full PSP	Formularios de PSP	Estrategia de enseñanza
Birla Institute of Technology	Estudiantes de Pregrado	Full PSP	Formularios de TSP	No se hace referencia
Universidad Carlos III	Estudiantes de Pregrado	PSP-Lite	PSP Student Workbook	Estrategia de enseñanza
Universidad Politécnica de Madrid	Estudiantes de Pregrado	Full TSP	Formularios de TSP	No se hace referencia
Universidad Nacional de Matanza	Estudiantes de Pregrado	PSP-Lite	Formularios de PSP	No se hace referencia

[1]Hace referencia a la aplicación de todo el cuerpo de conocimiento de PSP.
[2]Hace referencia a una versión simplificada o adaptada de PSP.

Umea University	Estudiantes de Pregrado	PSP-Lite	Se desarrolló localmente	No se hace referencia
Utah University	Estudiantes de Pregrado Estudiantes de Posgrado	PSP-Lite Full PSP	Se desarrolló localmente	No se hace referencia

Aunque una comparación visual entre la condición deseada y las demás propuestas hacen creer que existen grandes vacíos en las propuestas existentes en la literatura, esto no significa que las propuestas estén mal desarrolladas. Cada propuesta existe porque resuelve un problema en particular y hasta ahora no se observa una que considere todos estos atributos deseados. Cada propuesta nueva que deba ser estudiada dentro del proceso de integración tendrá que ser sometida a todos estos procesos de clasificación, para entender la relevancia de su aporte dentro del contexto particular.

Con base en el análisis de referencia se puede establecer que cada una de las experiencias reportadas tiene una particularidad dado el contexto y los intereses de formación para sus estudiantes. Como elemento diferenciador de esta propuesta, se encuentra que la implementación de una estrategia de aprendizaje es necesaria para la incorporación de conceptos de calidad durante el proceso de formación de los estudiantes de pregrado.

6 PLANTEAMIENTO METODOLÓGICO

El análisis del marco de referencia ha permitido deducir que la apropiación de las prácticas de PSP y TSP durante el proceso de formación básica, permite a los estudiantes comprender el concepto de proceso, lo que a la vez permite la adquisición de hábitos relacionados con el registro de información y posteriores mediciones. Así mismo, la recolección de datos individuales y de equipo permite tener una línea de referencia para la planificación y el desarrollo de proyectos futuros.

Del análisis de las experiencias de la implementación de PSP y TSP en la academia se han identificado diferentes prácticas, las cuales se convierten en objeto de análisis, con el fin de aportar los elementos que permitan la identificación de las prácticas fundamentales de PSP y TSP.

Teniendo en cuenta que se desea realizar una diagnóstico de las prácticas que actualmente realizan los estudiantes es sus proyectos de Programación, es necesario identificarlas. Para esto se considera fundamental el conocimiento de las métricas de PSP y TSP, de forma tal que se pueda definir una serie de categorías e indicadores que servirán como soporte al diseño del instrumento que se va a aplicar a los estudiantes de Ingeniería de Sistemas y Computación de la Universidad del Quindío.

6.1 Prácticas y métricas de PSP y TSP

El uso de métricas es una característica importante de todas las disciplinas de ingeniería. En un marco de trabajo de ingeniería, las métricas se refieren a estándares de las medidas usados para cuantificar aspectos específicos de un proceso, de un producto o de un proyecto de ingeniería (Pressman, 2002). La recolección de medidas es el primer paso para conocer cómo se controla y mejora el proceso de desarrollo de software (Moser, Janes, Russo, & Sillitti, 2005).

Las métricas que se extraen del proceso, como por ejemplo, medición de tiempo y esfuerzo, tienen usos privados y públicos (Grady, 1992). Es por esto que algunas de estas métricas recopiladas deberían ser privadas para el desarrollador y servir como indicador sólo para él. En esta filosofía de datos de procesos privados centra su enfoque el PSP, reconociendo por tanto que la mejora del proceso de desarrollo de software puede y debe comenzar en el nivel individual (Pressman, 2002).
Con base en las experiencias analizadas en el marco de referencia, se describe a continuación una serie de métricas basadas en (Humphrey W. , The Personal Software Process, 2000) y (Pérez, 2009), que servirán para la realización de diferentes prácticas por parte de los estudiantes. Las

siguientes mediciones sirven como referencia para la estructuración de la estrategia de enseñanza definida.

Tabla 9. Métricas de PSP.(Humphrey W. , Introduction to the Team Software Process, 2000), (Pérez, 2009).

Métrica	Descripción
Esfuerzo de precisión de la estimación	100 * (Esfuerzo Actual - Esfuerzo Estimado) / Esfuerzo Estimado
Tamaño de precisión de la estimación	100 * (Tamaño Actual - Tamaño Estimado) / Tamaño Estimado
Total de defectos	Defectos encontrados durante las fases del proceso
Densidad de Defectos	Defectos encontrados / KLOC.
Productividad	LOC / Horas, Tamaño de producto desarrollado por hora
Costos	Costos / LOC
Tamaño del programa	Número de líneas de código (LOC) del programa
Tiempo de desarrollo	Tiempo en minutos del proceso y de sus fases.
Intensidad de error	Número total de defectos / Hora
Número de defectos en prueba	Defectos de prueba / KLOC
% de tiempo en fase	% del tiempo total del proyecto en cada fase
Defectos en compilación y pruebas	Número de defectos encontrados en la fase de compilación y pruebas.
Tiempo de interrupción	Tiempo de interrupción en cada una de las fases del proyecto
Defectos eliminados	Número de defectos eliminados por fases del proceso
Páginas de documentación	Cantidad de páginas de documentación
Tiempo en cada fase	% de tiempo en cada fase del proceso.
% Defectos inyectados y removidos por fase	% total de los defectos que son inyectados y removidos en cada fase.
% Reutilización	Tasa de reutilización de código ya desarrollado
% Nuevo reutilizable	Tasa de incorporación de nuevo código reutilizable.
Rendimiento de Fase (Yield Phase),	Yield (de una fase) = 100 * (defectos encontrados) / (defectos encontrados + no encontrados) Defectos no encontrados (defectos inyectados antes o durante la fase, pero detectados)

Rendimiento del Proceso (Yield)	% de defectos insertados y removidos antes de la primera compilación. Yield (overall) = 100 - (defectos removidos antes de compilación) / Defectos Insertados antes de compilación.
Defectos de prueba por KLOC	Defectos de prueba / KLOC = 1000 - Defectos removidos en pruebas/total de líneas adicionadas y modificadas
Defectos totales por KLOC	Total de defectos inyectados durante el proceso. Defectos totales / KLOC = 1000 - Total Defectos removidos /total de líneas adicionadas y modificadas
Efectividad de remoción de defectos	Número de defectos removidos por hora en revisión de diseño, revisión de código, compilación y test. Eficiencia de remoción de defectos = 60 - defectos removidos en fase / tiempo en fase (minutos)
Costos de fallas – Failure COQ	Failure COQ = 100*(tpo comp + tpo test) / (tpo total). El porcentaje de tiempo total gastado en el desarrollo de compilación y prueba
Tasa de revisión	Líneas de código revisadas por hora LOC / Hora
Tasa de remoción de defectos	% eliminación de defectos en una fase revisión del diseño, revisión de código, compilación y pruebas). Defectos / hora

Con datos de tamaño, tiempo y defectos, existen muchas formas de medir, evaluar, y manejar la calidad de un programa. PSP y TSP proveen una serie de mediciones de calidad que ayudan a los desarrolladores a examinar la calidad de sus programas desde varias perspectivas. Considerando las anteriores medidas, se puede afirmar que existen diversas formas para analizar, medir y gestionar la calidad tanto del producto como del proceso. La existencia de estas medidas para PSP y TSP se encuentran asociadas como indicadores confiables de calidad.

A continuación se definen las categorías y los indicadores para el desarrollo del diagnóstico definido en la propuesta de este trabajo.

6.2 Categorías e indicadores

Como marco inicial para determinar los posibles indicadores de las prácticas que usan los estudiantes en su proceso de desarrollo personal, es pertinente considerar aquellas implicaciones a tener en cuenta como ejes de una línea de base, que soporten de manera clara la razón del uso

de estas prácticas en actividades de estimación de tamaño, esfuerzo, defectos y tiempo, de trabajo en equipo, documentación y productividad.

Se pretende identificar y analizar las mejores prácticas de PSP y TSP, para seleccionar las métricas en la estructuración de la estrategia de aprendizaje. Se considera fundamental para el desarrollo de esta estrategia de formación, el conocimiento de una forma más integral de los estudiantes a los cuales está dirigida la misma. Esta situación, desde el punto de vista de la investigación, implica que al hablar de integral, se asuma el conocer tanto técnicas individuales como de equipo para el desarrollo de software. Se pretende identificar las prácticas relacionadas con el manejo del tiempo, la gestión de los defectos, las estimaciones de tamaño, la planeación y el trabajo en equipo.

Considerando las medidas definidas en la sección anterior, se proponen 6 categorías para agrupar una serie de indicadores que muestren la presencia de las variables anteriormente mencionadas, y que son las que soportan el desarrollo del proyecto. Estas categorías, son:

- Administración y Gestión del tiempo
- Manejo y Gestión de defectos
- Manejo del tamaño del producto
- Trabajo individual
- Planeación del proyecto y del proceso de desarrollo
- Organización y trabajo en equipo

A continuación se hace una descripción de cada una de las categorías y su aporte al diagnóstico del estado actual de las prácticas de desarrollo individual y de los equipos.

- **Administración del tiempo**

Con esta categoría se asume que los equipos de trabajo y las personas deben gestionar correctamente su tiempo. Es necesario tener una valoración de esta categoría para observar las prácticas que actualmente realizan en este sentido.

Como primera aproximación a la valoración de esta categoría, se definen los siguientes indicadores de tipo cualitativo. Para nuestro contexto, la palabra proyecto se usará para referirse también a

tarea, práctica, trabajo o actividad. En la tabla 10 se muestran los indicadores de tipo cualitativo que se definen para esta categoría.

Tabla 10. Categorías e indicadores de administración del tiempo

Categoría	Indicadores
Administración del tiempo	Se registra el tiempo invertido en la realización de un proyecto.
	Se identifican los factores de interrupción de la actividad cuando realiza un proyecto.
	Se registra el tiempo de las interrupciones cuanto se está realizando un proyecto.
	Se estima el tiempo a invertir cuando se va a realizar un proyecto.
	Se distribuye el tiempo de un nuevo proyecto basado en la distribución del tiempo de proyectos anteriores.
	Se analiza la cantidad de tiempo que se pasa en cada fase del proceso.
	Se excede generalmente el tiempo establecido para realizar un proyecto.

- **Manejo y gestión de los defectos**

Se define esta categoría teniendo en cuenta que es necesaria la correcta identificación y gestión de los defectos que se introducen durante las fases del proceso de desarrollo de software. Los siguientes son los indicadores de tipo cualitativo. En la tabla 11 se muestran las categorías y los indicadores asociadas a la gestión de los defectos.

Tabla 11. Categorías e indicadores de gestión de defectos.

Categoría	Indicadores
Manejo y gestión de los defectos	Se registra y analiza los defectos al realizar un proyecto.
	Se comprenden los defectos introducidos al realizar un proyecto.
	Se cuenta el tiempo que tarda en encontrar y corregir cada defecto al realizar un proyecto.
	Se aplica un método para encontrar y corregir defectos cuando se realiza un proyecto.
	Se cuenta la cantidad de defectos cuando realiza un proyecto.

- **Manejo del tamaño del producto**

Esta categoría permite identificar las estimaciones para estimar los esfuerzos individuales y de equipo en la realización de un proyecto. Para PSP y TSP la unidad de medida del tamaño del producto son las líneas de código (LOC). La tabla 12 muestra los indicadores de tipo cualitativo que se definen para esta categoría.

Tabla 12. Categoría e indicadores del tamaño del producto

Categoría	Indicadores
Manejo del tamaño del producto	Se estima el número de líneas de código para la realización de un proyecto.
	Se realiza un conteo de las líneas de código modificadas cuando está reparando un defecto.
	Se determina las líneas de código por hora producidas en los proyectos.
	Se planea las líneas de código por hora a escribirse en un proyecto.

- **Trabajo individual**

Este indicador permite conocer las acciones y prácticas de calidad que usan las personas de acuerdo con su rol dentro de un equipo de desarrollo de software. La tabla 13 contiene los indicadores de tipo cualitativo que se definen para esta categoría.

Tabla 13. Categoría e indicadores de trabajo individual.

Categoría	Indicadores
Trabajo individual	Se identifican las acciones que realizan otros para que sus proyectos funcionen mejor.
	Se aplican prácticas relacionadas con estándares de codificación.
	Se documentan los problemas al momento de realizar un proyecto.
	Se usan modelos de diseño cuando va a realizar un proyecto.

- **Planeación del proyecto y del proceso de desarrollo**

Este indicador pretende identificar la forma como se planifica un proyecto de acuerdo con unos recursos que se encuentran disponibles. La tabla 14 muestra los indicadores de tipo cualitativo que se definen para esta categoría.

Tabla 14. Categoría e indicadores para planeación del proyecto.

Categoría	Indicadores
Planeación del proyecto y del proceso de desarrollo	Se planifica la forma como se va a realizar un proyecto.
	Se sigue un cronograma y se hace seguimiento al proyecto.
	Se centra en aspectos de programación, sin considerar un proceso.
	Se planifican las pruebas para verificar la funcionalidad del producto
	Se usa un proceso de desarrollo de software cuando va a realizar un proyecto.
	Se reúne información de los proyectos para la planeación de proyectos futuros.

- **Organización y trabajo en equipo**

Con esta categoría se pretende definir una serie de indicadores alrededor del trabajo en equipo, mediante la asignación de roles dentro del proceso de desarrollo de software. La tabla 15 contiene los indicadores de tipo cualitativo que se definen para esta categoría.

Tabla 15. Categoría e indicadores de trabajo en equipo

Categoría	Indicadores
Organización y trabajo en equipo	Se toman decisiones concertadas en equipo cuando se realiza un proyecto.
	Se registran los defectos encontrados.
	Se identifica en los integrantes del grupo los conocimientos, aptitudes y habilidades, para la conformación del equipo.
	Se definen los roles y responsabilidades de cada miembro del equipo para realizar un proyecto.
	Se planea la realización de un proyecto.
	Se analiza en grupo los defectos encontrados.

Una vez definidos los indicadores de tipo cualitativo, se procede a realizar un diagnóstico de las prácticas que aplican los estudiantes en su proceso de desarrollo de software.

6.3 Diagnóstico de los indicadores

Este capítulo se centra en validar el conjunto de indicadores propuestos para valorar las prácticas de desarrollo de software, tanto individuales como de equipo. Esta validación parte de la

construcción de un instrumento que recoge información de los estudiantes del programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío.

A partir de la información recolectada, se analiza la validez de los indicadores propuestos en cada categoría, de forma que permitan valorar las prácticas de desarrollo de software individuales y en equipo. Los pasos que se siguieron para el análisis de los indicadores propuestos consistió, en:

- Análisis y diseño de la encuesta para la obtención de la información.
- Selección de la población objetivo.
- Aplicación del instrumento a la población objetivo.
- Análisis de los resultados obtenidos, producto de la aplicación del instrumento.

6.3.1 Contexto para la aplicación del caso de estudio

El programa de Ingeniería de Sistemas y Computación fue creado el 12 de agosto de 1996, según Acuerdo del Consejo Superior 064, recibiendo a los primeros estudiantes en el primer semestre académico de 1997.

El Programa ha tenido una gran acogida por parte de la comunidad. El número de nuevos estudiantes que ingresan semestralmente es en promedio 70 en la franja diurna y 75 en la franja horaria nocturna. En el primer semestre de 2012 se encuentran matriculados en el Programa 340 estudiantes en su franja diurna y 326 en su franja nocturna (Universidad del Quindío, 2012).

La primera promoción de los estudiantes del Programa egresó el 16 de abril de 2002. A partir de esta fecha y hasta abril de 2012 se han graduado aproximadamente 460 estudiantes, para un total de 21 promociones. Los egresados del Programa se han vinculado a empresas nacionales e internacionales o han creado sus propias empresas, laborando en cargos acordes con la formación recibida.

La información básica del Programa de Ingeniería Sistemas se muestra en la tabla 16.

Tabla 16. Información Programa de Ingeniería de Sistemas y Computación. (Universidad del Quindío, 2010)

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 1080 de abril de 2005 del MEN
Año de iniciación de actividades académicas	1997

El Programa de Ingeniería de Sistemas y Computación ha definido en su Proyecto Educativo las competencias y perfiles que debe tener el profesional egresado. En lo referente al perfil laboral, se destaca la capacidad para analizar y resolver problemas de forma individual o interdisciplinaria, proponiendo soluciones que generen impacto social.

Por su parte, el perfil ocupacional habilita a los egresados para ejercer su profesión en los siguientes cargos: Jefe de proyectos de sistemas, Directores - Gerentes de departamentos de sistemas, Desarrollador de software, empresario en el sector del desarrollo de software, soporte a usuarios, Analista de sistemas, administrador y docente-investigador (Universidad del Quindío, 2007).

La estructura curricular del Programa, ésta se basa en la Política Académico Curricular, Acuerdo del Consejo Superior 018 del 18 de diciembre de 2003. De acuerdo con esta política, las actividades académicas se clasifican en: Académicas Básicas, Académicas Profesionales, Electivas Complementarias, Electivas Profesionales, Obligatorias de Ley y Obligatorias Institucionales, Básicas de profundización y Profesionales de profundización. La figura 6 muestra la proporción de créditos por áreas de estudio.

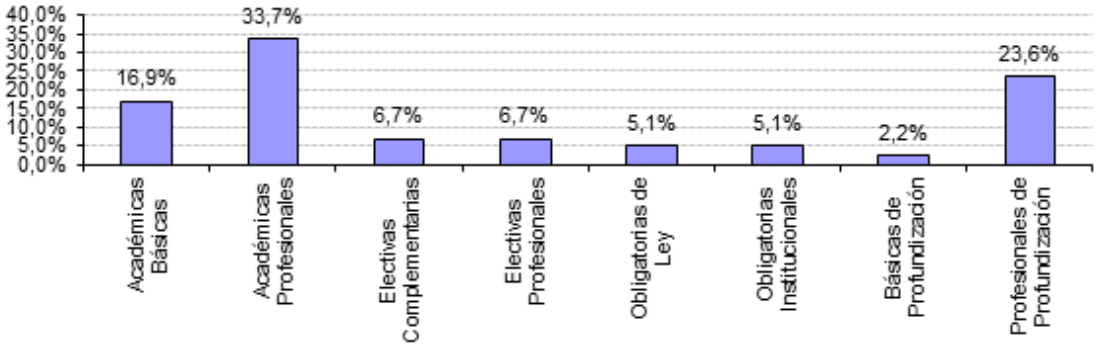


Figura 6. Proporción de créditos académicos por áreas. (Universidad del Quindío, 2007)

Las asignaturas del Área de Programación y Algoritmia son las siguientes:

- Paradigma Orientado a Objetos
- Fundamentos de Algoritmia
- Lenguajes de Programación
- Estructuras de Datos
- Análisis de Algoritmos I

El diseño de la estrategia de enseñanza del proyecto, se realizará para las asignaturas anteriormente mencionadas.

6.3.2 Recolección de Información

Es necesario realizar un diagnóstico sobre el uso de buenas prácticas de desarrollo de software por parte de los estudiantes del programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío, relacionados con el área de Programación y Algoritmia, para definir una línea base de esta investigación.

Con base en las categorías e indicadores anteriormente, se diseñó el instrumento para la recolección de información en la población seleccionada. El instrumento se enfocó en cada una de las categorías definidas. Para cada una de estas categorías se cuenta con un conjunto de indicadores y sus respectivas preguntas, las cuales hacen parte de este instrumento. El instrumento de encuesta consta de 3 secciones, las cuales están estructuradas de una forma sencilla para su respuesta.

La primera sección contiene la explicación de cómo responder las preguntas y los campos para diligenciar la información de la persona que responde la encuesta. La segunda sección contiene las 32 preguntas de las 6 categorías definidas y la última sección contiene un espacio abierto para ampliar sobre los temas consultados o para observaciones personales.

6.3.3 Aplicación del instrumento

Los resultados obtenidos provienen de la información suministrada a través de una encuesta aplicada a los estudiantes de Ingeniería de Sistemas y Computación de la Universidad del Quindío sobre las prácticas que se realizan en su proceso de desarrollo de software personal. Estos resultados permiten identificar el nivel de aplicación de algunos criterios de calidad durante la realización de los proyectos de desarrollo de software.

La población total de estudiantes del programa de Ingeniería de Sistemas y Computación es de 342 estudiantes. Para tomar la muestra poblacional se consideró una muestra por estratos, determinada por la cantidad de estudiantes por semestre. El error de muestreo definido es del 6%, por lo tanto la muestra analizada es de 151 estudiantes. En la tabla 18 se muestra la proporción por estratos de los estudiantes para el análisis de la muestra.

Tabla 18 – Población total y muestra poblacional.

Semestre	Número de estudiantes total	Proporción	Número de estudiantes para la muestra
I	102	0,30	45
II	71	0,21	31
III	30	0,09	13
IV	26	0,08	11
V	10	0,03	7
VI	32	0,09	14
VII	16	0,05	9
VIII	21	0,06	10
IX y X	25	0,07	11

6.4 Análisis de la aplicación de instrumentos

Los resultados obtenidos se analizan desde las evaluaciones realizadas a los estudiantes sobre las prácticas que se realizan en su proceso de desarrollo de software personal. Estos resultados permiten identificar el nivel de aplicación de algunos criterios de calidad durante la realización de los proyectos de desarrollo de software.

Considerando que se ha tomado la muestra más grande posible de estudiantes del programa académico, se debe tener en cuenta el nivel de conocimiento en áreas de Programación y de Ingeniería de Software. El total de estudiantes que respondieron la encuesta fue de 151. Los estudiantes de semestres iniciales han visto cursos básicos de Programación, Estructura de datos y Análisis de algoritmos. Los estudiantes de semestres avanzados ya han visto cursos de Ingeniería de Software y de Gestión de la información, y por lo tanto han trabajado con conceptos de procesos de software y de calidad.

A continuación se presentan los resultados y el análisis de la aplicación del instrumento, manteniendo el esquema de categorías.

La figura 7 muestra los resultados de las preguntas asociadas con las prácticas de administración del tiempo. Se observa que un 82,8% de los estudiantes no registra el tiempo cuando está realizando un proyecto de Programación (pregunta 1); en cuanto a la estimación del tiempo, se observa que un 66,2% estima el tiempo que va a invertir cuando va a realizar un proyecto de Programación (pregunta 2), y sólo el 21,9% de los estudiantes analizan la cantidad de tiempo que pasa en cada una de las fases del proceso de desarrollo de software (pregunta 3). Con relación a la administración del tiempo, es fundamental propiciar escenarios en los cursos de Programación, en los cuales los estudiantes realicen registro del tiempo en cada una de las fases del proceso de desarrollo.

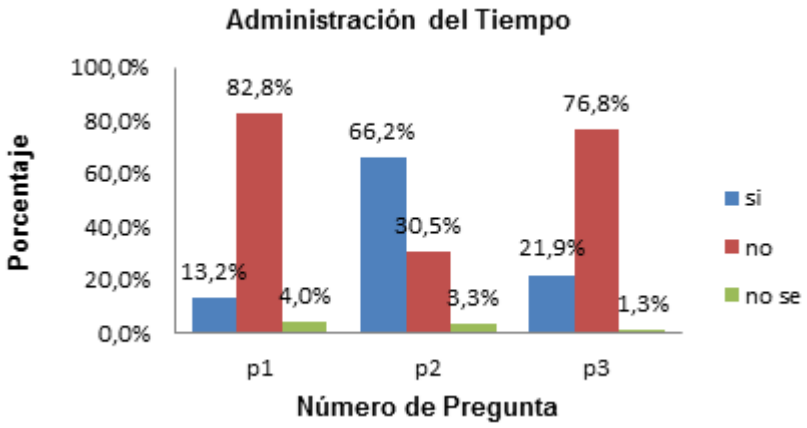


Figura 7 – Evaluación de la práctica de administración del tiempo.

En cuanto a las prácticas relacionadas con el manejo de defectos, los cuales son aspectos que reducen la capacidad de los trabajos o proyectos para cumplir completa y efectivamente las necesidades de los usuarios, en la Figura 8, se observa que sólo el 31,8% de los estudiantes registra y analiza los defectos que ha introducido al realizar un proyecto de Programación (pregunta 4). Con relación al registro del tiempo que los estudiantes tardan en encontrar y corregir cada defecto, se observa que el 89,4% de estudiantes no realiza esta actividad (pregunta 5). En cuanto a la práctica de contabilizar la cantidad de defectos que encuentra al realiza un proyecto, sólo el 23,2% realiza esta actividad (pregunta 6).

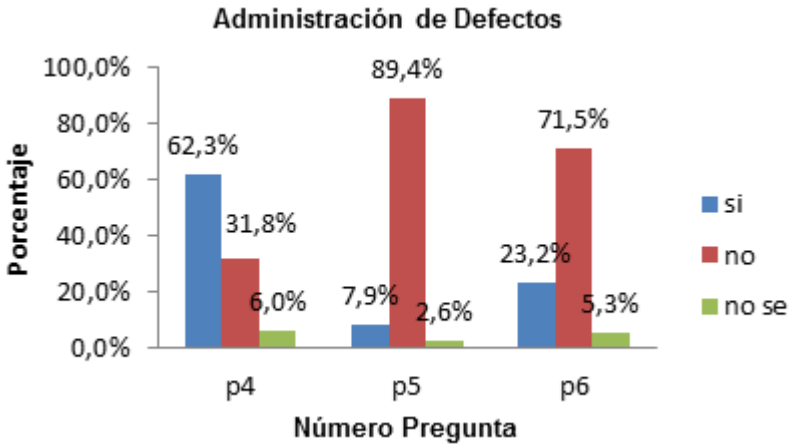


Figura 8 – Evaluación de la práctica de registro de defectos.

De lo anterior se puede decir que los estudiantes reconocen el impacto que puede generar los defectos en los productos y proyectos de software; sin embargo no contabilizan el tiempo que tardan en corregir estos defectos.

En cuanto a la estimación del tamaño del producto en términos de líneas de código que se generan al codificar un programa, en la figura 9 se observa que el 84,1% no realiza estimaciones del número de líneas de código necesarias para la realización de un proyecto (pregunta 7). Así mismo, para el conteo de las líneas de código que modifica en la reparación de un defecto, sólo el 7,9% de los estudiantes realizan esta práctica (pregunta 8). Finalmente, el 95,4% de los estudiantes no determinan cuántas líneas de código por hora producen cuando realizan un trabajo (pregunta 9).

La no aplicación de las prácticas de gestión del tamaño del producto se debe a que para realizar estimaciones es necesario tener una línea base con información de programas que han sido

anteriormente construidos. Para esta métrica se deben establecer actividades que permitan a los estudiantes trabajar sobre proyectos ya finalizados.

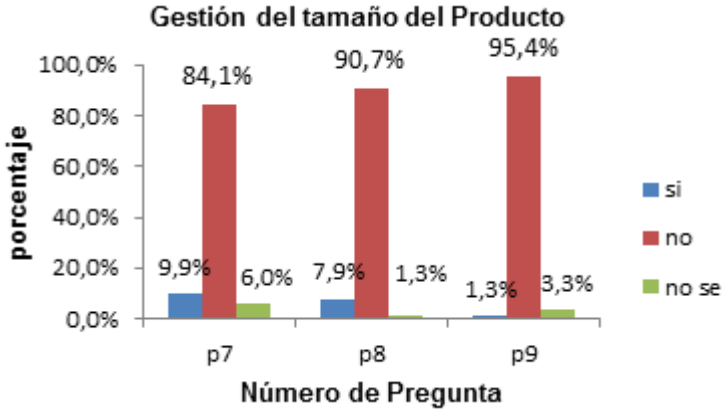


Figura 9 – Evaluación de la práctica de estimación de tamaño

En cuanto al trabajo individual para realizar trabajos o proyectos, en la figura 10 se observa que un alto porcentaje de los estudiantes identifica las acciones que realizan los otros para que sus tareas o proyectos funcionen mejor (pregunta 10). En cuanto a la aplicación de prácticas relacionadas con estándares de codificación (pregunta 11), se observa que casi la mitad de los estudiantes las aplican y el 39,1% de los estudiantes no aplican estos estándares; como aspecto a tener en cuenta, casi el 16% de los estudiantes afirman no saber, esto necesariamente implica la revisión de la pregunta pues a priori se puede decir que muchos estudiantes no la entendieron. En relación con el uso de diagramas o modelos cuando se va a realizar un proyecto, se encuentra que el 73,5% de los estudiantes si los usan, y el porcentaje restante no los aplican o no los conocen.

De lo anterior se puede decir que un alto porcentaje de los estudiantes analizan la forma como otros pares realizan sus trabajos para realizar los suyos propios. Otro aspecto a tener en cuenta está relacionado con que un alto porcentaje de los estudiantes no aplican o no conocen estándares de codificación y con ello su aplicación en los proyectos de Programación.

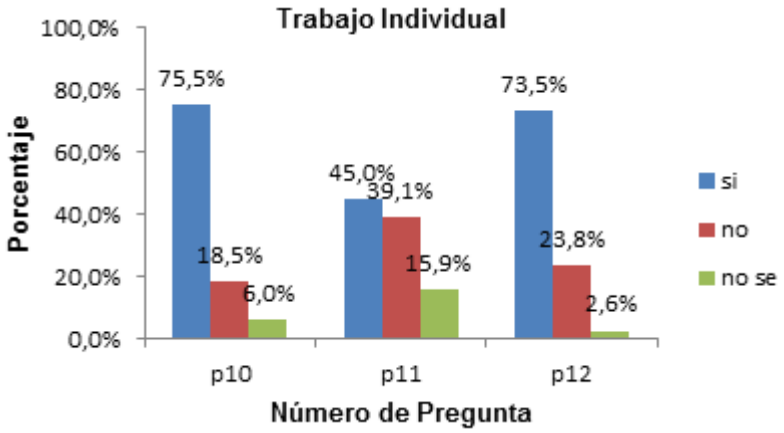


Figura 10 – Evaluación de la práctica de estimación de tamaño.

En cuanto planeación del proyecto y al proceso de desarrollo, en la figura 11 se observa que un alto porcentaje el 70,5% de los estudiantes planifica la forma como va a realizar un proyecto (pregunta 13). En cuanto a la pregunta si el estudiante se centra en aspectos de Programación, sin tener en cuenta un proceso, se encuentra que el 57,6% de los estudiantes perciben la necesidad de un proceso antes de llegar a la fase de planeación. En cuanto al conocimiento y aplicación de las fases de proceso de desarrollo de software cuando se va a realizar un proyecto, sólo el 20,5% de los estudiantes respondieron afirmativamente.

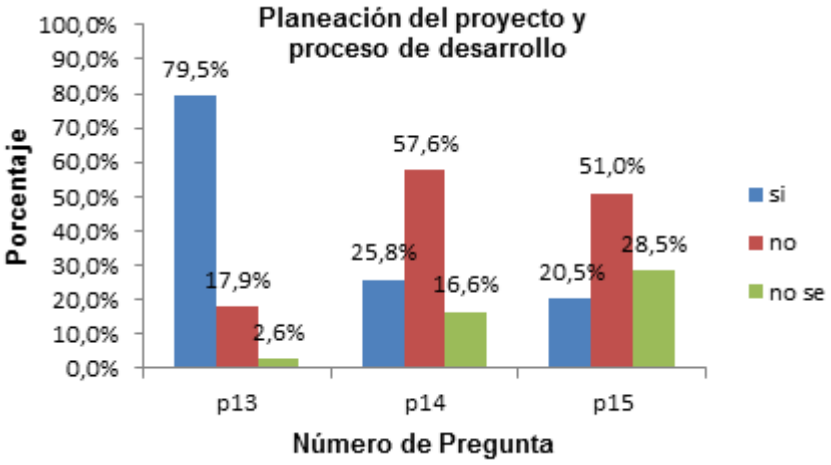


Figura 11 – Evaluación de la práctica de estimación de tamaño.

Como aspectos a tener en cuenta, se observa que es necesario trabajar desde el comienzo en aspectos básicos de procesos de desarrollo de software, de forma tal que el estudiante reconozca

su importancia y valor para aplicarlos en sus proyectos o tareas de Programación. Como aspecto positivo, se resalta que los estudiantes estén planeando sus proyectos de Programación.

Con relación al trabajo en equipo, se observa en la figura 12, que un alto porcentaje de los estudiantes cuando trabajan en equipo toman decisiones concertadas cuando se está realizando un trabajo o proyecto (pregunta 16). Un 78,8% de los estudiantes al momento de la conformación de los equipos identifica en cada uno de sus integrantes los conocimientos, aptitudes y habilidades, para un buen desempeño (pregunta 17). Un 68,3 % de los estudiantes no conoce o no saben sobre la asignación de roles y responsabilidades de cada miembro del equipo para realizar un trabajo o proyecto (pregunta 18).

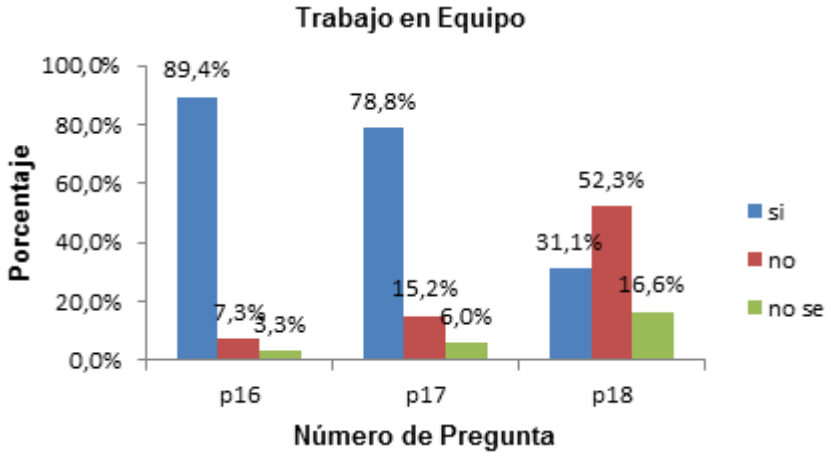


Figura 12 – Evaluación de la práctica de estimación de tamaño.

Teniendo en cuenta el análisis de los indicadores definidos, en el siguiente capítulo se hará el diseño de la estrategia de aprendizaje que va a tener en cuenta lo considerado en los resultados obtenidos anteriormente, ya sea desde la perspectiva de mejorar algunos aspectos o reforzar aquellos que los ameriten, desde el punto de vista de la formación de los estudiantes.

6.5 Propuesta Metodológica

La investigación es de tipo experimental, además exploratoria, por cuanto se inicia la implementación de estrategias de aprendizaje que permitan la mejora de las dificultades encontradas en los estudiantes al gestionar su proceso personal y en equipo al momento de desarrollar software, y reconocer hasta dónde la estrategia permite comprender con mayor claridad las temáticas tratadas.

La población objeto de estudio está definida por dos grupos del primer curso de Programación de computadores del plan de estudios de Ingeniería de Sistemas y Computación de la Universidad del Quindío.

Para esta investigación se debe tener en cuenta que los grupos que ingresan tengan condiciones homogéneas o donde todos los estudiantes tengan las mismas características, en términos de los presupuestos teóricos con que cuentan al inicio de la ejecución del proyecto de investigación, para que permitan mayor eficiencia en las pruebas y actividades que se realicen para alcanzar los objetivos propuestos.

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

- Definición del grupo experimental y el grupo control.
- Aplicar al grupo control y al grupo experimental el pre-test para conocer los conceptos previos que tienen los estudiantes sobre el tema objeto de estudio.
- Desarrollar en el grupo control los temas de la asignatura, siguiendo el modelo tradicional, y desarrollar en el grupo experimental las estrategias de aprendizaje definidas.
- Hacer seguimiento continuo de los avances y logros alcanzados por cada uno de los implicados en el proyecto (profesores, estudiantes).
- Comparar e interpretar los resultados obtenidos en la prueba diagnóstica y los resultados finales de la implementación, mediante un pos-test que involucre los temas objeto de estudio.
- Realizar un análisis de los resultados.

A continuación se muestra en la figura 13 un esquema de la metodología para la realización de la investigación.

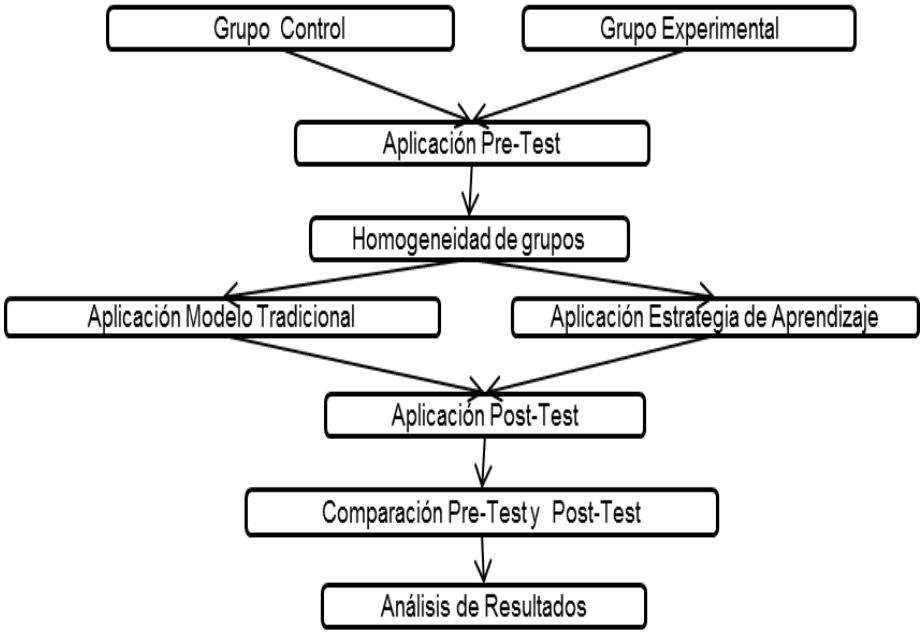


Figura 13. Metodología de desarrollo del proyecto. (Cardona & Rincón, 2012)

6.5.1 Diagnóstico población objeto

La propuesta de formación está enmarcada en el área de Programación y Algoritmia; los estudiantes de primer semestre no tienen experiencia en Programación ni en trabajo en equipo. Los estudiantes de segundo semestre en adelante ya cuentan con una serie de conocimientos, habilidades y actitudes relacionadas con la Programación, el desarrollo de proyectos y el trabajo en equipo.

Para el diagnóstico de la población objeto se seleccionó el grupo experimental y el grupo control en el programa de Ingeniería de Sistemas y Computación; se diseñó y se aplicó al grupo control y al grupo experimental el pretest, el cual consta de un cuestionario de 10 preguntas con opción (Sí-No), para conocer los conceptos previos que tenían los estudiantes en relación con prácticas individuales y en equipos de desarrollo de software. Se calificaron los cuestionarios, se procesaron los datos y se analizaron los resultados con el fin de establecer condiciones de homogeneidad en los grupos objeto de investigación.

La cantidad de estudiantes de los grupos control y experimental que respondieron la encuesta fueron 31 y 35 respectivamente.

6.5.2 Test de Conocimientos previos

La siguiente es una prueba diagnóstica para los estudiantes de primer semestre del programa de Ingeniería de Sistemas y Computación que presentaron la encuesta y con la cual se pretende evaluar el nivel de desarrollo del pensamiento de los estudiantes en el campo conceptual, las prácticas individuales y en equipo para el desarrollo de software. Las preguntas se estructuraron basadas en las categorías definidas con anterioridad y las cuales son:

- Administración y Gestión del tiempo
- Manejo y Gestión de defectos
- Manejo del tamaño del producto
- Trabajo individual
- Planeación del proyecto y del proceso de desarrollo
- Organización y trabajo en equipo

Las preguntas 1 y 2 tienen la intencionalidad de diagnosticar las prácticas relacionadas con la administración del tiempo. El dominio de estos conceptos es de gran importancia para que los estudiantes identifiquen la importancia de controlar el tiempo cuando estén desarrollando software.

Las preguntas 3 y 4 pretenden indagar por el nivel de aplicación de prácticas asociadas con la gestión de defectos al momento de realizar un trabajo de programación. Este concepto aporta en el sentido que los estudiantes deben identificar dentro de sus trabajos de Programación cuáles practicas pueden implementar para mejorar la calidad del programa en término de los defectos introducidos.

La pregunta 5 indaga los conceptos previos sobre la aplicación de técnicas de codificación. El concepto se considera importante, dada la necesidad de implementar buenas prácticas de codificación, de acuerdo con los estándares establecidos.

La pregunta 6 tiene como intencionalidad identificar la buena práctica de aplicación de estándares de codificación definidos para la construcción de los programas.

Las preguntas 7 y 8 indagan los conceptos relacionados con la planificación individual y en equipo para la elaboración de los trabajos de Programación.

La pregunta 9 consulta sobre lo relacionado con la asignación de roles y de trabajo en equipo, al momento de realizar trabajos de Programación.

6.5.3 Análisis de PreTest

Se realizó una encuesta a estudiantes de primer semestre de Ingeniería de Sistemas y computación (Ver Anexo 2), teniendo en cuenta las necesidades propias de la investigación, entre las cuales se consideró que los estudiantes no hubieran trabajado con temas de Programación, tales como: métodos, ciclos, arreglos y métodos de ordenamiento; y que tuvieran los mismos conocimientos previos de Lenguaje de programación.

Se escogió, con la ayuda del profesor titular, el grupo control y el grupo experimental; inicialmente se aplicó el pretest a los estudiantes que asistieron ese día a clase.

- Durante la ejecución de la estrategia se llevó un control de asistencia.
- Se aplicó el pretest al 100% de los estudiantes que asistieron ese día a clase.
- Para el análisis final se descartaron los estudiantes que no habían presentado el pretest, y aquellos estudiantes que no asistieron al 100% de las clases de la asignatura.

Con la prueba del pretest aplicada a ambos grupos, se analizó el nivel de homogeneidad en cada una de las preguntas del instrumento tanto para el grupo experimental como el de control. Para verificar la homogeneidad de los grupos se corrió un Análisis de la Varianza ANOVA (*ANalysis Of VAriance*) cuya variable de respuesta es la calificación de la pregunta y el factor es el grupo con los niveles de control y experimental. Para cada una de las preguntas se aplicaron los supuestos de aleatoriedad, homogeneidad de varianzas y la distribución normal de residuos. En caso que los tres supuestos no se cumplan, entonces se aplica la prueba No paramétrica de Kruskay y Wallis. A continuación se analiza y se muestran los resultados para cada una de las preguntas.

Para la pregunta 1 en el análisis de varianza el p_valor indica que no hay diferencias estadísticamente significativas entre los grupos experimental y control, pero al verificar los supuestos no se cumple que los residuos se ajustan a una distribución normal, según la prueba de

Shapiro-Wilk el p_valor<0,0001; debido a lo anterior se acude a la prueba No paramétrica de Kruskay y Wallis, la cual permite afirmar que no existe diferencia significativa entre los grupos. La figura 14 muestra los resultados asociados al control de tiempos para ambos grupos.

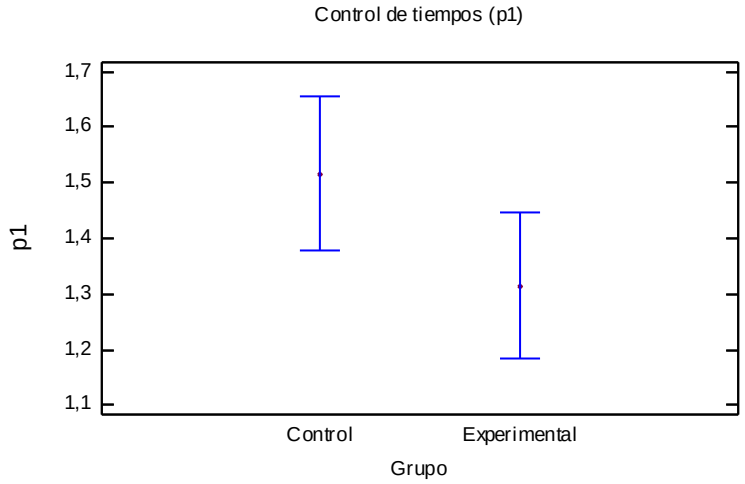


Figura 14. Gráfico de homogeneidad para control de tiempos.

Para la pregunta 2 el análisis indica que no existen diferencias estadísticamente significativas entre ambos grupos, pero al verificar los supuestos se encuentra que no se cumple que los residuos se ajustan a una distribución Normal. Por lo anterior, se acude a la prueba No paramétrica de Kruskay y Wallis, la cual permite afirmar que los grupos son homogéneos. La figura 15 muestra los resultados asociados a la pregunta 2.

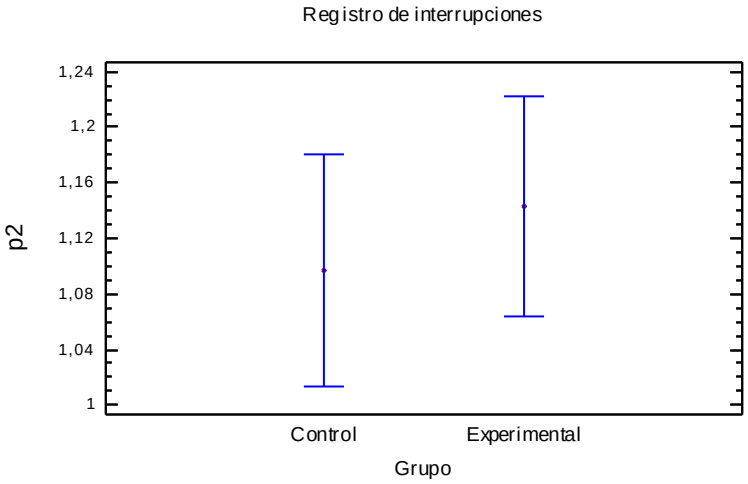


Figura 15. Gráfico de homogeneidad para registro de interrupciones.

Para la pregunta 3 se observan los resultados obtenidos son similares a las anteriores preguntas y con la prueba no paramétrica se puede afirmar que no hay diferencias estadísticamente significativas entre los grupos experimental y control. En la figura 16 se observa el resultado asociado de cada grupo para el registro de errores.

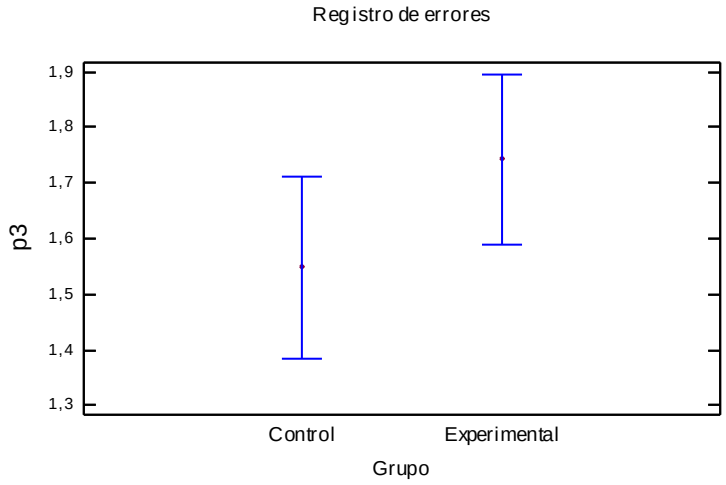


Figura 16. Gráfico de homogeneidad para registro de errores.

El resultado de la pregunta 4 en la prueba No paramétrica, muestra que no existe diferencia significativa entre los grupos de control y experimental. La figura 17 muestra los resultados de los grupos en cuanto a los errores de codificación.

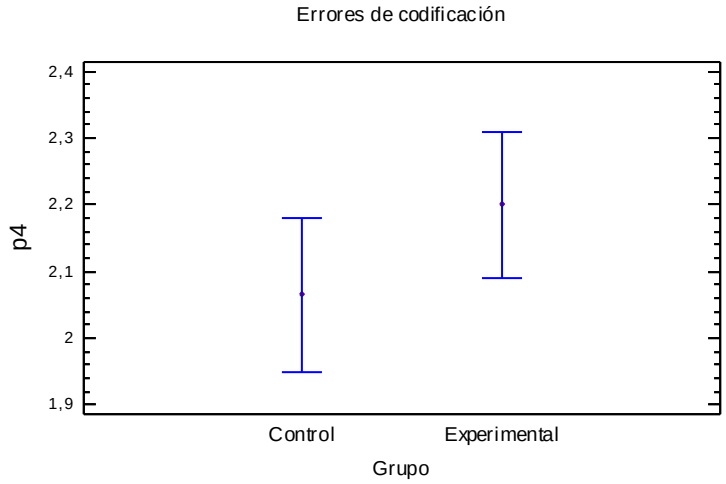


Figura 17. Gráfico de homogeneidad para el registro de errores de codificación.

Para la pregunta 5 se observa que al verificar los supuestos se encuentra que no se cumple que los residuos se ajustan a una distribución Normal; debido a lo anterior se acude a la prueba no paramétrica de Kruskay y Wallis, que permite decir que no existe diferencia significativa entre los grupos control y experimental. La figura 18 muestra el resultado de ambos grupos en relación con las estrategias empleadas para la solución de los errores de codificación.

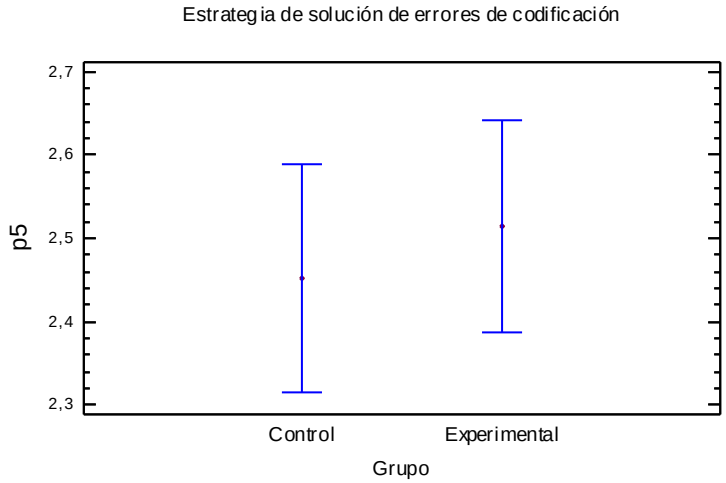


Figura 18. Gráfico de homogeneidad para las estrategias de solución de errores.

Los resultados obtenidos par a la pregunta 6 indican que no existe diferencia estadísticamente significativa entre los grupos control y experimental. La figura 19 muestra el resultado para ambos grupos en relación con los estándares de codificación.

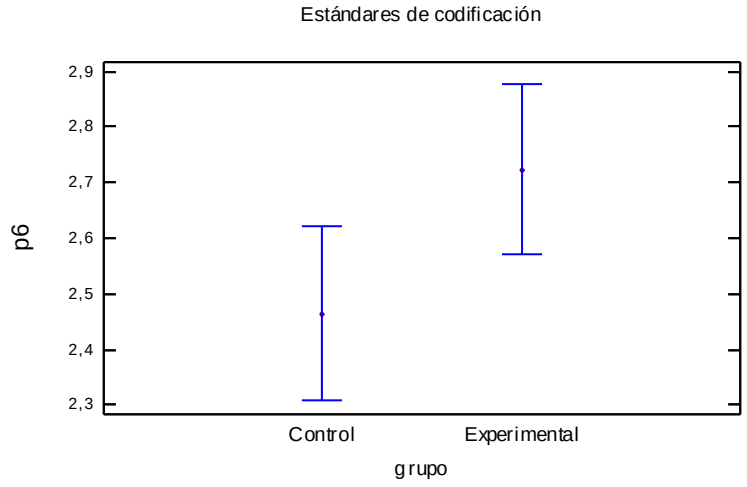


Figura 19. Gráfico de homogeneidad para el uso de estándares de codificación.

El análisis de varianza indica que no hay diferencias estadísticamente significativas entre los grupos experimental y control, al verificar los supuestos se encuentra que se cumple la homogeneidad de las varianzas pero no se cumple que los residuos se ajustan a una distribución Normal; debido a lo anterior se acude a la prueba No paramétrica de Kruskay y Wallis, que permite decir que no existe diferencia significativa entre los grupos control y experimental. La figura 20 muestra los resultados aplicación de estándares de codificación.

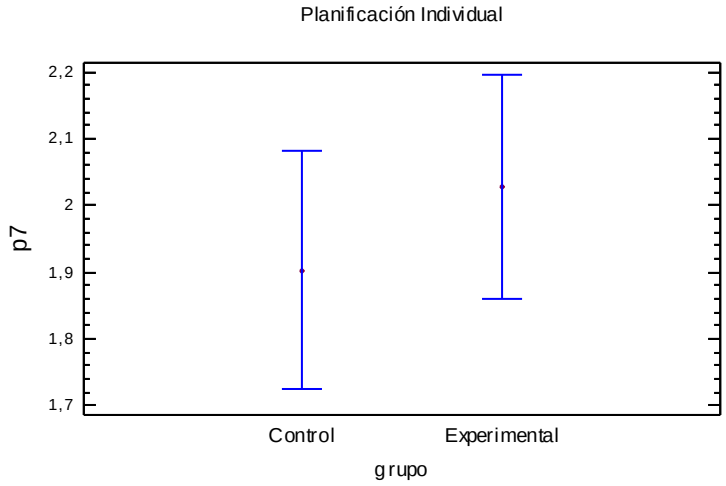


Figura 20. Gráfico de homogeneidad para la planificación individual.

Los resultados obtenidos de la prueba No paramétrica indican que no existe diferencia estadísticamente significativa entre los grupos control y experimental. La figura 21 muestra los resultados de los grupos asociados a la planificación en equipo.

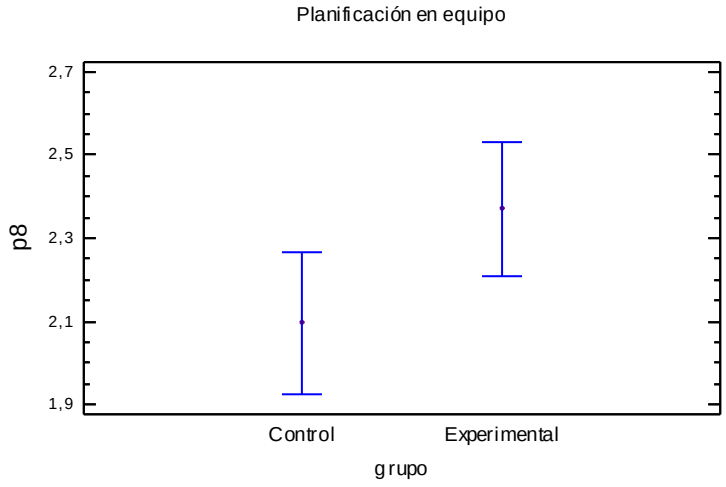


Figura 21. Gráfico de homogeneidad para la planificación en equipo.

El análisis de varianza indica que no hay diferencias estadísticamente significativas entre los grupos experimental y control, al verificar los supuestos se encuentra que se cumple la homogeneidad de las varianzas pero no se cumple que los residuos se ajustan a una distribución Normal; debido a lo anterior se acude a la prueba No paramétrica de Kruskay y Wallis, que permite decir que no existe diferencia significativa entre los grupos control y experimental. La figura 22 muestra los resultados asociados a los roles y las responsabilidades de los equipos.

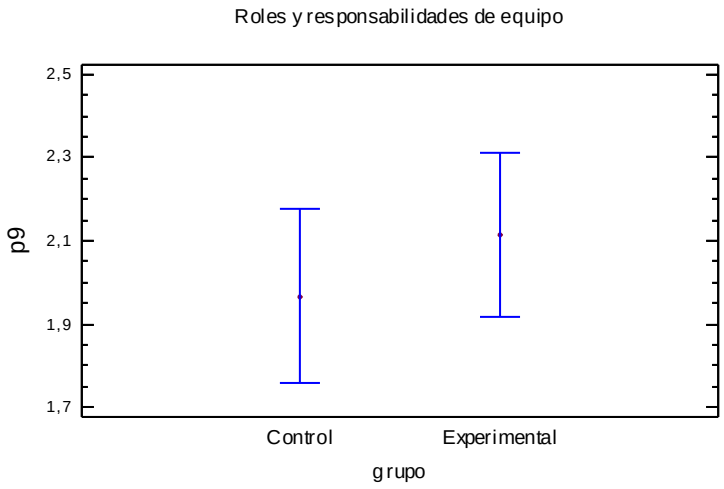


Figura 22. Gráfico de homogeneidad para roles y responsabilidad de equipo.

En la tabla 19 se muestran los resultados del análisis estadístico realizado para cada una de las preguntas realizadas a los estudiantes. Cada una de ellas permite afirmar que tanto el grupo de control y experimental son homogéneos y por lo tanto es posible iniciar con la estrategia de enseñanza en el grupo experimental.

Tabla19. Análisis de homogeneidad por pregunta

Pregunta	p_valor	Homogeneidad de varianzas (levens)	Distribución normal de residuos Shapiro-Wilk	Prueba No paramétrica Kruskay y Wallis
1	0,1409	0,140908	4,35409E-15	0,1848
2	0,5739	0,573945	3,54809E-14	0,569942
3	0,2237	0,413037	7,81931E-10	0,162059
4	0,2356	0,0750583	4,44089E-16	0,204958
5	0,6374	0,808796	1,11022E-16	0,712758
6	0,1495	0,149459	1,26715E-10	0,1848
7	0,4727	0,0511165	1,0578E-9	0,451819

8	0,1023	0,617618	3,42777E-7	0,132956
9	0,4686	0,151367	1,7582E-10	0,48251

Con base en los anteriores resultados se puede afirmar que los grupos de control y experimental son homogéneos en todas las preguntas y por lo tanto fue posible continuar con la metodología definida para la investigación.

6.5.4 Estrategia de aprendizaje grupo experimental

La estrategia didáctica para el grupo experimental se desarrolló a través de etapas, las cuales se van desarrollando en torno al aprendizaje de los estudiantes.

La primera etapa consiste en una introducción y presentación que permita una orientación al curso por parte del profesor. Se establecen los compromisos y la metodología de trabajo dentro del curso. En la segunda etapa se hace énfasis en el aprendizaje significativo, en donde se pretende que el estudiante adquiera nuevos conocimientos. En esta etapa se aprecia el trabajo que realizan los estudiantes de forma individual, reforzado por medio de debates grupales, siempre con el acompañamiento del profesor. En la tercera fase se busca que el nivel de aprendizaje incremente mediante la cooperación. En esta fase se hace mucho más énfasis en la cooperación mediante un proyecto en grupo usando una situación problemica. La última etapa comprende la retroalimentación de los aprendizajes obtenidos y se valida mediante diferentes tipos de pruebas.

6.5.5 Estrategia de aprendizaje grupo de control

En el grupo control se aplicó la metodología tradicional de enseñanza, definida para el curso de Paradigma orientada a objetos, la misma estuvo bajo la responsabilidad de uno de los profesores titulares de la asignatura.

El curso está dividido en 5 unidades temáticas que corresponden a objetivos pedagógicos específicos. El estudiante deberá desarrollar los ejercicios planteados. Al final de cada ejercicio debe obtenerse un programa que pueda ser verificado a través de pruebas. En la parte final del curso se realizará un proyecto en el cual se deben poner en práctica la mayoría de los temas vistos a lo largo del semestre, y algunos temas de investigación adicionales propuestos por el profesor.

Se desarrollaron los temas:

- Unidad 1: Lenguaje de programación Java
- Unidad 2: Programación Condicional
- Unidad 3: Métodos
- Unidad 4: Programación Iterativa
- Unidad 5: Arreglos

7 DISEÑO DE LOS ESPACIOS ACADÉMICOS

La revisión curricular de un curso es un proceso complejo e integral de análisis de los diversos aspectos que conllevan el diseño y las estrategias del mismo; por ejemplo, objetivos, contenidos, metodología de enseñanza y recursos disponibles.

Para el desarrollo de la propuesta de formación se tendrá en cuenta el Proyecto Educativo del Programa y el plan de estudios del programa de Ingeniería de sistemas y Computación de la Universidad del Quindío. En ese sentido se considerarán las competencias definidas para los estudiantes del programa de Ingeniería de Sistemas y Computación.

En este capítulo se hará una recopilación de los elementos fundamentales (temas, subtemas, objetivos de formación y logros esperados) de cada una de las materias a las que se les hará el diseño de la estrategia de enseñanza.

- Paradigma orientado a objetos
- Fundamentos de Algoritmia
- Lenguajes de programación
- Estructuras de datos
- Análisis de algoritmos I

Teniendo en cuenta que el desarrollo del trabajo presenta un reto en cuanto a la incorporación de los conceptos de PSP y TSP a cada uno de los cursos anteriormente mencionados, se debe establecer la forma de distribuirlos uniformemente durante el desarrollo de un semestre en particular y los siguientes semestres. Se pretende incorporar progresivamente los temas semanalmente. Para cada uno de los cursos, a cada uno de los estudiantes se les asignará una serie de tareas por separado; adicionalmente, los estudiantes deben asistir a los Laboratorios de las clases y a las clases mismas, así como también realizar lecturas de los materiales diseñados.

A cada uno de los grupos de trabajo se les asignó un proyecto de programación. Cada uno de ellos involucró aspectos de PSP (registros de tiempo, de defectos y los formularios del resumen), así mismo se involucraron actividades de TSP. Como soporte al proceso, los conceptos se impartían mediante diferentes estrategias de enseñanza.

7.1 Diseño metodológico para las asignaturas.

Para cada uno de los espacios académicos se definieron los objetivos generales de aprendizaje, los logros esperados de la actividad académica, las estrategias de enseñanza generales y los medios de apoyo a la estrategia. Para el desarrollo de lo anterior se tuvo en cuenta el plan de estudios del Programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío. Es posible que las estrategias de enseñanza varíen de acuerdo con la estructuración de la propuesta de formación.

7.2 Diseño para Paradigma orientado a objetos.

Hoy en día se tienen a disposición diversos paradigmas de programación, cada uno fuerte en un dominio en particular, lo que requiere de cierta fundamentación teórica y lógica antes de llegar a dominarlo correctamente.
Es necesario que los estudiantes conozcan conceptos teóricos, con una gran fundamentación de Lógica de programación, que sirva de base para enfrentarse a cualquier paradigma y Lenguaje de programación.

A continuación se describen los objetivos generales, los logros esperados, las estrategias de enseñanza y los medios de apoyo que servirán para estructurar la propuesta.

- Desarrollar la lógica de programación
- Expresar un problema en términos del uso de tipos de datos y métodos.
- Definir métodos y atributos.
- Implementar la solución de un problema haciendo uso del Lenguaje Java.
- Usar un ambiente de desarrollo.
- Hacer uso de estándares de documentación y codificación.
- Utilizar estructuras de decisión y repetitivas.
- Usar adecuadamente estructuras contenedoras de tamaño fijo (arreglos).

Los logros describen las habilidades que se pretenden desarrollar en los estudiantes que cursen este espacio académico.

Logros esperados de la actividad académica
• Desarrollar lógica de programación. • Identificar algunos conceptos importantes para el diseño estructurado. • Conocer y utilizar una herramienta para el uso de lenguaje JAVA. • Analizar problemas sencillos por medio de la utilización de métodos, variables y constantes. • Utilizar instrucciones condicionales simples y compuestas, que nos permitan llegar a distintos casos de solución.

Las estrategias de enseñanza y los medios de apoyo describen los elementos a considerar para la realización de las actividades que se proponen en el curso.

Estrategias de enseñanza y medios de apoyo
• Lecturas de textos relacionados con el tema objeto de estudio. • Informes de lectura grupal sobre los temas objeto de estudio. • Uso de mapas conceptuales tanto para la explicación de temas como para la presentación de tareas individuales y colectivas. • Realización de tareas individuales y en equipo con productos concretos. • Realización de trabajos colaborativos con recursos disponibles en la plataforma Moodle. • Prácticas expositivas del profesor. • El profesor propiciará que el alumno cuestione sobre el proceso de desarrollo y de los ciclos de vida de software en mesas de discusión. • Resolución, autoevaluación y análisis individual / grupal de ejercicios, cuestionarios, trabajos.

En la tabla 20 se muestra la propuesta para la estructura del curso, la cual contiene el nombre la unidad, el contenido temático y los temas de PSP y TSP que se van a desarrollar durante un semestre académico. Es de tener en cuenta que estos contenidos se orientaran de forma transversal durante el curso.

Tabla 20. Estructura del curso Paradigma Orientado a Objetos.

Número de Unidad	Contenido Temático	Temas de PSP
Unidad 1 Lenguaje de programación Java	Conceptos básicos Comentarios de línea y bloque Variables, constantes y nomenclatura Tipos de datos primitivos Conceptos de Objetos Operadores y Expresiones	Introducción al concepto de calidad • Conceptos de calidad de software • La importancia del trabajo de alta calidad
Unidad 2 Programación Condicional	Decisiones simples (*if, if-else*) Decisiones anidadas Decisiones múltiples (*switch*)	Introducción al proceso de desarrollo • Concepto de proceso • Concepto de proceso de desarrollo de software
Unidad 3 Métodos	Conceptos básico de métodos Métodos que retornan valor Métodos que no retornan valor Paso de parámetros	Proceso Personal de Referencia y de Trabajo en grupo • Proceso actual de desarrollo de software • Introducción al concepto de calidad de software • Conceptos básicos de trabajo en equipo
Unidad 4 Programación Iterativa	Contadores Acumuladores Ciclo condicionado al final (*do-while*) Ciclo condicionado al inicio (while - for)	Proceso Personal de Referencia • Introducción al PSP • Introducción al PSP0 • Planificación del tiempo
Unidad 5 Arreglos	Creación de un arreglo Acceso a las posiciones de un arreglo Recorrido de un arreglo Arreglos unidimensionales Métodos de ordenamiento	Proceso Personal de Referencia – • Concepto de gestión y control de tiempo PSP0 • Registro de tiempo y defectos • Estándar de tipos de defectos

Teniendo en cuenta que en este curso se definió trabajar con el proceso de referencia PSP0, se definen estrategias que permitan la aplicación de las prácticas actuales que utilizan los ingenieros para el desarrollo de sus proyectos. Las prácticas definidas son:

- Registro de tiempo para la realización del proyecto.
- Registro de los defectos y de los tipos de defectos.

7.3 Diseño para Fundamentos de Algoritmia

En este espacio académico se pretende que el estudiante obtenga los conocimientos y las habilidades para resolver problemas usando el paradigma orientado a objetos, utilizando el lenguaje de programación Java. Estos conocimientos se obtendrán mediante el trabajo en equipo de manera colaborativa, responsable, honesta y comprometido con su aprendizaje.

A continuación se describen los objetivos generales para el curso de Fundamentos de Algoritmia.

- Identificar las características fundamentales de los arreglos.
- Conocer las operaciones fundamentales que se pueden realizar con los arreglos.
- Definir los conceptos relacionados con las clases y los objetos.
- Determinar las similitudes y diferencias entre clases y objetos.
- Conocer las características de los métodos Constructores, getters and setters.
- Determinar los mecanismos de comunicación entre clases y entre métodos.
- Escribir correctamente la documentación de los programas.
- Realizar pruebas automáticas al código.
- Manejar los conceptos básicos de las interfaces de usuario.
- Aplicar los conceptos básicos de los arreglos de objetos.

Los logros describen las habilidades que se pretenden desarrollar en los estudiantes que cursen este espacio académico.

Logros esperados de la actividad académica
• Aplicar operaciones correctas para el manejo de los arreglos. • Entender las diferencias en la implementación de las clases y los objetos. • Aplicar correctamente el concepto de método constructor con sus diversas variantes. • Documentar los programas de acuerdo con estándares definidos. • Entender el significado de las pruebas unitarias como criterio de calidad. • Construir interfaces de usuario con su correspondiente gestión de eventos.

Las estrategias de enseñanza y los medios de apoyo describen los elementos a considerar para la realización de las actividades que se proponen en el curso.

Estrategias de enseñanza y medios de apoyo
• Lecturas de textos relacionados con el tema objeto de estudio. • Informes de lectura grupal sobre los temas objeto de estudio. • Uso de mapas conceptuales tanto para la explicación de temas como para la presentación de tareas individuales y colectivas. • Realización de tareas individuales y en equipo con productos concretos. • Realización de trabajos colaborativos con recursos disponibles en la plataforma Moodle. • Prácticas expositivas del profesor. • El profesor propiciará que el alumno cuestione sobre el proceso de desarrollo y de los ciclos de vida de software en mesas de discusión. • Resolución, autoevaluación y análisis individual / grupal de ejercicios, cuestionarios, trabajos. • Invitación a un experto del tema que sirva de motivador a los estudiantes.

En la tabla 21 se muestra la propuesta para la estructura del curso, la cual contiene el nombre la unidad, el contenido temático y los temas de PSP y TSP que se van a desarrollar durante un semestre académico. Al igual que en el primer curso de programación, los contenidos se orientaran de forma transversal durante el curso.

Tabla 21. Estructura del curso Fundamentos de Algoritmia

Número de Unidad	Contenido Temático	Temas de PSP - TSP
Unidad 1 Arreglos Multidimensionales	Creación de un arreglo Acceso a las posiciones de un arreglo bidimensional Recorrido de arreglos multidimensional	Proceso Personal de Referencia PSP0 • Registro de tiempo y defectos • Estándar de tipos de defectos • Estrategias de trabajo en equipo
Unidad 2 Fundamentos de programación orientada a objetos	Paquetes, Clases, Objetos Métodos Constructores Atributos Visibilidad Relaciones entre clases Llamado de métodos	Proceso Personal de Referencia – PSP0.1 • Planeación y Medición del Tamaño • Medidas de tamaño • Estimación del Tamaño

	Documentación javadoc Aplicaciones con varias clases	
Unidad 3 Pruebas Unitarias	Buenas prácticas de programación Pruebas unitarias automáticas,	Proceso Personal de Referencia PSP0.1 Conceptos básicos trabajo en equipo • Propuesta de Mejora de Procesos (PIP) • Concepto de roles y responsabilidades
Unidad 4 Interfaz Gráfica	Conceptos básicos Contenedores Componentes gráficos Eventos	Proceso Personal de Referencia – PSP0.1 Conceptos básicos trabajo en equipo • Estándar de Código
Unidad 5 Arreglos de Objetos	Estructuras contenedoras de tamaño fijo Arreglos de objetos Estructuras contenedoras de tamaño variable	Proceso Personal de Referencia PSP0.1 • Medición de tamaño • Asignación de responsabilidades

Teniendo en cuenta que en este curso se definió trabajar con el proceso de referencia PSP0, se definen estrategias que permitan la aplicación de las prácticas actuales que utilizan los ingenieros para el desarrollo de sus proyectos. Las prácticas definidas son:

- Registro de tiempo para la realización del proyecto.
- Registro de los defectos y de los tipos de defectos.
- Resumen del plan de proyecto.
- Estándar para documentar y reportar los tipos de defectos.

En el PSP0.1 se define un método para realizar conteo de líneas de código (LOC) de cada uno de los trabajos que se realicen, como también se establece una forma para que los estudiantes identifiquen y documenten su propio de proceso de desarrollo de software que les permita identificar las oportunidades de mejora en su trabajo.

Los elementos a tener en cuenta para este nivel son:

- Documento que permita documentar una propuesta de mejora del proceso (PIP)

- Definición de un estándar de conteo de líneas de código y de un estándar para la codificación durante el proceso de construcción del producto.

Teniendo en cuenta que dentro del curso se planea la realización de trabajo en equipos, se muestra la importancia de aplicar procesos en un ambiente de trabajo, así como también la definición de los roles y las responsabilidades de cada uno de los integrantes.

7.4 Diseño para Lenguaje de Programación

Este espacio académico da continuidad a las temáticas tratadas en Fundamentos de Algoritmia. Lenguaje de Programación brinda los conocimientos necesarios para que el estudiante resuelva problemas que involucren el uso de estructuras contenedoras y de persistencia. Para lograr este objetivo el estudiante aprenderá técnicas adecuadas que le permitan expresar el modelo del mundo del problema y la arquitectura de la solución.

A continuación se describen los objetivos generales del curso de Lenguaje de Programación.

- Conocer el concepto de herencia entre clases de implementación.
- Entender la implementación de interfaces.
- Conocer el significado de herencia entre interfaces.
- Hacer uso de las estructuras de excepciones de Java.
- Utilizar las instrucciones *throw, throws y finally.*
- Identificar los flujos de entrada y salida en diferentes estructuras.
- Entender el concepto de serialización de objetos.
- Identificar los elementos conceptuales relacionados con los hilos.
- Conocer el concepto de recursividad.

Los logros describen las habilidades que se pretenden desarrollar en los estudiantes que cursen este espacio académico.

Logros esperados de la actividad académica
• Desarrollar programas reutilizables aplicando el concepto de herencia. • Aplicar la herencia entre interfaces como mecanismo de reutilización. • Gestionar adecuadamente las estructuras de excepciones de Java.

• Aplicar correctamente el concepto de flujos de entrada y de salida con diferentes elementos. • Conocer y entender el concepto de hilos de implementación. • Aplicar la recursividad entre métodos como estrategia de implementación.

Las estrategias de enseñanza y los medios de apoyo describen los elementos a considerar para la realización de las actividades que se proponen en el curso.

Estrategias de enseñanza y medios de apoyo
• Realizar juegos con los estudiantes, como una manera de revisar material contenido en las pruebas. • Lecturas de textos relacionados con el tema objeto de estudio. • Informes de lectura grupal sobre los temas objeto de estudio. • Uso de mapas conceptuales tanto para la explicación de temas como para la presentación de tareas individuales y colectivas. • Realización de tareas individuales y en equipo con productos concretos. • Realización de trabajos colaborativos con recursos disponibles en la plataforma Moodle. • Prácticas expositivas del profesor. • El profesor propiciará que el alumno cuestione sobre el proceso de desarrollo y de los ciclos de vida de software en mesas de discusión. • Resolución, autoevaluación y análisis individual / grupal de ejercicios, trabajos.

En la tabla 22 se muestra la propuesta para la estructura del curso, la cual contiene el nombre la unidad, el contenido temático y los temas de PSP y TSP que se van a desarrollar durante un semestre académico.

Tabla 22. Estructura del curso Lenguaje de Programación

Número de Unidad	Contenido Temático	Temas de PSP - TSP
Unidad 1 Herencia	Herencia de la Clase *Object* Herencia entre clases Manejo e implementación de interfaces Herencia entre interfaces	Proceso de Planeación Personal – PSP1 • Estimación de tamaño

Unidad 2 Excepciones	Estructura de excepciones La instrucción try-catch Objetos tipo *Exception* Las instrucciones *throw, throws y finally*	Proceso de Planeación Personal – PSP1 • Método de Estimación PROBE • Reporte de pruebas
Unidad 3 Flujos de Entrada - Salida	Flujo entrada salida de texto Flujo entrada salida binarios Archivos *Properties* Serialización de objetos	Proceso de Planeación Personal – PSP1 • Reporte de pruebas
Unidad 4 Hilos de Ejecución	Estados de un hilo Creación de hilos Sincronización	Proceso de Planeación Personal – PSP1.1 • Planeación de tareas • Planeación de calendarios
Unidad 5 Recursividad	Conceptos básicos de recursividad Caso base – caso inductivo	Proceso de Planeación Personal – PSP1.1 • Estimación de Recursos y Medición del proceso

En el nivel PSP1 en el cual se trabaja la planeación personal, se busca que los estudiantes entiendan la relación existente entre el tamaño del programa (LOC) y el tiempo que invierte en total para su desarrollo. Se pretende incorporar el concepto de planeación y trabajo organizado para que el estudiante pueda realizar estimaciones de tiempo y tamaño. Teniendo en cuenta que se trata de un proceso incremental, en PSP1 se adiciona:

- Un método para estimaciones del tamaño denominado PROBE.
- Método para el reporte de pruebas de software.
- Formato para la estimación del tamaño del producto.

Con relación al PSP1.1 se establece una metodología para que los estudiantes realicen:

- Actividades que permitan la planeación del proyecto y de los tiempos necesarios para su realización. Es necesario en este punto la planeación de tareas.

7.5 Diseño para Estructuras de Datos

Este espacio académico permite que el estudiante tenga un primer acercamiento con las estructuras de datos. El estudio de las estructuras de datos es sumamente importante, pues gracias a ellas es posible manipular la información de manera adecuada y oportuna. Se orienta desde el punto de vista de la abstracción, permitiéndole al estudiante adquirir habilidades para la utilización de técnicas de encapsulación y ocultación de información, logrando con ello que los desarrollos realizados tengan mayor calidad y claridad.

A continuación se describen los objetivos generales, los logros esperados, las estrategias de enseñanza y los medios de apoyo.

- Conocer las Interfaces *Collection, List e Iterator.*
- Entender el concepto de pilas, listas y colas implementando la interfaz *Collection*.
- Identificar las Estructuras de tipo *Map*.
- Entender los elementos conceptuales fundamentales del *Backtracking*.
- Identificar las características principales de los Árboles binarios ordenados.
- Reconocer el recorrido de los árboles: inorden preorden y postorden.
- Representar correctamente el concepto de Árboles AVL.
- Conocer las principales implementaciones de los algoritmos aplicados a los grafos.

Los logros describen las habilidades que se pretenden desarrollar en los estudiantes que cursen este espacio académico.

Logros esperados de la actividad académica
• Aplicar en contextos determinado el concepto de *Collection, List e Iterator.* • Aplicar correctamente los conceptos abstractos de listas, pilas y colar partiendo de la interfaz *Collection*. • Aplicar las estructuras de tipo *Map* en problemas concretos de implementación. • Usar el concepto de *Backtraking* a partir de los elementos fundamentales de la recursividad. • Aplicar en problemas concretos el recorrido de los arboles. • Implementar algoritmos apoyados en los conceptos fundamentales de los grafos.

Las estrategias de enseñanza y los medios de apoyo describen los elementos a considerar para la realización de las actividades que se proponen en el curso.

Estrategias de enseñanza y medios de apoyo
• Ilustración y análisis de casos concretos. • Lecturas de textos relacionados con el tema objeto de estudio. • Informes de lectura grupal sobre los temas objeto de estudio. • Uso de mapas conceptuales tanto para la explicación de temas como para la presentación de tareas individuales y colectivas. • Realización de tareas individuales y en equipo con productos concretos. • Realización de trabajos colaborativos con recursos disponibles en la plataforma Moodle. • Prácticas expositivas del profesor. • El profesor propiciará que el alumno cuestione sobre el proceso de desarrollo y de los ciclos de vida de software en mesas de discusión. • Resolución, autoevaluación y análisis individual / grupal de ejercicios, cuestionarios, trabajos.

En la tabla 23 se muestra la propuesta para la estructura del curso, la cual contiene el nombre la unidad, el contenido temático y los temas de PSP y TSP que se van a desarrollar durante un semestre académico.

Tabla 23. Estructura del curso Estructuras de Datos.

Número de Unidad	Contenido Temático	Temas de PSP - TSP
Unidad 1 Estructuras lineales enlazadas y algoritmos Recursivos	Interfaces *Collection, List e Iterator* Pilas, listas y colas usando la interfaz *Collection* Estructuras de tipo *Map*	Administración de calidad personal – PSP2 • Introducción al proceso de calidad de software. • Métodos para la remoción de defectos,
Unidad 2 Estructuras no lineales y *backtracking*	*Backtracking* Árboles binarios ordenados Recorridos de los árboles inorden preorden y postorden Árboles AVL	Administración de calidad personal PSP2 • El proceso de Diseño • Revisiones de código y diseño
Unidad 3 Grafos	Representación Grafo dirigido Búsqueda en grafos	Administración de calidad personal PSP2.1 • Rol de diseño • Métodos de verificación de diseño.

	Algoritmo de *Dijkstra* Ordenación topológica	

En la implementación del nivel PSP2, se pretende que el estudiante gestione los defectos encontrados mediante una estimación previa, antes de revisar formalmente el programa construido. Los elementos que se utilizarán en este nivel son:

- Formato de revisión de diseño y de revisión de código.

En el nivel PSP2.1, se adiciona una lista para la comprobación y revisión de diseño.

7.6 Diseño para Análisis de Algoritmos

Este espacio académico pretende apropiar las técnicas de diseño, evaluación y análisis de algoritmos, que le permitan al estudiante establecer el grado de complejidad y de eficiencia de cualquier algoritmo que implemente.

A continuación se describen los objetivos generales, los logros esperados, las estrategias de enseñanza y los medios de apoyo.

- Aprender a analizar matemáticamente la eficiencia de los algoritmos.
- Definir técnicas de diseño eficiente de algoritmos.
- Adquirir destreza en la aplicación de estrategias de programación, tales como voraces, dividir/vencer, programación dinámica, para la solución de problemas.
- Evaluar el tiempo de ejecución y el orden de complejidad de los algoritmos clásicos.
- Conocer y evaluar los principales algoritmos que existen para búsqueda, ordenamiento, operaciones polinomiales, operaciones matriciales.
- Conocer y evaluar algoritmos aplicados que usen estructuras de datos simples y complejos.

Los logros describen las habilidades que se pretenden desarrollar en los estudiantes que cursen este espacio académico.

Logros esperados de la actividad académica

- Aplicar las técnicas matemáticas correctas para determinar el nivel de eficiencia de un algoritmo.
- Establecer estrategias de implementación de algoritmos, dadas restricciones de tiempo y de tipo computacional.
- Calcular el tiempo de ejecución de algoritmos iterativos y recursivos.
- Aplicar en contextos determinados algoritmos de ordenamiento, de acuerdo con las posibles restricciones.
- Seleccionar correctamente la estructura de datos, de acuerdo con una serie de restricciones.

Las estrategias de enseñanza y los medios de apoyo describen los elementos a considerar para la realización de las actividades que se proponen en el curso.

Estrategias de enseñanza y medios de apoyo
• Ilustración y análisis de casos concretos. • Lecturas de textos relacionados con el tema objeto de estudio. • Informes de lectura grupal sobre los temas objeto de estudio. • Uso de mapas conceptuales tanto para la explicación de temas como para la presentación de tareas individuales y colectivas. • Realización de tareas individuales y en equipo con productos concretos. • Realización de trabajos colaborativos con recursos disponibles en la plataforma Moodle. • Prácticas expositivas del profesor. • El profesor propiciará que el alumno cuestione sobre el proceso de desarrollo y de los ciclos de vida de software en mesas de discusión. • Resolución, autoevaluación y análisis individual / grupal de ejercicios, cuestionarios, trabajos.

En la tabla 24 se muestra la propuesta para la estructura del curso, la cual contiene el nombre la unidad, el contenido temático y los temas de PSP y TSP que se van a desarrollar durante un semestre académico.

Tabla 24. Estructura del curso Análisis de Algoritmos

Número de Unidad	**Contenido Temático**	**Temas de PSP - TSP**

Unidad 1 Análisis de Algoritmos	Hábitos de diseño de algoritmos Tiempo de ejecución Algoritmos Iterativos y recursivos Notaciones asintóticas Complejidad computacional Solución de recurrencias Algoritmos de ordenamiento	Team Software Process TSP • Panorama al TSP • ¿Qué es TSP? • La lógica del TSP • Estrategia y principios
Unidad 2 Estrategias de Programación	Divide y vencerás Algoritmos Devoradores Programación Dinámica	Team Software Process TSP • Características de liderazgo • Equipos autodirigidos
Unidad 3 Algoritmos aplicados a grafos y árboles	Árboles binarios de búsqueda Juegos y grafos Algoritmos sobre grafos Árboles de recubrimiento mínimo.	Team Software Process TSP • Métodos para medir el tamaño, el tiempo, y defectos.
Unidad 4 Aspectos asociados con la Algoritmia, límites de la algorítmica	Análisis de la complejidad computacional de los grafos Estructuras de datos dinámicas Clases P y NP Algoritmos probabilistas	Team Software Process TSP • Marco de planeación de tareas y el plan de seguimiento.
Unidad 5 Optimización y pruebas de algoritmos	Técnicas de Optimización de Código Pruebas de Corrección	Team Software Process TSP • Comprención de los datos de TSP

En este curso se pretende explicar los conceptos fundamentales TSP, se busca que los estudiantes entiendan y apliquen métodos para entender y medir el impacto de los defectos dentro de los proyectos de desarrollo de software. También entender sobre la importancia de las reuniones y los controles que deben considerar los equipos de trabajo.

8 CAPITULO 4: IMPLEMENTACIÓN Y EVALUACIÓN DEL CURSO PILOTO

La fase de implementación de la estrategia de aprendizaje parte del diseño planteado anteriormente, y busca presentar, mediante un esquema general, todo el diseño del curso. También se pretende identificar las dificultades en los estudiantes, de forma que pueda regularizar su propio sistema de aprendizaje y lo mejore progresivamente.

Adicional a los objetivos propios del curso, con esta materia se pretende los siguientes objetivos que propendan en el desarrollo de habilidades para los estudiantes:

- Implementar un programa mediante un proceso y sus respectivas fases, trabajando de forma individual y en equipo de forma ordenada.
- Conocer y aplicar correctamente cada una de los formatos definidos para el proceso de construcción de un programa.
- Entender cómo se debe medir y analizar un proceso de desarrollo de software personal, que permita entender el desempeño individual y en equipo.
- Conocer la metodología PSP que sirva como referencia para un mejor control en el desarrollo de un programa.
- Identificar los errores más frecuentes durante el proceso de construcción de un programa.
- Aprender a controlar, administrar y mejorar los procesos de desarrollo de software, trabajando de manera individual y en equipo.

Con relación a las actividades de trabajo en equipo, se pretenden los siguientes objetivos para el desarrollo de habilidades:

- Establecer roles y responsabilidades que permitan definir las funciones y las tareas que se deben realizar para la construcción de un programa de computador.
- Aprender a planificar un proyecto mediante fases de un proceso de desarrollo de software.

8.1 Diseño del Curso piloto

El diseño del curso muestra la estructura de cada una de las unidades de la asignatura Paradigma Orientado a Objetos.

A continuación se presenta en detalle la estructura del curso para cada una de las unidades, las cuales constan de los siguientes elementos:

- Unidad: contiene el tema general.
- Subtemas: contiene cada uno de los subtemas de la Unidad.
- Tema de PSP: describe los tópicos que se van a trabajar relacionados con PSP.
- Metodología: contiene la secuencia de acciones que se van a realizar durante el proceso de enseñanza, en ella se muestra la estrategia a seguir por parte del profesor y en donde fundamentalmente se busca generar expectativas y una alta motivación a los estudiantes del curso.
- Actividades: son las acciones a realizar tanto por el profesor como por el estudiante. Detalla la forma en la cual se van a ejecutar las actividades dentro del curso.

En la tabla 25 se muestra la estructura para la unidad I denominada Lenguaje de Programación Java.

Tabla 25. Estructura de la primera unidad del curso

Número de Unidad	Subtemas	Temas de PSP	Metodología	Actividades
Unidad 1: Lenguaje de programación Java	Conceptos Comentarios Variables Constantes Tipos de datos Objetos Operadores y Expresiones	El trabajo del Ingeniero de Software Importancia de una buena Ingeniería de Software La importancia del trabajo de alta calidad Introducción al concepto de calidad en el trabajo	Se presenta por parte del profesor la metodología del curso, con los recursos disponibles y el sistema de evaluación del mismo. Mediante técnica expositiva, el profesor muestra los conceptos fundamentales relacionados con la importancia de incorporar el concepto de calidad en cada una de las actividades que realice. A manera de motivación, se exponen casos de éxito y fracaso en proyectos de desarrollo de software considerados como hitos en esta área del conocimiento. Mediante técnica expositiva, se introduce al alumno en el enfoque basado en procesos en el desarrollo de software.	Se realiza una presentación de cada uno de los participantes en el curso en la cual se conocen sus saberes previos, sus expectativas frente al curso. Los estudiantes se organizan en equipos de trabajo y seleccionan un tópico relacionado con el tema de calidad aplicado en diferentes contextos. Nuevamente, trabajando en grupo y mediante técnica expositiva, los estudiantes deben realizar la presentación de los casos de éxitos más relevantes en proyectos de desarrollo de software. Como trabajo colaborativo, los estudiantes deben entregar un trabajo en el cual se establezca una analogía entre la construcción de un edificio y

		Conceptos de calidad de software.	En los temas relacionados con variables, constantes, tipos de datos, expresiones, objetos, operadores y expresiones; se muestra la importancia de aplicar buenas prácticas para la construcción de aplicaciones.	la construcción de un programa de computador. Los estudiantes en cada uno de los temas fundamentales identifican algunas buenas prácticas de desarrollo de software que deben aplicar para la entrega de sus trabajos.

Como resultado de esta primera unidad se espera que el estudiante identifique la importancia de incorporar el concepto de calidad desde el inicio de su proceso de formación. Así mismo, que conozca las fases de un proceso de desarrollo de software. En la tabla 26 se muestra la estructura para la segunda unidad del curso.

Tabla 26. Estructura de la segunda unidad del curso

Número de Unidad	Subtemas	Temas de PSP	Metodología	Actividades
Unidad 2: Programación Condicional	Decisiones simples (*if, if-else*) Decisiones anidadas Decisiones múltiples (*switch*)	Concepto de proceso de desarrollo de software Conceptos fundamentales del proceso de desarrollo de software.	Mediante técnica expositiva, el profesor formaliza el concepto de proceso de desarrollo de software. Mediante la estrategia de los mapas conceptuales, se realiza la definición de producto, proyecto, tarea, proceso, fase, plan y trabajo.	Los estudiantes deben presentar un primer trabajo de programación en el cual se definen claramente las etapas del proceso de desarrollo de software. Los estudiantes deben construir de forma colaborativa un mapa conceptual que amplíe las definiciones dadas en la metodología.

		Importancia de la planeación del trabajo.	El profesor, mediante técnica expositiva, presenta la importancia de la planeación de cada una de las actividades que se realizan cotidianamente.	Los estudiantes en su cuaderno de ingeniería deben explicar la forma como realizan la planificación de sus actividades y posteriormente en una mesa redonda, exponen a sus compañeros la metodología que aplican.
		Planeación del trabajo	El profesor muestra un esquema básico de cuaderno de trabajo, en el cual resalta que para cada una de las actividades, tareas o proyectos que realizan en la unidad 2, deben entregar un documento que contenga: descripción del trabajo, fecha, proceso, el tiempo estimado y el tiempo real.	Los estudiantes entregan un primer trabajo de programación en el cual adjuntan su cuaderno de trabajo. El profesor analiza cada uno de los cuadernos y su diligenciamiento, con el objetivo de mostrar las diferencias en las primeras estimaciones del tiempo que realizan los estudiantes.

Se espera como resultado de esta segunda unidad que los estudiantes formalicen sus conceptos relacionados con el proceso de desarrollo de software y que de una forma experimental identifiquen y apliquen los conceptos fundamentales de planeación del trabajo. La aplicación de estos temas se debe evidenciar en el cuaderno de trabajo, en el cual el rol del profesor es fundamental, teniendo en cuenta que debe realizar un análisis de la información proporcionada por los estudiantes, y de forma que los motive a diligenciar correctamente los datos en el cuaderno de trabajo.

En la tabla 27 se muestra la estructura de la tercera unidad del curso denominada métodos,

Tabla 27. Estructura de la tercera unidad del curso

Número de Unidad	Subtemas	Temas de PSP - TSP	Metodología	Actividades
Unidad 3: Métodos	Métodos que retornan valores Paso de parámetros	Proceso Personal de Referencia y de Trabajo en grupo.	El profesor, mediante técnica expositiva, muestra importancia de trabajar con un proceso que pueda ser definido y controlado.	Los estudiantes deberán investigar y profundizar en los roles de desempeño dentro de un proceso de desarrollo de software.
			El profesor establecerá los tipos de recursos humanos y los roles que pueden desempeñar los Ingenieros de Software.	El profesor organizará en grupos a los alumnos para que determinen en un trabajo asignado el tipo de roles que van a desempeñar los estudiantes en la realización de un trabajo de programación.
		Conceptos básicos de trabajo individual y en equipo.	El profesor, mediante técnica expositiva, propone las ventajas de trabajar el PSP y el TSP en el desarrollo de software.	Los estudiantes conocen los conceptos fundamentales de los procesos de desarrollo de software y realizan una investigación en la cual deben responder a una serie de preguntas de carácter exploratorio.

Se espera como resultado de esta unidad que los estudiantes conozcan los conceptos fundamentales del trabajo individual y del trabajo en equipo cuando están desarrollando software. Se considera que a partir de este momento el proceso y las actividades que van a desarrollar los estudiantes deberán estar mucho más formalizadas, de forma tal que el proceso evidencia que está definido y planeado correctamente.

En la tabla 28 se muestra la cuarta unidad del curso la cual se denomina programación iterativa y contiene los conceptos fundamentales de los ciclos.

Tabla 28. Estructura de la cuarta unidad del curso

Número de Unidad	Subtemas	Temas de PSP - TSP	Metodología	Actividades
Unidad 4: Programación Iterativa	Contadores Acumuladores Ciclo condicionado al final (*do-while*)	Introducción al PSP Proceso Personal de Referencia – PSP0 , PSP0.1	Mediante técnica expositiva, el profesor presenta los conceptos fundamentales de PSP, para que el estudiante conozca el script del proceso y lo aplique a cada uno de sus trabajos de programación.	El alumno leerá artículos sobre los conceptos fundamentales de PSP0 y PSP 0.1 El alumno en cada una de las tareas de programación debe usar el script de procesos y se asigna por parte del

	Ciclo condicionado al inicio (*while - for*)	Planificación del tiempo en los trabajos	El profesor mediante técnica de exposición, mostrará la importancia de controlar el tiempo en cada una de las actividades que se realizan. El profesor explicará la plantilla de registro de tiempo en la cual se detallan el tiempo de trabajo real y el tiempo de las interrupciones. Se explicará a los estudiantes cómo realizar una estimación de tiempo para sus trabajos y una serie de sugerencias para gestionar el tiempo cuando se esté realizando un trabajo o tarea de programación.	profesor los ejercicios 1A, 2A, 3A y 4A para que los estudiantes conformen grupos de trabajo y desarrollen sus tareas. Para cada una de las tareas de programación que se han diseñado, es necesaria la entrega de la plantilla de tiempos. Con base en los resultados entregados por los estudiantes, el profesor realizará un análisis del rendimiento del trabajo de los grupos. Los estudiantes presentan sus resultados obtenidos en cuanto al tiempo invertido en cada una de sus tareas de programación. Se establece una dinámica de grupo en la cual cada participante retroalimenta a los compañeros sobre su impresión de trabajar con la hoja de registro de tiempo.

Se espera como resultado de esta unidad que los estudiantes apliquen lo referenciado en el tutorial de PSP y que las plantillas de script del proceso y de la planeación de los tiempos se implementen para cada una de las tareas o proyectos que realicen. Como particularidad en

esta unidad, los estudiantes deben trabajar de forma individual y el rendimiento de los mismos será analizado detenidamente por el profesor del curso. En la tabla 29 se muestra la quinta unidad del curso denominada Arreglos,

Tabla 29. Estructura de la quinta unidad del curso

Número de Unidad	Subtemas	Temas de PSP - TSP	Metodología	Actividades
Unidad 5: Arreglos	Creación de un arreglo Acceso a las posiciones de un arreglo Recorrido de un arreglo Arreglos unidimensionales Métodos de ordenamiento	Proceso Personal de Referencia – PSP0 Concepto de gestión y control de tiempo Registro de tiempo y defectos	Mediante técnica expositiva, el profesor presenta los conceptos fundamentales de PSP0, PSP 0.1 Y PSP1 para que el estudiante conozca el script del proceso y lo aplique a cada uno de sus trabajos de programación. El profesor mediante técnica de exposición, mostrará la importancia de controlar el tiempo y los defectos en cada una de las actividades que se realizan. El profesor explicará la plantilla de registro de tiempo en la cual se detallan el tiempo de trabajo real y el tiempo de las interrupciones. También se explicará la plantilla de registro de defectos y mostrará su importancia en cada una de las	El alumno leerá artículos sobre los conceptos fundamentales de PSP0, PSP 0.1 y PSP1. El alumno en cada una de las tareas de programación debe usar el script de procesos y el profesor asigna los ejercicios 5A y 6A para que los estudiantes conformen grupos de trabajo y desarrollen sus tareas. Para cada una de las tareas de programación que se han diseñado, es necesaria la entrega de la plantilla de tiempos y defectos. Con base en los resultados entregados por los estudiantes, el profesor realizará un análisis del rendimiento del trabajo de los grupos. Los estudiantes presentan sus

		Estándar de tipos de defectos. Estándar de codificación de código	actividades que se realizan. El profesor explicará la plantilla de registro de tiempo en la cual se detallan el tiempo de trabajo real y el tiempo de las interrupciones. Se explicará a los estudiantes cómo aplicar un estándar de codificación en sus trabajos	resultados obtenidos en cuanto al tiempo invertido en cada una de sus tareas de programación. Se establece una dinámica de grupo en la cual cada participante retroalimenta a los compañeros sobre su impresión de trabajar con la hoja de registro de tiempo y también sobre los defectos que encontraron con mayor frecuencia.

Se espera como resultado de esta unidad que los estudiantes apliquen lo referenciado en el tutorial de PSP y que las plantillas de script del proceso y de la planeación de los tiempos y defectos se implementen para cada una de las tareas o proyectos que realicen. Los estudiantes deben trabajar de forma individual y el rendimiento de los mismos será analizado detenidamente por el profesor del curso.

8.1.1 Diseño de los escenarios de aprendizaje

Los escenarios planteados deben brindar a los estudiantes los elementos que permitan entender el contexto del objeto de aprendizaje. Para nuestro caso, los procesos de desarrollo PSP y TSP. Es necesario definir el espacio del problema y que los estudiantes lo puedan entender. A continuación se describen las actividades necesarias para el desarrollo de la actividad.

1. Actividades individuales que permitan la recuperación de la información en donde el estudiante identifique fuentes confiables que sirvan de marco de referencia. Para ello puede apoyarse en buscar información en Internet sobre el tema planteado; si se considera necesario, el profesor puede dar las pautas para la búsqueda de información.
2. Actividades participativas en grupo que permitan una discusión alrededor del tema objeto de aprendizaje.
3. Encontrándose en una situación en la cual es necesaria plantear alternativas para solucionar el problema, se genera un espacio para la indagación colectiva en la cual se discute sobre las alternativas, considerando las restricciones dadas.
4. El estudiante expone su solución planteada y el profesor, en caso de ser necesario, debe gestionar los posibles errores de forma tal que le permita al estudiante una concientización de la situación y que le oriente para ir evolucionando en sus competencias. Se aplica en este punto una técnica colaborativa a nivel de debate que permita una discusión sana del tema que se está tratando. También la controversia estructurada puede generar un debate a nivel grupal y su impacto puede ser positivo si se gestiona adecuadamente.
5. Para que este aprendizaje sea significativo, se plantea un reto que contenga una motivación; en este caso, en la actividad se muestran situaciones en las cuales se pueda generar valor agregado a toda la información. Se debe verificar que el estudiante aplique efectivamente los conceptos aprendidos y los pueda trasladar al escenario planteado. Para este caso es necesario realizar una actividad individual que pueda ser controlada y validada por la persona (s), encargada (s) de la actividad.
6. Se definen las conclusiones por parte de los encargados de la actividad y se exponen los puntos de vista de cada uno de los participantes.

Articulado con las actividades anteriormente descritas, los escenarios que se plantean en la estrategia de aprendizaje son los siguientes:

- Escenario 1: Procesos de indagación y análisis.
- Escenario 2: Procesos de solución de problemas
- Escenario 3: Análisis del error
- Escenario 4: Construcción de significado
- Escenario 5: Trabajo colaborativo
- Escenario 6: Procesos emocionales y motivacionales

Para cada uno de los temas del curso están definidos los diferentes escenarios de aprendizaje y que se espera soporten la estrategia definida. A continuación en la tabla 30 se describen detalladamente cada uno de los escenarios que se consideraran dentro del curso y que fundamentan la estrategia de enseñanza.

Tabla 30. Escenarios para la estrategia de enseñanza.

Escenario	Actividades	Acciones	Recursos
Escenario 1 Procesos de indagación y análisis.	Realización de las entrevistas y tabulación de la información (realizada en sesiones anteriores). Búsqueda de técnicas que permitan representar e interpretar los datos obtenidos en las entrevistas.	Crear un escenario para establecer relaciones de confianza y de reconocimiento mutuo. Reconocer en los estudiantes sus expectativas, motivaciones, inquietudes respecto al objeto de aprendizaje. Incentivar en el estudiante su participación activa dentro de la actividad de aprendizaje. Aplicación de técnicas participativas, las preguntas al grupo permitirán una apertura a las expectativas que se puedan dar al inicio de la actividad.	Guía de estrategia de aprendizaje. Recursos en Moodle.
Escenario 2 Procesos de solución de problemas	Representar gráficamente la información suministrada por los estudiantes. Exposiciones individuales y grupales sobre el análisis de los datos registrados.	Identificar los conceptos previos de los aprendices y a partir de la identificación, ya sea de apreciaciones erróneas o correctas. Generar escenarios de indagación en los cuales se identifiquen diferentes alternativas de solución, de acuerdo con unas restricciones. Identificar el espacio del problema, de forma que los estudiantes comprendan los objetivos de aprendizaje que se esperan durante el curso. Desarrollar actividades de recuperación de la información, en donde el	Herramienta para la representación de información (hoja de cálculo) Experto en el tema que gestione la información proporcionada por los estudiantes.

		estudiante construya su conocimiento mediante diversas fuentes de información (internet, foros, artículos).	Diferentes tipos de herramientas TICS
Escenario 3 Análisis del error	Generar preguntas que ayuden a generar espacios de reflexión de su proceso. Establecer relaciones entre los elementos ya disponibles dentro de la estructura cognitiva de los estudiantes y los nuevos conceptos.	Gestionar estos errores para que el estudiante de una forma evolutiva pueda ir desarrollado las competencias esperadas, de acuerdo con un acto de concientización dentro del proceso. Promover trabajo colaborativo como el debate y el foro, pueden generar situaciones sanas de discusión en las cuales a través de la moderación del docente se pueden establecer posiciones diferentes y en muchos casos erróneas, en la cuales a través de una motivación pueden impactar positivamente y aclarar muchas dudas .	Experto en el tema que gestione los posibles errores de los estudiantes para generar concientización de los mismos.
Escenario 4 Construcción de significado	Generar inquietudes alrededor de la actividad, tales como: estamos hablando de un grupo de elementos con características comunes? Exposición de las soluciones, se evalúan y se propone la solución óptima. Se debe justificar.	Establecer vínculos sustantivos entre el contenido por aprender y lo que la persona ya sabe. Instrumentar una estrategia potencialmente interesante que facilite una asimilación dentro de su proceso de aprendizaje y que permita al estudiante que estos conceptos sean utilizados efectivamente cuando se lo exijan, dentro de unas circunstancias y contextos específicos. Dentro de este escenario, las prácticas individuales permiten establecer un conjunto de actividades que pueden ser controladas y validadas por un grupo de profesionales expertos en un contexto determinado.	Experto en el tema que oriente este escenario. Recursos en Moodle.

		La controversia estructurada también puede generar un debate a nivel grupal y su impacto puede ser muy positivo si se gestiona adecuadamente.	
Escenario 5 Trabajo colaborativo	Propiciar escenarios en los cuales los estudiantes puedan desarrollar actividades de cooperación y colaboración, con el fin de construir conocimiento a partir de una construcción colectiva. Estructurar estrategias de aprendizaje en la cuales el rol del estudiante sea altamente activo.	Conformación de grupos en los cuales cada uno de estos debe realizar de forma detallada un análisis del problema debidamente planteado. El profesor debe dar las pautas de búsqueda de esta información, debe procurar ser un coordinador de estas actividades, sin ser demasiado activo.	Experto en el tema que gestione la información proporcionada por los estudiantes. Diferentes tipos de herramientas TICS. Recursos en Moodle.
Escenario 6 Procesos motivacionales	Generar espacios de motivación para los estudiantes.	Es necesario favorecer el desarrollo de actitudes que representen felicidad a los estudiantes de una forma duradera. El experto debe resaltar el compromiso de los estudiantes con la actividad.	Experto en el tema que gestione la información proporcionada por los estudiantes.

8.1.2 Plan de Evaluación

El sistema de evaluación definido para la estrategia de aprendizaje considerará una serie de criterios que permitirán la consecución de los objetivos propuestos en los cursos. A continuación se presentan las que se tendrán en cuenta:

- De acuerdo con las actividades propuestas, se realizará observación de las actitudes y de las habilidades que van desarrollando los estudiantes.
- Durante el desarrollo de las actividades en clase se realizarán preguntas intencionadas a los estudiantes.
- Seguimiento al desarrollo de las prácticas que realizan los estudiantes en el laboratorio.
- Seguimiento a las tareas que deben desarrollar los estudiantes durante su trabajo independiente.
- Realización de evaluaciones de forma individual y en equipo.

La evaluación de cada uno de los cursos se basa en la evaluación continua, donde se realiza una constante verificación del avance de los diferentes trabajos realizado por los estudiantes. El seguimiento de los trabajos se realizará de forma semanal, utilizando diferentes estrategias y recursos.

8.2 Desarrollo de la estrategia de aprendizaje

Considerando lo indicado en los ítems de la fase anterior, el primer paso que se llevó a cabo fue realizar un proceso de búsqueda para ubicar diferentes recursos de información y de aprendizaje externos que sirvieran como material, como contenidos para el desarrollo del curso piloto. Tras una búsqueda de varias semanas se identificaron varios recursos que serían adecuados para los fines esperados para cada Módulo o subtema de los mismos. Estos luego fueron seleccionados considerando los criterios acogidos, e igualmente se identificó cuáles contenidos deberían ser de elaboración propia para ajustarse a lo pretendido, como contenido para el curso.

A continuación se realiza la descripción de las unidades que se implementaron para la estrategia de aprendizaje. Los temas fueron: Estructuras repetitivas, Arreglos Unidimensionales y Métodos de ordenamiento.

8.2.1 Guía estructuras repetitivas

A continuación se presenta una guía de aprendizaje para las estructuras de control repetitivas.

GUÍA ESTRUCTURAS REPETITIVAS	
Nombre de la Actividad: Introducción a los ciclos condicionados al inicio	Duración en horas
Resultados de Aprendizaje • Identificar las fases de un proceso de desarrollo de software. • Conocer la estructura de los ciclos condicionados al inicio. • Aplicar el concepto de ciclo para resolver problemas sencillos. • Resolver problemas en los cuales se aplique la iteración. • Aplicar el concepto de método a problemas que impliquen iteración. • Comprender y aplicar los conceptos de acumuladores y contadores.	8 horas

MOTIVACIÓN
Actualmente la construcción de aplicaciones informáticas debe responder a requerimientos muy complejos y de carácter crítico de las organizaciones. Es por ello que 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. La calidad del software depende en gran medida de la calidad del proceso empleado. Los procesos están diseñados para ayudar a los Ingenieros a organizar y planificar su trabajo, el seguimiento de su rendimiento, la gestión de defectos de software, y analizar y mejorar su proceso de desarrollo personal. La siguiente gráfica muestra las actividades que se deben emplear dentro de un proceso de desarrollo de software.

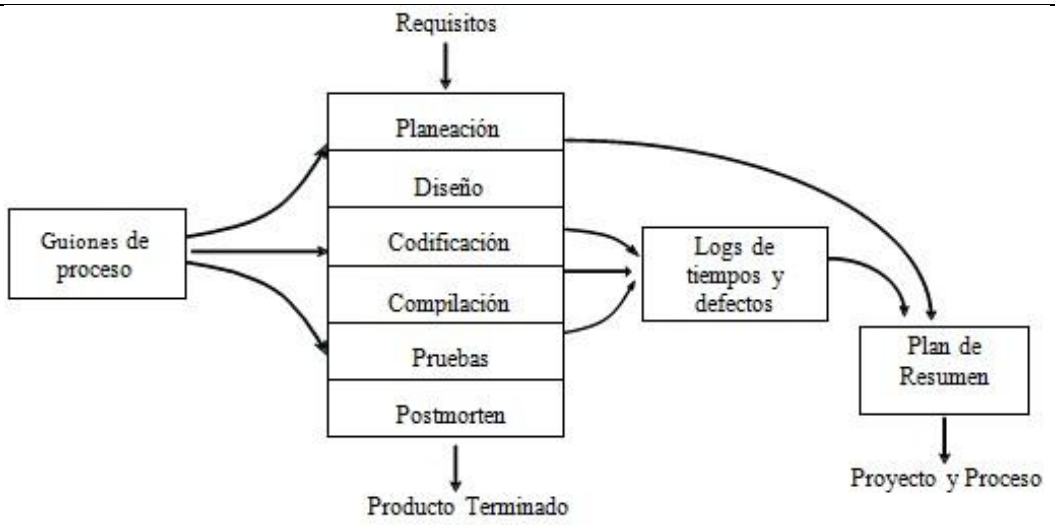


Figura 23. Flujo de procesos de PSP (Humphrey W. , The Personal Software Process, 2000)

Se observa que todo empieza por los requisitos del cliente, a partir de los cuales se debe realizar una planeación para comenzar el trabajo. A nivel de ingeniería se realizan diseños con diagramas de clases, posteriormente se codifica usando un lenguaje de programación, se compila y mediante una serie de pruebas se verifica la funcionalidad del producto. En el postmorten se evalúan los resultados del desarrollo del proyecto. Cada una de estas actividades se realiza de forma secuencial. Paralelo a estas actividades, el ingeniero se puede apoyar en registros de tiempo y de defectos para analizar su desarrollo personal de software.

Nota: Ver guía de actividades – Actividad Motivación

DESCRIPCIÓN DEL PROCESO
El trabajo de un ingeniero gira alrededor de un proceso de desarrollo de software que le permita incorporar los conceptos básicos de calidad en sus prácticas personales. El plan de proyecto permite realizar la planeación de las actividades realizadas durante el proceso de desarrollo de software. Para tener un control del tiempo de las actividades realizadas durante el desarrollo de cada una de las tareas, se tiene la plantilla de registro de tiempo. El objetivo es registrar el tiempo que se invierte y se tarda en la realización de cada uno de los trabajos de programación. A continuación en la figura 24 se muestra un mapa conceptual con los elementos fundamentales de un proceso de desarrollo de software.

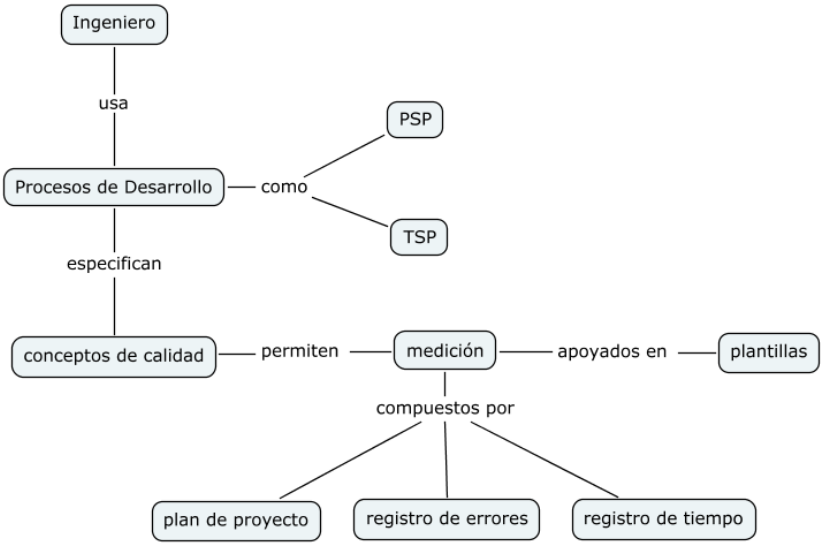


Figura 24. Mapa conceptual proceso de desarrollo.

También es necesario llevar un control del registro de los defectos. Cuando se habla de defectos se hace referencia a los errores producidos durante el desarrollo de todo un Proyecto, desde su fase de planeación hasta la fase de pruebas.

Nota: Ver guía de actividades – Actividad Proceso

INTRODUCCIÓN A LAS ESTRUCTURAS DE CONTROL

En los temas anteriores del Curso se escribieron programas que se ejecutan una sola vez, y en donde se encontraba una secuencia de instrucciones, una tras otra, hasta llegar a un final. Hay ocasiones en que un programa necesita realizar la misma tarea repetidamente por medio de instrucciones, estas instrucciones reciben el nombre de estructuras de control o ciclos, ya que con ellos se puede lograr que una instrucción o un conjunto de instrucciones se ejecuten más de una vez, dependiendo de una condición (expresión lógica). El siguiente mapa conceptual define los elementos fundamentales. La figura 25 muestra un mapa conceptual para las estructuras de control.

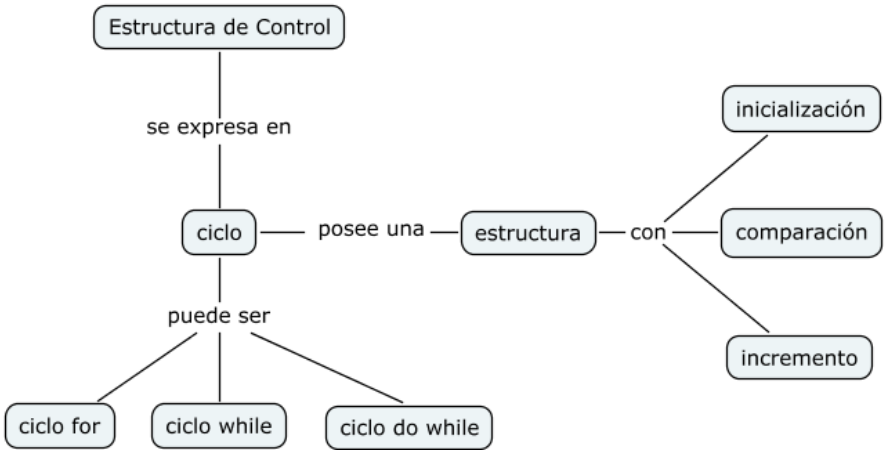


Figura 25. Mapa conceptual estructuras de control.

EL CICLO FOR

Es una estructura que permite que un conjunto de órdenes se ejecuten un número determinado de veces. Por lo tanto, el ciclo *for* permite realizar acciones repetidas sobre un conjunto de instrucciones. El ciclo *for* se compone de cuatro partes fundamentales:

- El valor de inicialización del ciclo
- La condición bajo la cual el ciclo se repite
- Los incrementos que cambian el estado del ciclo
- La instrucción o conjunto de instrucciones del ciclo

La forma como se define un ciclo *for* es la siguiente:

```
for (inicialización, condición o condiciones, incremento)
{
      // instrucción o conjunto de instrucciones
}
```

- Inicialización está relacionada con el valor que toma la variable al comenzar el ciclo.
- La parte de condición o condiciones está relacionada con una expresión booleana. Mientras esta expresión booleana en su evaluación sea verdadera, el ciclo *for* ejecuta la instrucción o el conjunto de instrucciones que se encuentran dentro del bloque marcado por los corchetes {}.
- El incremento marca cómo se modifica el valor de la variable de inicialización.

Gráficamente se puede identificar que existe una condición, que en el caso de ser verdadera, siempre ejecuta el bloque de instrucciones y vuelve a ella para evaluarse. Cuando la condición se evalúe como falsa, termina la ejecución del ciclo y continúa con el conjunto de instrucciones posteriores. En la figura 26 se muestra un gráfico con la estructura para el ciclo *for*.

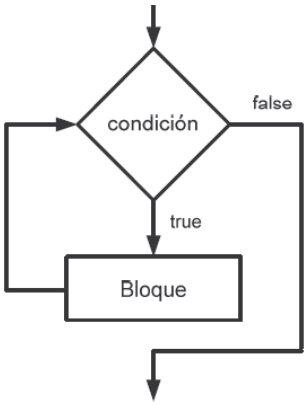


Figura 26. Mapa conceptual estructuras de control. (Berzal, 2012)

Es recomendable no cambiar el valor de la variable que controla el *for* dentro del cuerpo del ciclo, ya que se pueden tener resultados inesperados.

EJEMPLO DE CICLO FOR

El siguiente programa suma los cinco primeros números enteros positivos, utilizando la estructura de control *for*.

```
public class ejemplo{
      public static void main(String[] args) {
            int suma =0;
            for (int i=0; i <= 5; i++)
            {
                  suma+=i;
            }
            JOptionPane.showMessageDialog(null," Valor: " + suma);
      }
```

```
}
```

A continuación en la tabla 31 se muestra la prueba de escritorio con las instrucciones y el resultado final.

Tabla 31. Prueba de escritorio para el ciclo *for*.

i	Suma	i < = 5	suma+=i	i++	Valor suma:
0	0	0 <= 5 (verdadero)	0	1	
1		1 <= 5 (verdadero)	1	2	
2		2 <= 5 (verdadero)	3	3	
3		3 <= 5 (verdadero)	6	4	
4		4 <= 5 (verdadero)	10	5	
5		5 <=5 (verdadero)	15	6	
6		6 <= 5 (falso)			15

EJEMPLO DE APLICACIÓN DE CICLO FOR – NÚMEROS PRIMOS

Se desea determinar si un número entero positivo predeterminado es primo o no lo es. El conjunto de los números primos es un subconjunto de los números naturales que contienen aquellos números que son divisibles exactamente por sí mismos y por la unidad. Son números primos entre otros: 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31. A continuación se muestran dos métodos que solucionan este problema.

```
public boolean esPrimo( int numero ){
      int i, resultado;
      resultado = 0;
      for ( i = 2 ; i <= numero / 2 ; i++ ){
            if ( numero % i == 0 )
            {
                  resultado = 1;
            }
      }
      if ( resultado == 0 ){
            return true;
      }
      else{
            return false;
```

```
        }
    }
```

Una segunda implementación del método primo se muestra a continuación.

```
    public boolean esPrimo1( int numero )
    {
        for ( int i = 2 ; i <= (int)Math.sqrt( numero ) ; i++ )
        {
            if ( numero % i == 0 )
            {
                return false;
            }
        }
    return true;
    }
```

Nota: ver guía de actividades – Ciclo *for*

EL CICLO WHILE

Esta sección muestra otro tipo de ciclo con el cual es posible trabajar en Java, denominado ciclo *while*. Este ciclo tiene los mismos elementos del ciclo *for*. La diferencia fundamental radica en la posición de los elementos dentro de su estructura. Por lo tanto el ciclo *while* se compone de cuatro partes fundamentales:

- El valor de inicialización del ciclo
- La condición bajo la cual el ciclo se repite
- Los incrementos que cambian el estado del ciclo
- La instrucción o conjunto de instrucciones del ciclo.

La forma básica como se define un ciclo while es la siguiente:

```
    while (condición o condiciones)
    {
        // instrucción o conjunto de instrucciones
```

```
}
```

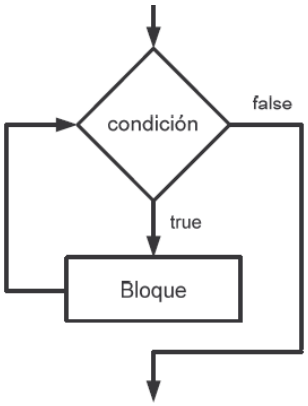


Figura 27. Mapa conceptual estructuras de control. (Berzal, 2012)

Donde condición o condiciones es una expresión booleana. El ciclo *while*, al igual que el ciclo *for*, se ejecuta mientras la evaluación de la expresión booleana sea true. En caso de ser la expresión falsa, termina la ejecución del ciclo y se continúa con el flujo normal del programa.

EJEMPLO DE CICLO WHILE

A continuación se tiene un ejemplo con el ciclo *while*. Calcula el factorial de un número entero positivo.

```
public class factorial {
      public static void main(String[] args)
      {
            int i,res=1, numero;
            String valor = "";
            valor=JOptionPane.showInputDialog("Ingrese número:");
            numero = Integer.parseInt( valor );

            i=numero;
            while(i >=1){
                  result*=i;
                  i--;
            }
            JOptionPane.showMessageDialog(null,"Factorial: "+ res);
```

```
        }
    }
```

En la figura 32, se muestra el comportamiento matemático para el factorial de un número.

Tabla 32. Valores para el número factorial

Numero Entero	Factorial
0!	1
1!	1
2!	2 * 1 = 2 * 1!
3!	3 * 2 * 1 = 3 * 2!
4!	4 * 3 * 2 * 1 = 4 * 3!
n!	n (n – 1)!

El siguiente programa solicita al usuario el ingreso de un número entero positivo y posteriormente muestra todos los números divisores del valor que se ingresó.

```
public class divisores {
      public static void main(String[] args)
      {
            int val,numeros;
            String valor = "", resultado = " ";
            valor=JOptionPane.showInputDialog("Ingrese valor: ");
            val = Integer.parseInt( valor );

            numeros = val;
            while (numeros>0)
            {
                  if (val%numeros == 0){
                        resultado+=numeros+"\n";
                  }
            numeros--;
            }
            JOptionPane.showMessageDialog(null,"Divisores de " +
            val + " :\n" + resultado);
            }
```

```
}
```

EJEMPLO DE APLICACIÓN DE CICLO WHILE – SERIE DE FIBONACCI

Introducidos por Leonardo de Pisa, quien se llamaba a sí mismo Fibonacci. Cada número de la secuencia se obtiene sumando los dos anteriores. Por definición, los dos primeros valores son 0 y 1 respectivamente. Los otros números de la sucesión se calculan sumando los dos números que le preceden. A continuación se muestran los primeros 12 números de Fibonacci.

Tabla 33. Números de la serie de Fibonacci

n	F_n	n	F_n
0	0	6	8
1	1	7	13
2	1	8	21
3	2	9	34
4	3	10	55
5	5	11	89

La siguiente es la implementación del método de la serie, el cual dado un número entero positivo, retorna su equivalente en la serie. Por ejemplo si se ingresa n=6, el resultado que genera el método es 8.

```
public int fibo1(int n){
      int penult = 0;
      int ult = 1;
      int sig = 0;
      int i = 1;
      while(i < n){
            sig = penult + ult;
            penult = ult;
            ult = sig;
            i++;
      }
      return sig;
}
```

Nota: Ver guía de actividades – Ciclo while

EL CICLO DO – WHILE

Es una estructura de control repetitivo con la condición al final. A diferencia de los ciclos *while* y *for*, el ciclo *do-while* permite que al menos una vez se ingrese al cuerpo de este ciclo.

La forma básica como se define un ciclo *do-while* es la siguiente:

```
do{
      // instrucción o conjunto de instrucciones
} while (condición o condiciones);
```

La palabra *do* es una palabra reservada e indica el comienzo del ciclo, luego se tienen las instrucciones; después se evalúa la condición que se encuentra en el *while*. Si la condición es verdadera, se regresa al código *do* y se vuelven a ejecutar los pasos hasta encontrar el código *while*, donde se evalúa una vez más en la condición; el proceso se repite hasta cuando la condición sea falsa y el programa continúa. En este bloque de instrucciones es posible tomar decisiones, codificar otros ciclos y en general escribir cualquier instrucción.

Dentro de los usos del *do-while* cabe destacar la validación de los datos de entrada. Validar un dato de entrada es condicionar al programa para que únicamente acepte determinados valores.

EJEMPLO DEL CICLO *DO-WHILE*

El siguiente programa validará un rango de valores de tipo entero, el ciclo se repetirá hasta cuando el usuario ingrese el valor 1 o el valor 0. En el caso de ingresar un valor diferente, el ciclo operará de forma repetitiva.

```
public class ejemplo
{
      public static void main(String[] args) {
            String  entrada;
            int     numero;
            do{
                  entrada=JOptionPane.showInputDialog("Digite 1 ó 2:");
            numero  = Integer.parseInt ( entrada );
```

```
            }
            while ( numero < 1 || numero > 2 );
      }
}
```

La ventana de diálogo solicitará al usuario que ingrese el número 1 ó el 2; si se ingresa un valor diferente, esta ventana de diálogo volverá a aparecer solicitando nuevamente el ingreso de ese valor. El ciclo *do-while* se repetirá hasta cuando la evaluación de la opción sea falsa.

EJEMPLO DE APLICACIÓN DEL CICLO *DO-WHILE* – NÚMEROS DE LUCAS

Otro ejemplo clásico de programación está relacionado con los números de Lucas. La sucesión de los números de Lucas están muy relacionados con la serie de Fibonacci. En la tabla se muestra los primeros 10 números de Lucas, donde n es el número ingresado por parámetro y Ln es el número correspondiente en la serie de Lucas, la cual se muestra en la tabla 34.

Tabla 34. Números de la serie de Lucas

n	Ln	n	Ln
0	2	5	11
1	1	6	18
2	3	7	29
3	4	8	47
4	7	9	76

```
public class lucas {
      public static void main(String[] args) {
            int x = 1, y = 3, i = 2, z, n;
            String valor ="";
            valor=JOptionPane.showInputDialog(null,"Ingrese número:");
            n = Integer.parseInt(valor);
            do
            {
                  z = x + y;
                  x = y;
                  y = z;
                  i++;
```

```
            }
            while(i<n);
            JOptionPane.showMessageDialog(null,"número en serie: "+y);
      }
}
```

A continuación se muestra otro programa en el cual el sistema solicita al usuario un número entero del 1 al 7 y escribe el día de la semana correspondiente, teniendo en cuenta que 1 es lunes, 2 es martes, 3 es miércoles, y así sucesivamente hasta llegar al día 7, que corresponde al domingo.

```
public class dias {
      public static void main(String[] args)  {

            String  entrada;
            int     num;
            do
            {
                  entrada=JOptionPane.showInputDialog("Número entre 1 y 4");
                  num  = Integer.parseInt ( entrada );
            }
            while ( num < 1 || num > 7 );

            switch(num)
            {
                  case 1:
                  JOptionPane.showMessageDialog(null,"Día"+num+"Lunes" );
                  break;
                  case 2:
                  JOptionPane.showMessageDialog(null,"Día"+num+"Martes" );
                  break;
                  case 3:
                  JOptionPane.showMessageDialog(nul,"Día"+num+"Miercoles" );
                  break;
                  default:
                  JOptionPane.showMessageDialog(null,"Día"+num+" Jueves" );
            }
      }
```

```
}
```

8.2.2 Actividades estructuras repetitivas

A continuación se detallan cada una de las actividades que se debe realizar para las estructuras de control repetitivas. Adicional a la realización de los ejercicios de programación propuestos, se deben diligenciar una serie de formatos. Este trabajo se debe realizar de forma individual.

INTRODUCCIÓN
El desarrollo de software implica la realización de un conjunto de actividades encaminadas a la construcción de productos de calidad. Una primera actividad está relacionada con la elaboración de un plan de proyecto o plan de trabajo que sirva como guía para la correcta realización de los trabajos. En su forma más básica, un plan de proyecto es un texto descriptivo del proyecto, que contiene: los objetivos, el método de trabajo y el alcance. Los objetivos son los resultados que obtendremos, el método de trabajo es un texto descriptivo de las actividades que se va a realizar y el alcance son las entregas.

ACTIVIDAD – PROCESO DE DESARROLLO DE SOFTWARE
Describa con sus propias palabras, cada una de las actividades del proceso de desarrollo de software: Requisitos: Planeación: Diseño: Codificación: Compilación:

Pruebas:

Postmorten:

A continuación se describen cuatro ejercicios que requieren el uso de ciclos. Para cada uno de ellos debe realizar las actividades propuestas:

ACTIVIDAD - EJERCICIO 1A

El conjunto de los números primos es un subconjunto de los números naturales que contienen aquellos números que son divisibles por sí mismos y por la unidad. Son números primos entre otros: 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31. La teoría de números está basada en el estudio de los números primos. Un uso muy importante de los números primos ha sido la codificación de claves secretas, mediante la utilización de dos números primos grandes, con los cuales, se puede hacer claves que son casi imposibles de descifrar por una tercera persona; este fue un método utilizado en la Segunda Guerra Mundial.

Requerimientos del Programa 1A

- Dado un número positivo, mostrar los números primos que existan hasta ese número; por ejemplo, si se ingresa el número n = 20, entonces el programa debe mostrar: 1, 2, 3, 5, 7, 11, 13, 17.
- El método primo que se implemente debe ser diferente a los mostrados en la guía ciclos, que se publicó en la plataforma Moodle.
- Durante el desarrollo del trabajo debe diligenciar el formato de registro de tiempos.
- Durante el desarrollo del trabajo debe diligenciar el formato de registro de errores.

Planeación: Describa la forma como va a planear la realización del ejercicio 1A

ACTIVIDAD - EJERCICIO 2A

Las series de números tienen constantes aplicaciones con los ciclos. A continuación se presentan los requerimientos para el ejercicio 2A.

Requerimientos del Programa 2A

Dado un número entero positivo, construya un programa que lea la cantidad de términos a generar de la siguiente serie: 1, 3, 4, 7, 11, 18. Los primeros dos términos de la serie son 1 y 3, el siguiente termino se genera al sumar los dos términos anteriores y así sucesivamente. Por ejemplo, si el usuario ingresa el número 4, se debe mostrar la serie: 1, 3, 7, 11.

- El usuario debe ingresar el valor n positivo
- Durante el desarrollo del trabajo debe diligenciar el formato de registro de tiempos.
- Durante el desarrollo del trabajo debe diligenciar el formato de registro de errores.

Planeación: Describa la forma como va a planear la realización del ejercicio 2A.

ACTIVIDAD - EJERCICIO 3A
Las series de números tienen constantes aplicaciones con los ciclos. A continuación se presentan los requerimientos para el ejercicio 3A. **Requerimientos del Programa 3A** • Dado un número entero positivo, construya un programa que lea la cantidad de términos a generar de la siguiente serie: 1,1, 2, 4, 8, 16, 32, 64, 128. • El usuario debe ingresar el valor n positivo. Por ejemplo, si el usuario ingresa n=5, se debe mostrar la serie: 1, 1, 2, 4, 8. **Planeación:** Describa la forma como va a planear la realización del ejercicio 3A.

8.2.3 Guía Estructuras Contenedoras

A continuación se presenta una guía de aprendizaje para las estructuras contenedoras.

GUÍA ESTRUCTURAS CONTENEDORAS	
Nombre de la Actividad: Introducción a las estructuras contenedoras	Duración en horas
Resultados de Aprendizaje • Identificar los conceptos fundamentales de las estructuras contenedoras. • Aplicar los conceptos de ciclos a las estructuras contenedoras.	12 horas

• Resolver problemas en los cuales se aplique la iteración en las estructuras contenedoras. • Conocer y aplicar el concepto de arreglo como estructura contenedoras de tamaño fijo. • Utilizar las estructuras contenedoras de tamaño, los cuales permiten almacenar una secuencia de valores.	

MOTIVACIÓN
La construcción de aplicaciones informáticas implica la manipulación de grandes volúmenes de información. Es por ello que es necesario que los estudiantes conozcan y apliquen los conceptos fundamentales de las estructuras contenedoras de tamaño fijo, con las cuales es posible la realización de una serie de operaciones que permiten la manipulación de información del mismo tipo.

INTRODUCCIÓN A LAS ESTRUCTURAS DE CONTENEDORAS
Es frecuente encontrar problemas en donde se necesita almacenar información y a través de la cual se pueda realizar diferentes tipos de operaciones. Para ello es posible utilizar los arreglos los cuales se pueden definir como objetos en los que se puede guardar más de una variable, es decir, al tener un único arreglo, este puede guardar múltiples variables de acuerdo con su tamaño o capacidad. El siguiente mapa conceptual de la figura 28 define los elementos fundamentales de las estructuras contenedoras.

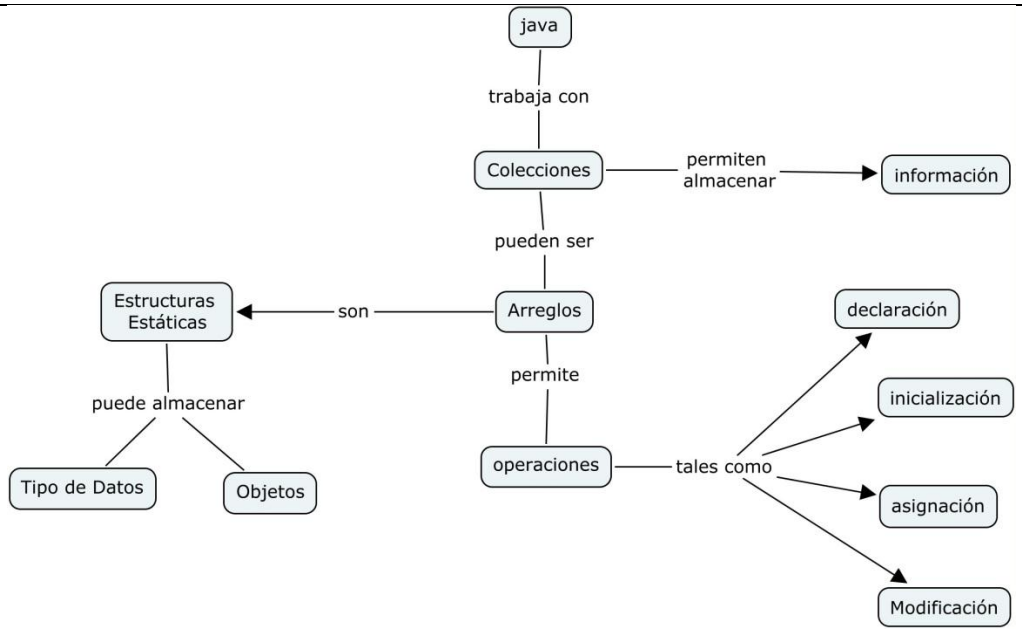


Figura 28. Mapa conceptual estructuras contenedoras

ARREGLOS UNIDIMENSIONALES

Un arreglo unidimensional es una colección de elementos del mismo tipo agrupados bajo un nombre de variable, que se diferencian entre sí por el número de índice. Cada celda de este tipo especial de variable es numerada del 0 hasta a n-1, donde n es el tamaño y determina su capacidad para almacenar elementos. Los índices de un arreglo en Java deben estar dentro de los límites, 0 hasta n-1, de lo contrario se generará un error durante la ejecución.

A continuación se describen las operaciones básicas de los arreglos:

- **Declaración de un arreglo:** Un arreglo se puede declarar usando una de las dos formas siguientes:

```
int vector [ ];                int [ ] vector;
```

  En forma general, para declarar un arreglo es similar a la declaración de una variable; es decir, se especifica el tipo de dato y el nombre del arreglo. Observe el uso de los corchetes [ ] vacíos, antes o después del nombre del arreglo.

- **Creación de un arreglo:** Después de declarar un arreglo, es necesario crearlo para poder utilizarlo en un programa en Java. Es en el momento de la creación donde se especifica el

tamaño del arreglo.

```
vector = new int [ n ];
```

donde vector es el nombre del arreglo y n es el tamaño reservado; es decir, la cantidad de elementos que pueden ser utilizadas dentro del arreglo. Es importante anotar que n puede ser una constante, variable o un literal (un número claramente especificado), por ejemplo 8.

Cuando Java encuentra la declaración de un arreglo

```
float nota [ ];
```

sabe que el usuario utilizará un arreglo llamado nota. Más adelante, deberá indicar el número de posiciones que desea utilizar. En este caso, si utiliza la instrucción

```
nota = new float [ 50 ];
```

para crear el arreglo, le indicará a Java que desea reservar 200 bytes contiguos para el arreglo nota, de tal manera que en cada espacio de 4 bytes sea almacenado un número flotante perteneciente al arreglo. Para identificar cada uno de los elementos de un arreglo, se debe usar un valor, llamado subíndice, que señala la posición del elemento dentro del arreglo.

Los arreglos permiten almacenar un conjunto de elementos los cuales comparten el mismo tipo de dato. Como aspecto fundamental, cada uno de los elementos tiene una posición que lo identifica, lo cual permite un fácil acceso a la información que posea. Java permite el uso de arreglos de una o más dimensiones.

UBICACIÓN DE LOS ELEMENTOS DENTRO DE UN ARREGLO

La manipulación de datos es una de las principales actividades que realizan los programas, por tal motivo se han creado varias estructuras que permiten manejar los datos de diferentes maneras, pero siempre buscando la optimización en su manejo. Una de las primeras estructuras de datos que se estudia son los arreglos.

En los capítulos anteriores se escribieron programas que procesaban información independiente. Los datos eran procesados y leídos unos tras otro, cada que se hacía una nueva lectura en las mismas variables, la información que tenían almacenada se perdía. Con los arreglos se puede procesar un conjunto de datos que pueden mantenerse en la memoria del computador y disponer

de cada uno en el momento en el que se requiera. Esto es una gran ventaja de usarlo puesto que los datos se organizan de forma que se puede hacer uso de ellos fácilmente.

Para distinguir el primer elemento del arreglo se usará el subíndice 0. El segundo elemento se usará el subíndice 1, y así sucesivamente. El último elemento siempre estará en la posición especificada por el tamaño del arreglo menos uno. Para un arreglo de tamaño n, los elementos son:

Elementos de un arreglo
dato [0]
dato [1]
dato [2]
. . .
dato [n – 2]
dato [n – 1]

Suponga que el valor de n es 5 y que los valores son:

0	1	2	3	4
100	200	300	400	500

Una forma de asignar los valores a las diferentes posiciones de un arreglo es usando el acceso directo.

Acceso directo a las posiciones de un arreglo

```
int dato[ ];
dato = new int [ 5 ];
dato [ 0 ] = 100;
dato [ 1 ] = 200;
dato [ 2 ] = 300;
dato [ 3 ] = 400;
dato [ 4 ] = 500;
```

También puede ser útil emplear una variable entera como subíndice que sirva para acceder a las posiciones del arreglo. Por ejemplo, si se desea asignar 2000 en la posición 3 usando un

subíndice, puede utilizar el acceso indirecto.

Acceso indirecto a las posiciones de un arreglo

`int dato[ ], i;`
`dato = new int [ 5 ];`
`i = 3;`
`dato [ i ] = 2000;`

Los valores almacenados en las posiciones de un arreglo son variables. Por lo tanto, ahora el valor de dato [3] es 2000 y no 400 como estaba inicialmente.

0	1	2	3	4
100	200	300	2000	500

Es muy práctico utilizar ciclos con el fin de acceder a todas las posiciones del arreglo, ya sea para leer, asignar, manipular o imprimir los valores que se encuentran almacenados en las diferentes posiciones.

Acceso indirecto a las posiciones de un arreglo	

`int dato [ ], i;`
`dato = new int [ 5 ];`
 	
`i = 0;`
`do`
`{`
`  dato [ i ] = 0;`
`  i++;`
`}`
`while ( i < 5 );`
	`int dato [ ], i;`
	`dato = new int [ 5 ];`
	
	`for ( i = 0; i < 5; i++ )`
	`{`
	`  dato [ i ] = 0;`
	`}`

A pesar de que en el ejemplo anterior no fue considerado el ciclo *while*, esta clase de ciclo también puede ser utilizada para este fin. No obstante, generalmente se utiliza el ciclo *for*, por facilidad en la programación.

Java también permite, usar una sintaxis abreviada para los arreglos en la cual se declara y reserva; lo anterior es posible inicializando directamente los elementos de un arreglo. La forma de inicializar el arreglo es la siguiente:

```
Tipo_Arreglo nombre_arreglo[] = {valor1, valor2, valor3 ... valorn}
```

Donde tipo_arreglo es el tipo del arreglo, nombre_arreglo es el nombre del arreglo, y los elementos del arreglo son valor1, valor2, valor3... valorn. Se debe aclarar que cada uno de estos valores debe ser del mismo tipo del arreglo. De no ocurrir esto, el compilador de Java generará un error. En este caso, el operador *new* no es utilizado y la memoria suficiente para el arreglo se provee automáticamente.

El siguiente caso muestra cuando los elementos del arreglo se crean directamente:

```
String nombres[] = {"julio", "mario", "andrea", "pedro"};
```

A continuación se mostrará un ejemplo de aplicación básico relacionado con el manejo de los elementos fundamentales de los arreglos.

EJEMPLO DE APLICACIÓN DE ARREGLOS UNIDIMENSIONALES

A continuación se tiene un ejemplo en el cual se genera un arreglo con elementos de tipo entero y posteriormente en pantalla se muestra cada uno de sus elementos.

```
public class Ejemplo1
{
      static int arreglo [] = new int [10];

      public static void main (String args[])
      {
            generar_Arreglo();
            resultado();
```

```
	}

	// Método que genera e inicializa los valores del arreglo.
	public static void generar_Arreglo()
	{
		for (int i=0; i<10;i++)
		{
			arreglo[i] = i;
		}
	}
	// Método que muestra los elementos del arreglo.
	public static String mostrar_Arreglo()
	{
		String Resultado = "";
		for (int i=0; i<10;i++)
		{
			Resultado += arreglo[i] +"\n";
		}
		return	Resultado;
	}



	public static void resultado()
	{

		JOptionPane.showMessageDialog(null, "resultado es: " + "\n"
							+mostrar_Arreglo());
	}
}
```

8.2.4 Actividades Estructuras Contenedoras

A continuación se detallan cada una de las actividades que debe realizar para las estructuras contenedoras. Adicional a la realización de los ejercicios de programación propuestos, se debe diligenciar una serie de formatos. Este trabajo se debe realizar de forma individual.

INTRODUCCIÓN
El desarrollo de software implica la realización de un conjunto de actividades encaminadas a la construcción de productos de calidad. Una primera actividad está relacionada con la elaboración de un plan de proyecto o plan de trabajo que sirva como guía para la correcta realización de los trabajos. En su forma más básica, un plan de proyecto es un texto descriptivo del proyecto que contiene: los objetivos, el método de trabajo y el alcance. Los objetivos son los resultados que obtendremos, el método de trabajo es un texto descriptivo de las actividades que va a realizar y el alcance son las entregas.

A continuación se describen cuatro ejercicios que requieren el uso de ciclos. Para cada uno de ellos debe realizar las actividades propuestas:

ACTIVIDAD - EJERCICIO 5A
En programación es frecuente el almacenamiento del salario mensual de los empleados. Se necesita una aplicación que permita realizar diferentes tipos de operaciones sobre un arreglo unidimensional. Para este caso asuma que el arreglo contiene 4 elementos. A continuación se describen los problemas fundamentales. **Requerimientos del Programa 5A** • Los sueldos deben ser solicitados al usuario por medio de un cuadro de diálogo. • Mostrar cuántos salarios son superiores a $500.000 • Calcular el promedio de los salarios de los empleados. • Mostrar el salario más alto de los empleados. • Identificar y mostrar la posición del menor salario de los empleados. • Mostrar los salarios que se encuentran en las posiciones impares del arreglo. • Mostrar los salarios desde la última posición hasta la primera posición. • Mostrar el menor salario dentro del arreglo. • Mostrar el mensaje “mayores los pares” si la suma de los elementos pares es mayor de la suma de los elementos impares; en caso contrario, mostrar el mensaje “mayores impares”. Durante el desarrollo del trabajo debe diligenciar el formato de **registro de errores.**

Planeación: Describa la forma como va a planear la realización del ejercicio 5ª.

ACTIVIDAD - EJERCICIO 6A
En programación es frecuente el almacenamiento del número de identificación de los empleados. Se necesita una aplicación que permita realizar diferentes tipos de operaciones sobre un arreglo unidimensional. Para este caso asuma que el arreglo contiene 4 elementos. A continuación se describen los problemas fundamentales. **Requerimientos del Programa 6A** • Los nombres deben ser solicitados al usuario por medio de un cuadro de diálogo. • Cuántos números de identificación de los empleados comienzan con el número 2. • Cuántos números de identificación son pares. • Muestre la posición de los elementos cuyo número de identificación termina en 8. • Cuál es el promedio de los números de identificación. Durante el desarrollo del trabajo debe diligenciar el formato de **registro de errores.** **Planeación:** Describa la forma como va a planear la realización del ejercicio 6ª.

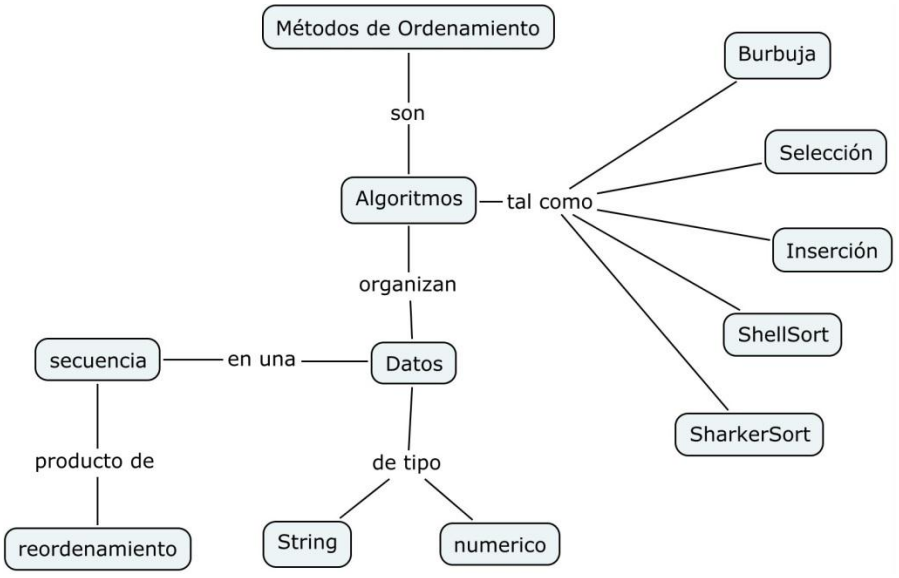
8.2.5 Guía Métodos de Ordenamiento

A continuación se presenta una guía de aprendizaje para las estructuras contenedoras.

GUIA MÉTODOS DE ORDENAMIENTO	
Nombre de la Actividad: Métodos de Ordenamiento	Duración en horas
Resultados de Aprendizaje • Identificar los conceptos fundamentales de los métodos de ordenamiento. • Aplicar los conceptos de ciclos y estructuras contenedoras al concepto de ordenamiento. • Resolver problemas en los cuales se aplique el ordenamiento de	8 horas

elementos de diferente tipo • Conocer y entender los diversos métodos de ordenamiento.	

MOTIVACIÓN
El objetivo de los algoritmos de ordenamiento es organizar una secuencia de datos, sea cual sea su tipo. Este tema ha sido ampliamente estudiado y aplicado en diferentes contextos de las ciencias de la computación. Se analizarán los algoritmos de ordenamiento considerados como clásicos. Algunos ejemplos de aplicación de estos algoritmos, pueden ser: • Lista ordenada por apellido de estudiantes del programa de Ingeniería de Sistemas. • Un directorio telefónico de usuarios. • Un reporte de las ventas totales para cada uno de los meses del año.

INTRODUCCIÓN A LOS MÉTODOS DE ORDENAMIENTO
El siguiente mapa conceptual define los elementos fundamentales, relacionados con los conceptos de métodos de ordenamiento. Ver figura 29.    Figura 29. Mapa conceptual Métodos de Ordenamiento.

Para este tema se analizarán todos los métodos de ordenamiento relacionados anteriormente.

A continuación se muestran algunos algoritmos de ordenamiento:

ORDENAMIENTO POR BURBUJA

Este método consiste en la comparación por parejas adyacentes e intercambiarlas, de acuerdo con el orden que desee darse. Este proceso se repite hasta cuando el conjunto de datos está completamente ordenado. Se llama burbuja, pues consiste en flotar hasta el final del arreglo el elemento mayor en cada iteración.

```
public void burbuja( int arreglo[] ){
     int temp, j, i;
     for( j = 1 ; j < arreglo.length ; j++) {
          for( i = 0 ; i<arreglo.length-1 ; i++){
               if ( arreglo[ i ] > arreglo[i+1] ){
                    temp = arreglo[ i ];
                     arreglo[i] = arreglo[i + 1];
                     arreglo[i+1] = temp;
               }
          }
     }
}
```

ORDENAMIENTO POR INSERCIÓN

Este método consiste en tomar cada uno de los elementos e insertarlos en una sección del conjunto de datos que ya se encuentra ordenado. El conjunto de datos que ya se encuentra ordenado se inicia con el primer elemento del mismo. Este método comúnmente se asocia con la forma de organizar un juego de cartas.

```
public void insercion( int arreglo[] )
{
     int i, llave;
     for ( int j = 1 ; j < arreglo.length ; j++ )
     {
          llave = arreglo[ j ];
          i = j - 1;
          while( i >= 0 && arreglo[ i ] > llave ){
               arreglo[i + 1] = arreglo[ i ];
```

```
                    i--;
                }
                arreglo[i + 1] = llave;
            }
        }
```

ORDENAMIENTO POR *SHARKERSORT*

Conocido como el método de la “Sacudida”, es una mejora del método de la burbuja. El funcionamiento de este algoritmo consiste en mezclar las dos formas en que se puede realizar el método de burbuja. En este algoritmo cada pasada tiene dos etapas. En la primera etapa, “de arriba hacia abajo”, se trasladan los elementos más pequeños hacia la parte de arriba del arreglo, almacenando en una variable la posición del último elemento intercambiado. En la segunda etapa, “de abajo hacia arriba”, se trasladan los elementos más grandes hacia la parte de abajo del arreglo, almacenando en otra variable la posición del último elemento intercambiado. Las pasadas sucesivas trabajan con los componentes del arreglo comprendidos entre las posiciones almacenadas en las variables. El algoritmo termina cuando la variable que almacena el extremo de arriba del arreglo es mayor que el contenido de la variable que almacena el extremo de abajo.

```
public void shakerSort( int x[ ] ){
    int aux, primero = 1;
    int ultimo = x.length - 1, dir = x.length - 1;

    while ( ultimo >= primero )
    {
        for( int i = ultimo ; i >= primero ; i--){
            if ( x [i - 1] > x[ i ] ) {
                aux = x[i - 1];
                x[i - 1] = x[ i ];
                x[ i ] = aux;
                dir = i;
            }
        }
        primero = dir + 1;
        for( int i = primero ; i <= ultimo; i++ ){
```

```
                    if ( x [i - 1] > x[ i ] ){
                          aux = x[i - 1];
                          x[i - 1] = x[ i ];
                          x[ i ] = aux;
                          dir = i;
                    }
               }
               ultimo = dir - 1;
          }
     }
```

ORDENAMIENTO POR SELECCIÓN

Este método de selección tiene este nombre puesto que la manera de realizar el ordenamiento es seleccionando en cada iteración el menor elemento del arreglo e intercambiarlo en la posición correspondiente. Inicialmente se encuentra el menor elemento dentro del arreglo y se intercambia con el primer elemento del arreglo. Luego, desde la segunda posición del arreglo se busca el menor elemento y se intercambia con el segundo elemento del arreglo.

Así sucesivamente, se busca cada elemento de acuerdo con su índice i y se intercambia con el elemento que tiene el índice k. Se ordenarán los elementos del arreglo de forma ascendente.

```
public void seleccion (int arreglo[] ){
     int i, j, k, menor;
     i = 0;
     while( i < arreglo.length - 1)
     {
          menor = arreglo [i];
          k = i;
          for( j = i+1; j < arreglo.length; j++)
          {
               if (arreglo [j] < menor ){
                    menor = arreglo [j];
                    k = j;
```

```
                        }
                    }
                    arreglo [k] = arreglo[i];
                    arreglo [i] = menor;
                    i++;
                }
            }
```

ORDENAMIENTO POR *SHELLSORT*

El método de ordenamiento *Shell* es una versión mejorada del método de Inserción directa. Recibe su nombre en honor a su autor, Donald L. Shell, quien lo propuso en 1959. El objetivo de este método es ordenar un arreglo formado por n enteros. Inicialmente ordena subgrupos de elementos separados k unidades (respecto de su posición en el arreglo) del arreglo original. El valor k es llamado incremento (Weiis, 2002).

Después que los primeros k subgrupos han sido ordenados, se escoge un nuevo valor de k más pequeño, y el arreglo es de nuevo partido entre el nuevo conjunto de subgrupos. Cada uno de los subgrupos mayores es ordenado y el proceso se repite de nuevo con un valor más pequeño de k. Eventualmente, el valor de k llega a ser 1, de tal manera que el subgrupo consiste de todo el arreglo ya casi ordenado.

```
public void shellSort(int a[])
{
    for (int incr = a.length/2; incr>0; incr/= 2 )
    {
          for (int i = incr ; i < a.length ; i++ )
          {
                int j = i - incr;


                while (j >= 0)
                {
                     if (a[j] > a[j + incr]){
                           int T = a[ j ];
                           a[ j ] = a[j+incr];
```

```
                                    a[j+incr] = T;
                                    j -= incr;
                                }
                                Else {
                                    j = -1;
                                }
                            }
                        }
                    }
                }
```

8.2.6 Herramienta de apoyo a la estrategia de aprendizaje

Teniendo en cuenta que la estrategia de aprendizaje plantea dentro de algunos de sus escenarios el desarrollo de procesos de formación autónomo en los estudiantes, se plantea como elemento pedagógico de soporte adicional el uso de una ambiente virtual de aprendizaje.

Para (Brocket & Hiemstra, 1993), la autodirección en el aprendizaje es una combinación de fuerzas tanto interiores como exteriores de la persona, que subrayan la aceptación por parte del estudiante de una responsabilidad cada vez mayor respecto a las decisiones asociadas con el proceso de aprendizaje.

Como instrumento de desarrollo del aprendizaje autónomo y como elemento de soporte al proceso de aprendizaje de los estudiantes, se utilizó una plataforma tecnológica LMS (*Learning Management System*), herramienta que permite la gestión de recursos pedagógicos con los cuales se pueden soportar los escenarios de aprendizaje diseñados.

El LMS que sirvió de apoyó para el curso fue el MOODLE (*Modular Object Oriented Dynamic Learning Enviroment*), según (Universidad del Quindío, 2010) su estructura posibilita el diseño y programación de los cursos, una vez se tienen definidos todos los elementos, recursos, mediaciones, cronograma, interacciones e interactividades. También ofrece una serie flexible de actividades para los cursos: foros, cuestionarios, consultas, encuestas, tareas, chat y talleres.

Para la implementación de la estrategia de aprendizaje se utilizaron fundamentalmente los foros, cuestionarios, las encuestas, tareas y talleres. Como recursos de soporte, se crearon

presentaciones con contenido digital, de acuerdo con las necesidades de aprendizaje de los estudiantes. A continuación se presenta una breve descripción de los recursos más utilizados.

- **Cuestionario:** Actividad que permite diseñar y proponer exámenes o pruebas, compuestos de preguntas de opción múltiple, verdadero/falso, y preguntas con respuestas cortas. También permite utilidades de calificación.

- **Foro:** Actividad que permite la interacción de los participantes ante algún tema en particular; se propone una discusión con base en un tema planteado por el profesor o por el estudiante.

- **Tarea:** Con esta actividad se establece un trabajo para los estudiantes con una fecha de entrega y una calificación. Puede usarse para informes, talleres, ensayos, proyectos, consultas, avances de investigación, recolección de datos, imágenes, entre otros.

Teniendo clara la estructura del curso, es importante presentar a continuación la estructura en Módulos, subtemas y disposición de contenidos, actividades e instrumentos de evaluación con que los estudiantes tuvieron su interacción.

La herramienta de apoyo a la estrategia de enseñanza se encuentra disponible en la plataforma virtual de la Universidad del Quindío. Dentro del campus virtual se encuentra alojado el curso denominado Paradigma Orientado a Objetos; la dirección del curso es: http://uv.uniquindio.edu.co/moodle/course/view.php?id=273 .

La presentación del curso contiene los datos de cada uno de los participantes, cada una de las unidades del curso con sus respectivos recursos. Se presenta un video motivacional sobre la importancia del trabajo del curso. Dentro de la ambientación, los estudiantes deben participar en el foro introductorio en el cual deben realizar una presentación de sus intereses particulares y así mismo de reconocimiento de sus intereses de formación.

En la figura 30, se muestra la página inicial del curso Paradigma Orientado a Objetos, en el cual se muestra la estructura temática del mismo.

Figura 30. Presentación Ambiente Virtual de Aprendizaje. (Cardona & Rincón, Ambiente virtual de aprendizaje para la implementación de prácticas de PSP y TSP en un curso de programación de computadores, 2012).

Para cada una de las unidades se presentan los objetivos de formación, así como los recursos disponibles y que sirven como soporte a la estrategia de enseñanza. La figura 31 muestra un esquema general que se aplica a cada una de las unidades diseñadas.

Figura 31. Contenido temático del Ambiente Virtual de Aprendizaje. (Cardona & Rincón, Ambiente virtual de aprendizaje para la implementación de prácticas de PSP y TSP en un curso de programación de computadores, 2012)

8.3 Recurso Humano

Para el soporte de esta propuesta metodológica alrededor de todos los cursos, se hace necesario contar con un recurso humano comprometido con la implementación de la misma. Para ello se propone la siguiente estructura y la cual esta en correspondencia con la normatividad de la Universidad del Quindío, específicamente en lo definido en el acuerdo 014 de 2006 del Consejo Superior. Sin embargo esta propuesta se plantea de la forma más genérica para ser extensible a otras instituciones de educación superior. En la tabla 35 se muestra esta propuesta.

Dentro del acuerdo se establecen actividades orgánicas complementarias forman parte de la asignación académica del profesor. Tales actividades son, entre otras, las siguientes: Capacitación de docentes que no implique comisiones de estudio, asistencia a los claustros de profesores del programa o la facultad, participación en actividades extracurriculares (representaciones institucionales y otras), participación en consejos y/o comités institucionales, permanentes o ad-hoc, evaluación de la producción intelectual de los profesores, fortalecimiento de la modalidad a distancia, asesorías de proyectos de grado de los estudiantes, pasantías y procesos de

autoevaluación, los cuales serán aprobadas por los Consejos de Facultad (Universidad del Quindío, 2006).

Tabla 35. Presupuesto de recurso humano.

Asignatura	Recurso Humano	Dedicación semanal
Paradigma Orientado a Objetos (3 créditos)	• 2 Profesores: Ingenieros de Sistemas y/o afines, con formación a nivel de posgrado con énfasis en informática o ingeniería de software. • 2 Estudiantes Monitores, con experiencia en cursos de programación y algoritmia.	Profesores: • 4 Horas Docencia Directa • 4 Horas de Seguimiento • 3 Preparación de sesiones Estudiantes: • 4 horas de asesoría
Fundamentos de Algoritmia (3 créditos)	• 2 Profesores: Ingenieros de Sistemas y/o afines, con formación a nivel de posgrado con énfasis en informática o ingeniería de software. • 2 Estudiantes Monitores, con experiencia en cursos de programación y algoritmia.	Profesores: • 4 Horas Docencia Directa • 4 Horas de Seguimiento • 3 Preparación de sesiones Monitores: • 4 horas de asesoría
Lenguaje de Programación (3 créditos)	• 2 Profesores: Ingenieros de Sistemas y/o afines, con formación a nivel de posgrado con énfasis en informática o ingeniería de software. • 2 Estudiantes Monitores, con experiencia en cursos de programación y algoritmia.	Profesores: • 4 Horas Docencia Directa • 4 Horas de Seguimiento • 3 Preparación de sesiones Monitores: • 4 horas de asesoría
Estructuras de Datos (3 créditos)	• 1 Profesor: Ingenieros de Sistemas y/o afín, con formación a nivel de posgrado con énfasis en informática o ingeniería de software. • 1 Estudiante Monitor, con experiencia en cursos de	Profesores: • 4 Horas Docencia Directa • 4 Horas de Seguimiento • 3 Preparación de sesiones

	programación y algoritmia.	Monitores: • 4 horas de asesoría
Análisis de Algoritmos I (3 créditos)	• 1 Profesor: Ingenieros de Sistemas y/o afín, con formación a nivel de posgrado con énfasis en informática o ingeniería de software. • 1 Estudiante Monitor, con experiencia en cursos de programación y algoritmia.	Profesores: • 4 Horas Docencia Directa • 4 Horas de Seguimiento • 3 Preparación de sesiones Monitores: • 4 horas de asesoría

Se debe contar con un coordinador del área que administre y controle las actividades tanto de los profesores como de los monitores. Los estudiantes deben aplicar el sistema de créditos de la misma forma para todas las asignaturas.

A continuación se presentan asignaturas en las cuales es posible la incorporación de practicas de PSP y TSP durante el proceso de formación de los estudiantes. Ver figura 32.

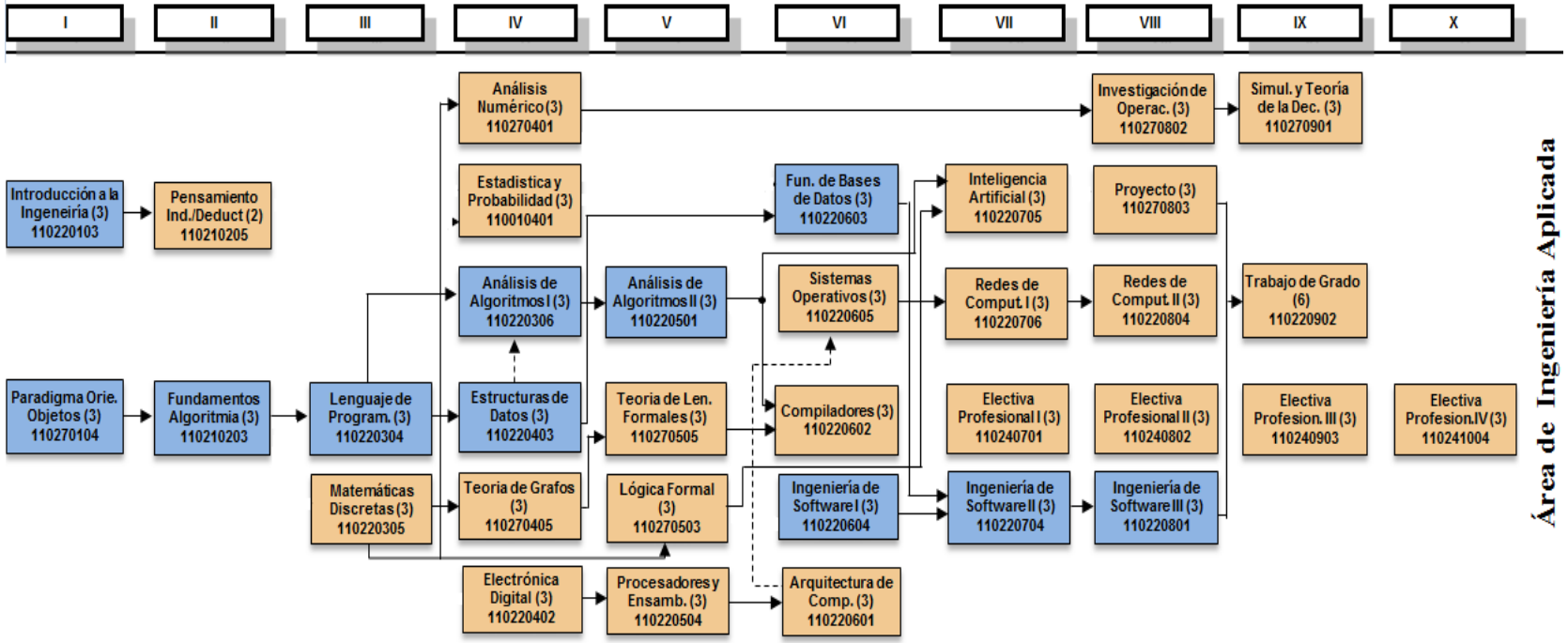


Figura 32. Propuesta de Intervención de PSP y TSP en el plan de estudios.

La integración de este trabajo con los demás cursos de plan de estudio, esta definida de forma incremental tanto a nivel vertical como a nivel horizontal. Las asignaturas seleccionadas (color azul), están ubicadas dentro del área de la Ingeniería aplicada y dadas sus características, permitirían la implementación de un conjunto de prácticas de PSP y TSP. Este tema daría la oportunidad de darle continuidad al desarrollo de este proyecto.

A pesar que en el plan de estudio también se tienen consideradas las área de la Ciencia básica y humanística, estas no fueron tenidas en cuenta debido a sus objetivos de formación y a la administración que se realiza de ellas dentro del currículo.

8.4 Evaluación del curso piloto

La intervención en el curso se realizó de forma incremental; cada semana en una clase teórica se orientaban los conceptos fundamentales del proceso de desarrollo personal y en equipo, además de los formularios diseñados para la realización de las actividades prácticas propuestas. Se proporcionó una guía de las temáticas definidas y una serie de actividades con ejercicios prácticos.

Como los estudiantes son de primer semestre, tenían un nivel de conocimientos muy básico en la resolución de problemas utilizando un lenguaje de programación. Considerando que dentro de las acciones que debe realizar el profesor es necesario realizar mediciones constantes, la recopilación de datos debe ser tomada de forma rigurosa y es posible que se incluyan errores de toma de datos. Para ello se han diseñado formatos que faciliten esta actividad.

Para la gestión de los materiales de aprendizaje de los cursos, publicación de información, trabajo en grupo, publicaciones en general y entrega de las tareas, se utilizó la plataforma virtual Moodle de la Universidad del Quindío.

La intervención realizada en el grupo experimental muestra que cada una de las actividades propuestas implica a los estudiantes un esfuerzo adicional, pues los mismos deben diligencias los formatos establecidos y adicionalmente realizar unas entregas de sus tareas más formalmente. Ante esta situación se sugiere que los estudiantes de primer semestre apliquen las prácticas más básicas de PSP. En la medida en que se van conociendo, entendiendo y aplicando las prácticas propuestas en PSP y TSP, los estudiantes pueden, de acuerdo con su interés personal, aplicar las mejores prácticas para el desarrollo de sus proyectos.

Una de las dificultades más significativas durante la intervención con los estudiantes radica en el diligenciamiento de los formatos, muchos de ellos no los completan totalmente y se encuentra que no son diligenciados correctamente.

El instrumento aplicado a los estudiantes para validar los conocimientos posteriores pretendía identificar los conocimientos y las actitudes de los estudiantes frente al PSP y al TSP. Se encuentra que muchos estudiantes no identifican y no reconocen los beneficios de estos procesos, y consideraron que algunas de estas actividades implicaron tiempo adicional para aprender aspectos relacionados con el lenguaje de programación. Se dio como alternativa a los estudiantes presentar electrónicamente los archivos de registro de tiempo y de errores. Los estudiantes utilizaron la

plataforma Moodle para subir esta información; esto permite al profesor una administración mucho más fácil y facilita el seguimiento de las entregas de los estudiantes. Un problema importante es que se ha encontrado que un buen número de estudiantes no completan los formularios correctamente.

8.5 Experiencias Vividas con los estudiantes

El trabajo con los estudiantes del grupo experimental se llevó a cabo a través de diferentes etapas: la etapa inicial consistió en la apropiación de los conceptos definidos para la estrategia; se proporcionó el material y se trabajaron los conceptos de Ciclos, Arreglos y Ordenamiento, en los cuales se observó que los estudiantes asimilaron adecuadamente los conceptos y contestaron con facilidad y rapidez las preguntas dirigidas por parte del profesor. Se debe resaltar que al inicio los estudiantes no comprendían claramente el objetivo de las actividades, por lo tanto el tema de Ciclos resultó complejo en cuanto a su comprensión. En la segunda fase se trabajó desde la perspectiva de la solución de problemas, se plantearon casos concretos donde las actividades propuestas eras de carácter individual, en estas actividades se evidenció mucha motivación y participación por parte de los estudiantes del grupo experimental.

Los estudiantes del grupo intervenido manifestaron motivación frente a las actividades realizadas gracias a la metodología utilizada (trabajo motivacional en cada una de las etapas) y se observó que a través del trabajo comprendían y se asimilaban los temas, lográndose un mayor nivel de conocimiento. Con este tipo de actividades se evidenció la necesidad y la importancia que tienen las estrategias didácticas en el proceso de enseñanza – aprendizaje de las prácticas de PSP y TSP, resultando una experiencia pedagógica significativa. Teniendo en cuenta que determinadas actividades y el proyecto final debieron ser realizadas en equipos, se experimentaron los siguientes problemas durante el desarrollo del proyecto:

- Se observan problemas de liderazgo debido a que todos los estudiantes no muestran la suficiencia necesaria al momento de argumentar la solución propuesta.
- A pesar de estar debidamente asignados los roles, muestra que es difícil establecer un compromiso de cada uno de los integrantes del equipo y se dificulta su trabajo cooperativo.
- En cuanto a la planeación del proyecto se evidenció en la entrega del proyecto final que los estudiantes no realizaron seguimiento al proyecto así como tampoco un esquema de trabajo claramente definido.

9 ANÁLISIS DE RESULTADOS

Con base en la recolección de información en los grupos control y experimental que hicieron parte de la muestra para validar las preguntas de investigación formuladas, este capítulo se enfoca en la descripción de los resultados obtenidos, producto de la implementación de la estrategia de aprendizaje para implementar las prácticas de PSP y TSP en un curso básico de Programación de computadoras.

9.1 Resultados

Inicialmente se analizará si para cada una de las preguntas del instrumento continúa existiendo homogeneidad entre los grupos. En primera instancia se afirma que si la estrategia de enseñanza planteada tuvo un efecto positivo, debe existir ya diferencia entre los grupos de control y experimental.

Con el objetivo de verificar si el grupo control y el grupo experimental son homogéneos para cada una de las preguntas del instrumento, se corrió un Análisis de Varianza cuya variable de respuesta es la calificación en cada pregunta y el factor es el grupo con los niveles control y experimental; se realiza un análisis de varianza en el cual se verifican los supuestos de aleatoriedad, la homogeneidad de las varianzas y que los residuos se ajusten a una distribución Normal. Cuando alguno de estos supuestos no se cumpla, se acude a la prueba No paramétrica de Kruskal y Wallis.

A continuación se muestran los resultados de los grupos control y experimental para el *postest*. En la figura 33 se muestran los resultados asociados al control de tiempos, en el cual se reporta que existe una diferencia estadísticamente significativo y por lo tanto los grupos no son homogéneos.

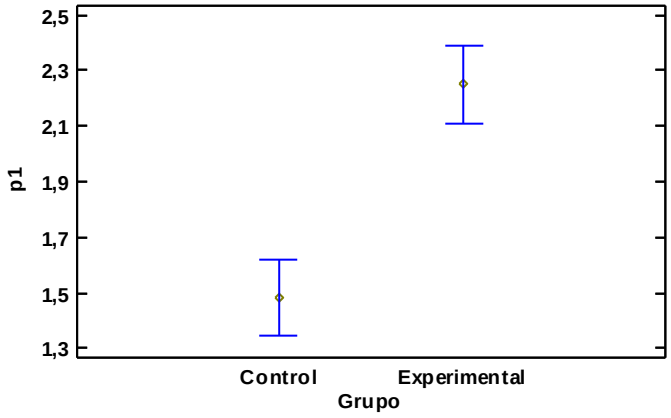


Figura 33. Resultado *postest* para la práctica de control de tiempos

En relación a la práctica asociada con el registro de interrupciones, el análisis realizado muestra que existe diferencias estadísticamente significativas en el *postest*, tanto para el grupo experimental como para el grupo de control. En la figura 34, se muestran los resultados para ambos grupos.

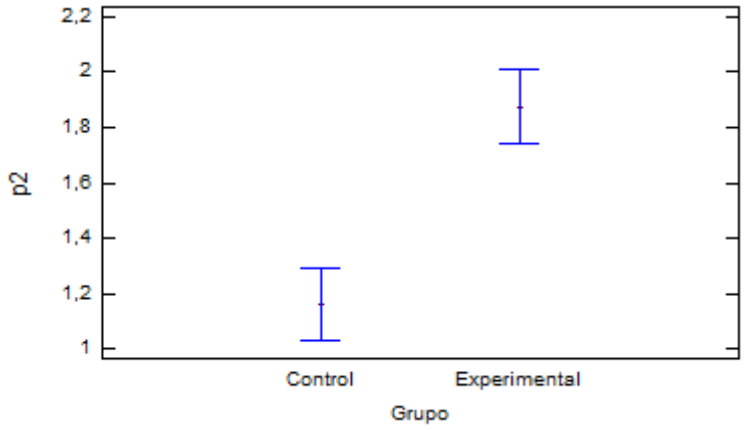


Figura 34. Resultados postest para la práctica registro de interrupciones.

Para la práctica de registro de errores, el análisis realizado muestra que no existen diferencias estadísticamente significativas entre ambos grupos y por lo tanto siguen conservando la propiedad de homogeneidad. En la figura 35 se muestran los resultados asociados a ambos grupos.

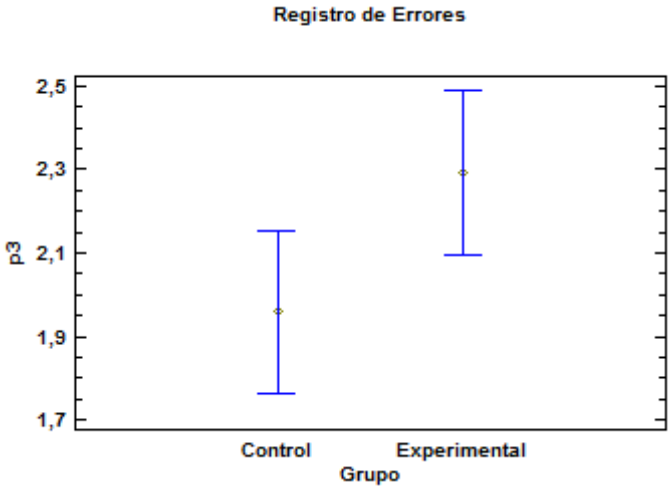


Figura 35. Resultados postest para la práctica de registro de errores.

Con relación a los errores de codificación, el análisis realizado muestra que existe diferencias estadísticamente significativas en el *postest* y por lo tanto se puede afirmar que los grupos son homogéneos. En la figura 36, se muestran los resultados para ambos grupos.

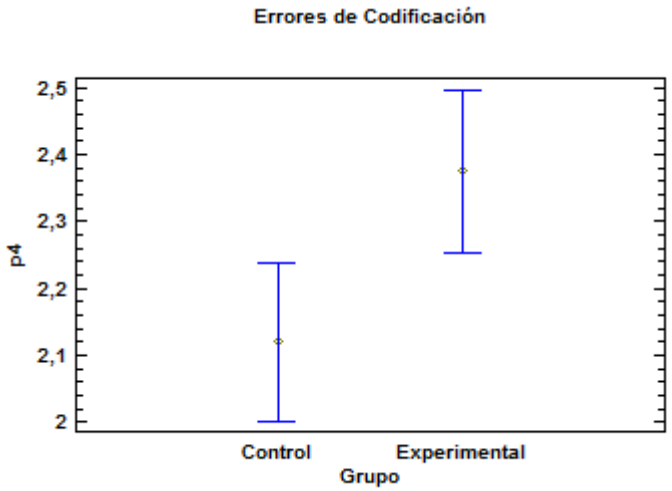


Figura 36. Resultados de postest para los errores de codificación

En cuanto a la estrategia de solución de errores de codificación, los resultados del análisis muestran que existe diferencia estadísticamente significativa entre los grupos de control y experimental. En la figura 37, se muestran los resultados para ambos grupos.

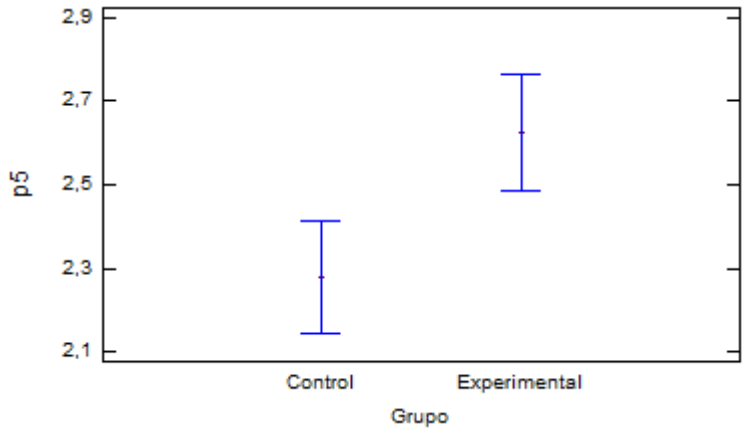


Figura 37. Resultados de *postest* para estrategia de solución de errores de codificación.

Para la práctica de uso de estándares de codificación, el análisis realizado muestra que no existen diferencias estadísticamente significativas entre ambos grupos y por lo tanto siguen conservando la propiedad de homogeneidad. En la figura 38 se muestran los resultados asociados a ambos grupos.

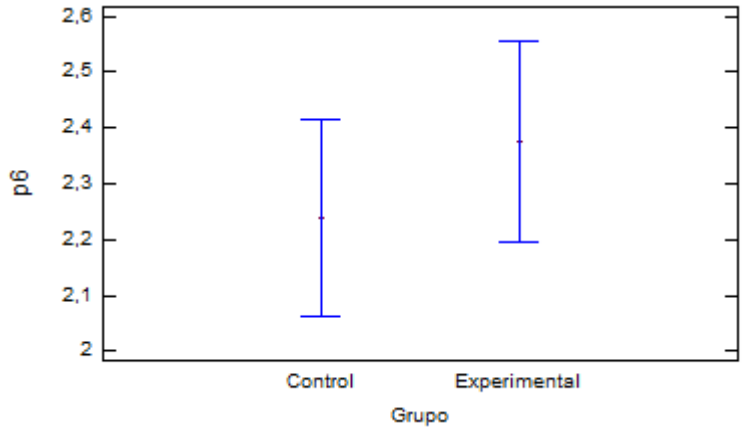


Figura 38. Resultado de *postest* para uso de estándares de codificación.

En cuanto a la estrategia de planificación individual, los resultados del análisis muestran que no existe diferencia estadísticamente significativa entre los grupos de control y experimental. En la figura 39, se muestran los resultados para ambos grupos.

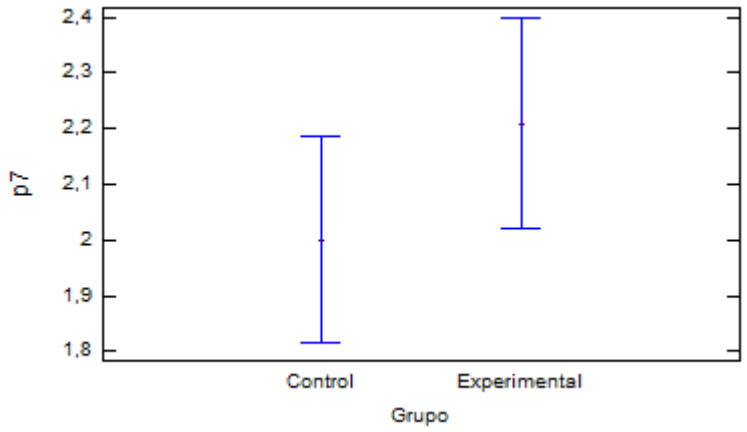


Figura 39. Resultado de *postest* para planificación individual

Para la práctica de planificación individual, los resultados muestran que no existe diferencia estadísticamente significativa entre los grupos de control y experimental. En la figura 40, se muestran los resultados para ambos grupos.

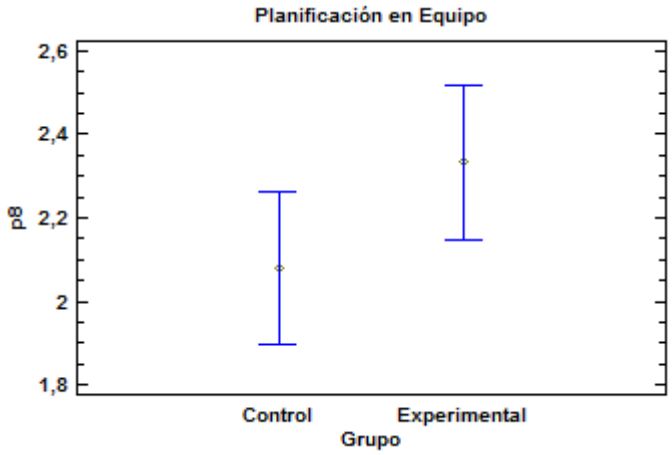


Figura 40. Resultados de *postest* para planificación en equipo

Para la práctica de planificación individual, los resultados muestran que no existe diferencia estadísticamente significativa entre los grupos de control y experimental. En la figura 41, se muestran los resultados para ambos grupos.

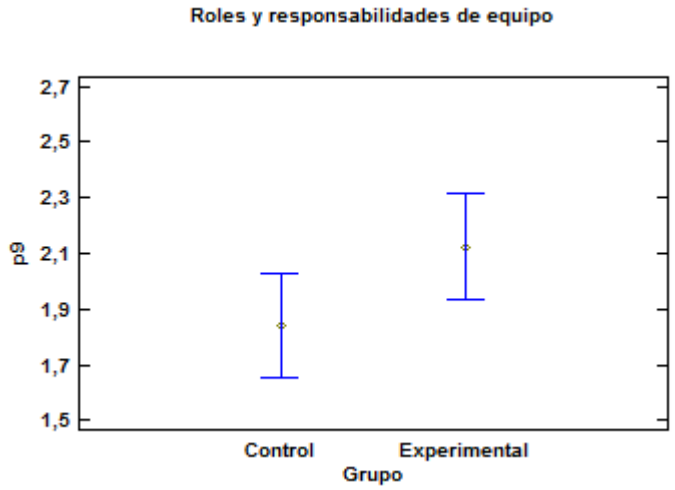


Figura 41. Resultado Postest para roles y responsabilidades de equipo.

A continuación en la tabla 36 se muestran los resultados numéricos del postest aplicado a ambos grupos.

Tabla36. Resultados numéricos postest

Pregunta	p_valor	Homogeneidad de varianzas (levens)	Distribución normal de residuos Shapiro-Wilk	Prueba No paramétrica Kruskay y Wallis
1	0,00000	0,00745645	1,76441E-8	0,00000928691
2	0,00000	0,27902	2,52995E-7	0,00001062
3	0,0958	0,486993	0,000573128	0,0914701
4	0,0386	0,0386222	5,99462E-11	0,0399977
5	0,0147	0,488702	2,91846E-7	0,0162911
6	0,4514	0,88233	1,142E-7	0,500208
7	0,2689	0,0595843	0,00000530401	0,311478
8	0,1717	0,492015	0,0000421398	0,164674
9	0,1329	0,121943	0,000103865	0,135523

De los anteriores resultados se puede afirmar que la intervención en el grupo experimental fue exitosa en 5 de las 9 preguntas. Las cuales con relación a las categorías definidas en la sección 4.2, muestra que para los aspectos relacionados con la administración y gestión del tiempo, el manejo y gestión de defectos y el manejo del tamaño del producto, la intervención mediante la estrategia de enseñanza fue exitosa.

A continuación se muestra un análisis inferencial de los resultados obtenidos en la intervención al grupo experimental.

9.2 Análisis de Resultados

Para probar el impacto de la intervención, se utilizó una prueba de hipótesis, que permite probar, en cada una de las preguntas, si al menos la mitad del grupo intervenido mejoró la aplicación de las buenas prácticas de desarrollo de software.

Se compararon las respuestas del pretest y del postest de los estudiantes intervenidos para construir una variable “resultado”, de la siguiente manera: si la nota del postest es mayor que la del pretest, la variable toma el valor de 1; si la nota es menor o igual en el postest, la variable toma el valor de 0; si en el pretest y en el postest calificaron la respuesta con (3) siempre, la variable toma el valor de 1. De esta manera la variable “resultado tiene sólo dos posibles valores 1 y 0, por lo tanto es una variable discreta con distribución Bernoulli, y p=0,5 porque se está usando el criterio de que al menos el 50% de los estudiantes mejoren del pretest al postest. Para el análisis final se

descartaron los estudiantes que no habían presentado el pretest, y aquellos estudiantes que no asistieron al 100% de la estrategia de enseñanza.

Luego se sumaron las respuestas de valor 1 y se obtuvo la variable "número de estudiantes que mejoraron con la intervención", la cual por ser suma de variables con distribución Bernoulli, es una variable con distribución Binomial, con n=23, que es el número de estudiantes del grupo experimental, y la probabilidad p=0,5 indica que al menos la mitad de los estudiantes mejora con la intervención. Luego se planteó un sistema de hipótesis que permitió elegir aquellas preguntas en las que los estudiantes mejoran sus prácticas. Se establece para el experimento una probabilidad de error del 4,7% afirmar que la pregunta en la intervención fue exitosa, esto también es equivalente a afirmar que los resultados obtenidos tienen un nivel de confianza del 95,7 %.

La figura 42 muestra la distribución Binomial para los 23 estudiantes del grupo experimental, con p=0,5. En aquellas preguntas donde 16 estudiantes o más, mejoran con la intervención, son las que permiten afirmar que la misma fue exitosa.

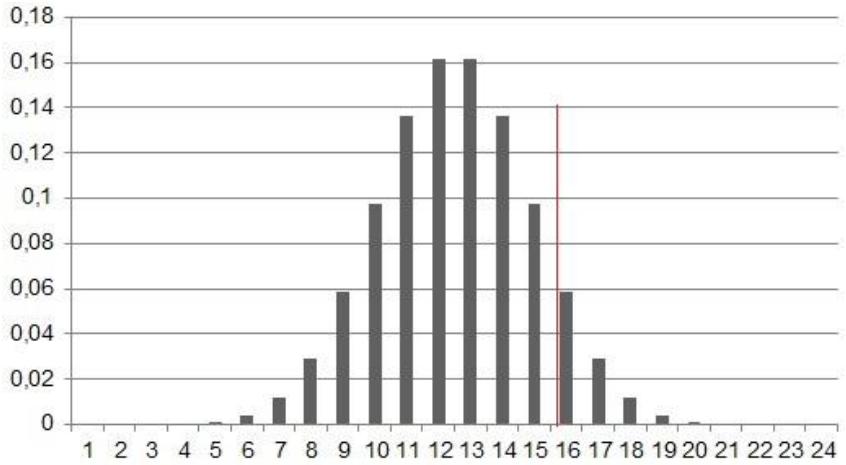


Figura 42. Distribución Binomial para postest.

Con base en lo anterior, se puede afirmar que la intervención a los estudiantes mediante la estrategia de enseñanza, en 5 de las 9 preguntas formuladas la intervención fue exitosa. A continuación se detallan los aspectos más importantes de cada una de las preguntas.

De la pregunta 1, la cual esta relacionada con el registro del tiempo se puede afirmar que los estudiantes registran de forma disciplinada el tiempo invertido en la realización de una tarea o trabajo de programación.

Con relación a la pregunta 2 relacionada con el registro de las interrupciones durante la realización de un proyecto, se encuentra que los estudiantes identifican los factores de interrupción durante la actividad, adicionalmente registraron de una forma ordenada el tiempo de las interrupciones, finalmente hicieron una aproximación al concepto estimación mediante el calculo de tiempo a invertir cuando van a realizar un proyecto. De lo anterior se puede también afirmar que los estudiantes identifican las fases de su proceso de desarrollo personal y que identifican en cuales de estas fases invierten la mayor cantidad de su tiempo. Con base en el análisis de los proyectos se observa que en todos los casos, los equipos de trabajo exceden el tiempo establecido para realizar un proyecto.

Para la pregunta 3, relacionada con el manejo y gestión de los defectos, los estudiantes de acuerdo a la plantilla diseñada para tal fin, documentaron sus errores más frecuentes, lo que implicó necesariamente un análisis de esta situación dentro de su proceso de desarrollo.

Con relación a la pregunta 4, los estudiantes comprenden los defectos introducidos al realizar un proyecto, adicionalmente los documentan para tener una línea base con relación a este ítem y que les sirva para sus proyectos futuros,

Para la pregunta 5, los estudiantes aplicaron algún método para encontrar y corregir defectos cuando se realiza un proyecto, muchos también documentaron cuanto tardaban en encontrar y corregir cada defecto al realizar un proyecto.

En cuanto a la pregunta 6 relacionada con el uso de estándares de codificación, se encuentra que en esta práctica a pesar que estaba definida una plantilla de trabajo, no fue aplicada por los estudiantes y de allí que en el resultado no se evidenciara mejora en el grupo experimental.

De la pregunta 7, se puede afirmar que los estudiantes no identifican las acciones que realizan otros para que sus proyectos funcionen mejor, así como tampoco realizan diseños preliminares para la realización de sus proyectos.

La pregunta 8 y la pregunta que estaban asociadas a la organización y trabajo en equipo, mostró que pocas veces se toman decisiones concertadas en equipo cuando se realiza un proyecto, no se definen los roles y responsabilidades de cada miembro del equipo para realizar un proyecto y

tampoco se registran y analizan en grupo los defectos encontrados al momento de construir el producto.

10 CONCLUSIONES

En este capítulo se presentan las conclusiones sobre la realización del proyecto, las perspectivas de continuidad para implementar prácticas de PSP y TSP dentro de los planes de estudio de ingeniería de Sistemas y finalmente se muestran los productos generados durante la realización de la tesis.

10.1 Conclusiones

Para impulsar la industria Colombiana de software es necesario contar con personal competente, que esté en los niveles de los mejores del mundo y con la capacidad de utilizar en su trabajo estándares internacionales. Por ello, desde las diferentes instituciones de educación superior se debe promover el desarrollo de competencias que estén asociadas con esa dinámica de cambio que presentan las empresas de desarrollo de software.

Particularmente, PSP implica un método riguroso para la recopilación y el análisis de información y por lo tanto no es un proceso sencillo, se corre el riesgo que el estudiante no lo perciba como un elemento que le agregue valor a su formación y lo vea mas como un método que no le aporta a su proceso de formación. En algunas Universidades en las cuales se han implementado las prácticas de PSP y TSP, estas fueron entendidas como un llenado de formatos, que resultan poco motivadores e interesantes para los estudiantes.

Los estudiantes deben aplicar las mejores prácticas de desarrollo de software tanto en el trabajo individual como de equipo, en temas como: gestión, estimación de tamaño, costos y tiempos, manejo de estándares, prevención y remoción de defectos. Es necesario que en su proceso de formación asuman roles y funciones, se expongan a ambientes de trabajo similares a la industria, en donde se consideren aspectos que incorporen diferentes prácticas, como una forma de ser o un hábito normal del ingeniero.

La incorporación de PSP desde los primeros semestres permite a los estudiantes entender mediante una serie de ejercicios prácticos qué es un proceso y una disciplina para realizarlo, así como adquirir el hábito de realizar mediciones, aspecto clave no sólo del desarrollo personal sino en cualquier faceta de la vida. Además, al recolectar datos sobre cómo desarrollan sus programas, se obtiene una línea base que ayudará a planificar trabajos futuros.

El desarrollo de este trabajo mostró una serie de desafíos asociados con el aprendizaje del proceso software, el cual está asociado con la madurez de los estudiantes para que reconozcan el valor de una disciplina aplicada al proceso de software (cuestión que todavía no han experimentado en etapas tempranas) y una introspección forzada para conocer cómo se desarrolla individualmente el software, comprendiendo sus hábitos y prácticas de desarrollo para mejorarlas. También se hizo necesario tomar en consideración algunas teorías existentes sobre las estrategias de enseñanza, que para ponerlas en nuestro contexto particular, implicó la incorporación de ideas acerca de cómo se debe intervenir las prácticas actuales para que los alumnos aprendan. Se identificaron las dificultades y errores más frecuentas de los estudiantes y se les incentivó para que su trabajo reflejara una alta calidad.

El trabajo realizado en la asignatura se centró en desarrollar habilidades concretas en Ingeniería del Software que involucran el registro y estimación de tamaños y tiempos a medida que se elaboran las prácticas semanales. El curso se enfocó en el desarrollo de competencias para entender el proceso de software y no en profundizar aspectos de programación. De esta manera, los estudiantes perciben por primera vez que la tarea de desarrollar software es un proceso complejo con un conjunto de actividades de ingeniería que requiere técnicas para construir software que van más allá de sus competencias iniciales sobre codificación de programas. PSP contiene un número significativo de plantillas y formularios. Se recomienda tener una herramienta que automatice dicho proceso. Por ejemplo, *PSP Student Workbook* facilita llenar los formularios PSP e implementa un registro histórico para poder llevar a cabo las estimaciones.

La intervención al grupo experimental mediante la estrategia de enseñanza, fue exitosa en 5 de los 9 criterios considerados en el instrumento aplicado a los estudiantes. Se destaca la apropiación conceptual y practica en aspectos tales como la administración y gestión del tiempo, el manejo y la gestión de los defectos. En cuanto a las estimaciones del tamaño del producto, el trabajo individual, la planeación del proyecto y el trabajo en equipo no se obtuvieron resultados favorables en el postest.

El análisis realizado dentro del marco de referencia, permite identificar proyectos, experiencias y reportes, que permitieron identificar los aspectos mas relevantes, y así realizar un análisis de los mismos, para posteriormente se retomados y adaptados al contexto particular de la investigación realizada. Comparando los resultados obtenidos con el grupo experimental con los resultados reportados en otras Universidades a nivel mundial, se puede afirmar que existe un alto grado de coincidencia en aspectos tales como: el reporte de los tiempos y los defectos, las dificultades

identificadas durante el proceso de implementación de la estrategia de enseñanza específicamente en la aplicación de las buenas prácticas de desarrollo de software.

La investigación realizada y llevada mediante la experimentación a la practica, generó un impacto positivo, tanto para el investigados como para la población objeto de estudio, particularmente los estudiantes de primer semestre del Programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío. La experiencia del curso piloto fue positiva y se constituye en un primer aporte para darle continuidad a procesos experimentales con estudiantes del programa académico.

Teniendo en cuenta la experiencia al momento de aplicar los instrumentos a los estudiantes, es necesario realizar adecuaciones que permitan un análisis estadístico mas detallado sobre los elementos y las variables que se quieren verificar durante el desarrollo de la experimentación, de esta forma será posible realizar análisis cualitativos de los cuales se pueda obtener mayor información de los individuos objeto de estudio.

10.2 Trabajos Futuros

Como trabajo futuro se pueden establecer estrategias de aprendizaje que permitan desarrollar en los estudiantes habilidades que potencien sus prácticas individuales y en equipo, para que al momento de insertarse en el mundo productivo puedan entender, explicar y aplicar en contextos reales las mejores prácticas en los proyectos de desarrollo. Para ello es necesario elaborar el material para todos los cursos del área de Programación y Algoritmia del Plan de estudios de Ingeniería de Sistemas y Computación asociándolos con los niveles de PSP2, PSP2.1 y PSP3, así mismo la elaboración de material de enseñanza para TSP.

La construcción de una ambiente virtual de aprendizaje sobre la plataforma Moodle, aportó al desarrollo de la estrategia, una mayor capacidad de comunicación con los estudiantes y respaldó la realización de algunas actividades planteadas. Es necesario continuar con la mejora de este ambiente virtual incorporando objetos de aprendizaje más dinámicos y con un mayor nivel de interacción de los estudiantes.

Es necesario que la estrategia de enseñanza para los otros cursos de programación, este respaldada por más profesores del programa de Ingeniería de Sistemas y Computación, es necesario formalizar la propuesta de participación para que se pueda permear a todos los demás cursos de esta área.

A nivel de investigación es necesario continuar con la realización de experimentos en los cuales se analices más detalladamente los indicadores propuestos en este trabajo u otros indicadores que surjan con base a posteriores trabajo. Es necesario identificar elementos adicionales de tipo cuantitativo y de tipo cualitativo que se deseen analizar bajo este contexto.

La implementación de la estrategia mediante el trabajo colaborativo permitiría que se validara la propuesta metodología en grupos diversos y de forma remota. En lo que respecta a futuros trabajos se podría investigar acerca de modelos colaborativos que permitan experimentación con estudiantes de otras Universidades tanto a nivel nacional como a nivel internacional.

A nivel pedagógico sería posible realizar un análisis sobre los tipos de aprendizaje de los estudiantes y definir una estrategia de enseñanza personalizada de acuerdo a los interés y las características propias de los estudiantes.

10.3 Productos del Trabajo

Como resultado del trabajo investigativo realizado en esta tesis de maestría, se generaron los siguientes artículos para presentación en eventos:

Evento	Nombre	Lugar
Ponencia Internacional. XXXIII Convención Panamericana de Ingenierías.	Análisis de las prácticas de desarrollo personal de software en estudiantes de pregrado.	La Habana – Cuba Del 9 al 13 de abril de 2012
Ponencia en evento. III Congreso Nacional y II Internacional de Investigación en Nuevas Tecnologías Informáticas	Implementación del Proceso Personal Software en un primer curso de Programación de Computadores	Tunja (Boyacá - Colombia) 14 de septiembre del 2012
Ponencia en evento. IV Congreso Iberoamericano	Ambiente virtual de aprendizaje para la implementación de prácticas de PSP y	Bucaramanga – Colombia, 4 y 5 de

“Soporte del Conocimiento con la Tecnología”	TSP en un curso de programación de computadores	octubre de 2012

Como otros productos que se generaron durante la realización de la tesis se puede evidenciar:

- Construcción de un Ambiente Virtual de Aprendizaje para soportar el curso.
- Participación como ponente en el primer foro local de la Mesa TIC: “Calidad del Software”, en Armenia – Quindío en el mes de marzo de 2012
- Asistencia al Curso de PSP Fundamentals, Medellín –Colombia, Noviembre de 2011.
- Participación como Profesor del Seminario: Administración de Proyectos de Software, Escuela de Administración y Mercadotecnia del Quindío. Marzo – Abril de 2011.

11 BIBLIOGRAFIA

Alva, A., & Valdez, B. (2011). Software Industry Excellence Center. Zapopan, Jalisco, México.

Aubin, V., Blauzic, L., & Dejean, G. (2011). *El uso de técnicas de PSP para el logro de competencias.* Buenos Aires.

Austin, T. (2011). *Estrategias de Aprendizaje.* Recuperado el 2012, de http://www.lapaginadelprofe.cl/UAconcagua/formacionprofesional/estrategiasdeaprendizaje.pdf

Ausubel, D. (1976). *Psicología educativa. Un punto de vista cognoscitivo.* Mexico: Trillas.

Ausubel, D. (2002). *Adquisición y retención del conocimiento. Una perspectiva cognitiva.* España: Paidós.

Barrios, P. (1992). *Propuesta de un programa de entrenamiento a docentes en estrategias cognoscitivas para la comprención con niños de educación primaria.* Mexico: UNAM.

Bayona, S., Calvo-Manzano, J., Cuevas, G., & Tomás, S. F. (2008). Teaching Team Software Process in graduate courses to increase productivity. *IEEE*, 440-446.

Bednar, G., & Levie, M. (1999). *Construcción del conocimiento escolar.* España: Paidos.

BenP, F., W. S., H., S., K., S., M., & A.Matvya. (1997). Results of Applying the Personal Software Process. *IEEE Computer*.

Bermon, L., Fernández, A., & Sanchéz, M. (2009). Experiencias Docentes en la Aplicación del Proceso Software Personal en Primero de Grado de Ingeniería Informática. *FINTDI*, 107-114.

Berzal, F. (30 de Agosto de 2012). *Curso de programación en Java.* Obtenido de http://courseware.ikor.org/java

Biggs, J. (1988). *Learning strategies and learning styles.* New York: Plenum Press.

Biggs, J. (1994). *Approaches to learning Nature and measurement of.* Oxford: Pergamon Press.

Börstler, J., Carrington, D., & Hislop, G. (2002). Teaching PSP: Challenges. *IEEE Software*, 42-48.

Brocket, R., & Hiemstra, R. (1993). *El aprendizaje autodirigido en la educación de adultos.* Barcelona: Routledge.

Cardona, S., & Rincón, R. (2012). *Ambiente virtual de aprendizaje para la implementación de prácticas de PSP y TSP en un curso de programación de computadores.* Armenia.

Cardona, S., & Rincón, R. (2012). *Implementación del Proceso Personal de Software en un primer curso de programación de computadores.* Tunja.

Coll, C. (1988). *Sicología y Curriculum.* Barcelona: Laia.

Coll, C. (1990). *Significado y sentido del aprendizaje escolar.* Barcelona: Paidós Educador.

Coll, C., & Rochera, M. (1990). *Estructuración y organización de la enseñanza: Las secuencias de aprendizaje.* Madrid: Alianza.

Coll, C., & Solé, I. (1990). *La interacción profesor/alumno en el proceso de enseñanza aprendizaje.* Madrid: Alianza.

Curtis, R., & Reigeluth, C. (1984). The use of analogies in written text. *Instructional Siente*, 99-117.

Dansereau, D. (1985). Learning strategy research. *Relating Instruction to Research*, 209-239.

Departamento Nacional de Planeación. (1 de Junio de 2007). Agenda Interna para la Productividad y la Competitividad. Bogotá, Cundinamarca, Colombia.

Díaz, F., & Hernandez, G. (2000). *Estrategias Docentes para un Aprendizaje Significativo.* Bogotá: McGraw-Hill.

Díaz, F., & Lule, M. (1978). *Efectos de las estrategias preinstruccionales en alumnos de secundaria de diferentes niveles socieconómicos.* México: UNAM.

Dyer, J. (1984). Team research and training: A state-of-the-art review. *Human factors review*, 285–323.

Esguerra, G., & Guerrero, P. (2010). Estilos de aprendizaje y rendimiento académico. *Diversitas Perspectivas en Sicología*, 97-109.

Fedesoft. (2009). *Sector de TI en Colombia.* Medellín.

Fedesoft. (13 de 12 de 2011). *Federación Colombiana de la Industria del Software*. Recuperado el 3 de 4 de 2012, de http://www.fedesoft.org/novedades/empresas-colombianas-de-software-ti-se-capacitaron-en-tsppsp

Fedesoft. (2011). *Sector de TI en Colombia año 2011 y proyecciones 2013.* Bogotá.

Ferguson, p., Humphrey, W., Khajenoori, S., & Macke, S. (1997). Results of Applying the Personal Software Process. *IEEE Computer*, 24-31.

García, M. (12 de 01 de 2011). *Universidad Tecnológica de Izucar Matamoros.* Obtenido de Calidad en el proceso de Desarrollo de Software: http://www.utim.edu.mx/~mgarcia/DOCUMENTO/CSW/CSW02.ppt

Giraldo, W. (2010). *Marco de Desarrollo de Sistemas Groupware Interactivos.* Castilla la Mancha.

Grady, R. (1992). *Practical Software Metrics for Project Management and Process Improvement.* New Jersey: Prentice Hall.

Hartley, J. (1985). *Designing Instructional text.* New York: NPC.

Hayes, W. ". (1998). Using a Personal Software Process to Improve Performance. *IEEE*, 61-71.

Hayes, W., & Over, J. (1997). *The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers.* Pittsburgh: Software Engineering Institute.

Hogan, J., & Thomas, R. (2005). Developing the software engineering team. *Australasian Computing Education*, 203-210.

Honig, W. (2008). Teaching Successful "Real-World" Software Engineering to the "Net". *IEEE*, 25-32.

Humphrey, W. (1995). *A Discipline for Software Engineering.* Addison Wesley.

Humphrey, W. (1996). *Introduction to the Personal Software Process.* Madrid: Pearson Education.

Humphrey, W. (2000). *Introduction to the Team Software Process.* Massachusetts: Addison Wesley Longman.

Humphrey, W. (2000). *The Personal Software Process.* Pittsburgh: CMU/SEI.

Instituto Español de Comercio Exterior. (2006). *El mercado del Software en Colombia.* Bogotá.

Keefe, J. (1988). *Profiling and Utilizing Learning Style.* Virginia: NASSP.

Marquez, L. (2001). Aprendizaje Cooperativo. *FACES*, 79-88.

Mayer, R. (1984). Aids to text comprehension. *Educational Psychologist*, 30-42.

Ministerio de Comercio, Industria y Comercio. (1 de Julio de 2008). Desarrollando el Sector de TI Como uno de Clase Mundial. Bogotá, Cundinamarca, Colombia.

Ministerio de comunicaciones. (1 de Marzo de 2008). Plan Nacional de Tecnologías de la Información y las Comunicaciones. Bogotá, Cundinamarca, Colombia.

Monereo, C. (1994). *Estrategias de enseñanza y aprendizaje. Formación del profesorado y aplicación a la.* Barcelona: Graó.

Moreira, M. (2000). *Aprendizaje Significativo: teoría y práctica.* Madrid: Visor.

Moser, R., Janes, A., Russo, B., & Sillitti, A. (2005). *PROM: taking an echography of your software process.* Udine: AGILE Publications.

Navas Robleto, J. (1996). *Conceptos y Teorías de Aprendizaje.* Puerto Rico: Publicaciones Puertorriqueñas.

Novak, J., & Gowin, D. (1988). *Aprendieno a aprender.* Barcelona: Martínez Roca.

Pérez, L. (2009). Adpatación de RUP para PSP. Modelo de proceso y casos de estudio. Montevideo, Uruguay.

Pozo, J., & Postigo, I. (1993). *Las estrategias de aprendizaje como contenido del currículo. .* Barcelona: Ediciones Doménech.

Pressman, R. (2002). *Ingeniería del Software: Unenfoque práctico.* Madrid: McGraw-Will.

ProExport. (2011). *Colombia: La transformación de un País. Software & Servicios de Tecnología de Información (TI).* Bogotá.

Proexport. (2012). *Software y Servicios de TI.* Bogota.

Raymond, D. (1993). Process Improvement and the Corporate Balance Sheet. *IEEE*, 28-35.

Resnick, L. (1987). Learning in school and out. *Educational Researcher*, 13-20.

Revista Dinero. (29 de Noviembre de 2002). *Artesanos del Software*. Recuperado el 1 de Marzo de 2012, de http://www.dinero.com/Imprimir.aspx?idItem=2086

Rickards, J., & Denner, P. (1978). Inserted questions as aids to reading text. *Instruccional science*, 313-346.

Rosca, D., Li, C., Moore, K., Stephan, M., & Weiner, S. (2001). PSP-EAT – Enhancing a Personal Software Process Course. *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference*.

Runeson, P. (2001). Experience from Teaching PSP for Freshmen. *14th Conference on Software Engineering Education &*, 98-107.

Runeson, P. (2003). *Using Students as Experiment Subjects.* Lund: Citeseer.

Schmeck, R. (1998). *Individual differences and educational perspective.* New York: Academic Press.

Schunk, D. (1991). *Learning Theories. And Educational Perspectiva.* New York: Mc Millan.

Shuell, T. (1990). Phases of Meaningful Learning. *Review Of Educational Research*, 531-548.

Silvestri, L. (11 de Enero de 2006). *Comunicaciones Científicas y Tecnológicas 2006.* Recuperado el 20 de Marzo de 2012, de www.unne.edu.ar/Web/cyt/cyt2006/09-Educacion/2006-D-011.pdf

Soto, D., & Reyes, A. (2010). Introduciendo PSP en el Aula. *Revista Colombiana de Tecnologías de Avanzada*, 1-5.

Sussy, B., Calvo-Manzano, J., Cuevas, G., & Feliu, T. (2008). Teaching Team Software Process in graduate courses to increase productivity. *Annual IEEE International Computer Software and Applications Conference*, 440-446.

Tamayo, M. (1999). *El Proyecto De Investigación.* Bogotá: Limusa.

Towhidnejad, M., & Hilburn, T. (1997). Integrating the Personal Software Process (PSP) across the undergraduate curriculum. *IEEE Computer Society* , 162-168.

Tucker, A. (1991). Computing Curricula . *Communications of the ACM*, 68-84.

Tucker, A. M. (1991). Computing Curricula. *Communications of the ACM*, 68-84.

Universidad del Quindío. (24 de Julio de 2006). Acuerdo 014 de 2006. *Criterior, políticas y mecanicmos de asignacion de labora academica y administrativa*. Armenia, Quindío, Colombia.

Universidad del Quindío. (2007). *Proyecto Educativo del Programa.* Armenia: Universidad del Quindío.

Universidad del Quindío. (2010). *Estrategia Virtual.* Armenia: Optigraf.

Universidad del Quindío. (2010). *Informe Final de Autoevaluación para la Acreditación ante el C.N.A.* Armenia: Universidad del Quindío.

Universidad del Quindío. (2012). *Reporte estudiantes activos del programa de ingeniería de sistemas y computación.* Armenia.

Valle, A., Barca, A., González, R., & Núñez, J. (1999). Las estrategias de aprendizaje:. *Revista Latinoamericana de Psicología*, 425-461.

Von Konsky, B., & Robey, M. (2005). A Case Study: GQM and TSP in a Software Engineering Capstone Project. *IEEE*, 215-222.

W. S., H. (mayo de 1996). Using a Defined and Measured PersonalSoftware Process. *IEEE Software*.

W. S., H. (1996). Using a Defined and Measured PersonalSoftware Process. *IEEE Software*.

Weinstein, C. E., & Mayer, R. E. (1986). *The teaching of learning strategies.* New York: Wittrock.

Wesslén, A. (2000). A Replicated Empirical Study of the Impact of the Methodsin the PSP on Individual Engineers. *ACM*, 93-123.

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

Zhong, X., Madhavji, N., & El Emam, K. (2000). Critical Factors Affecting Personal Software Process. *IEEE Software*, 71-75.

ANEXO 1 – INSTRUMENTO PARA LA CARACTERIZACIÓN DE ESTUDIANTES

Teniendo en cuenta que Colombia se está proyectando como un país con capacidad para desarrollar software de alta calidad, es necesario que los estudiantes y profesionales inmersos en esta disciplina apliquen las mejores prácticas de software desde su formación temprana y de esta forma puedan adaptarse rápidamente a las dinámicas de trabajo de las empresas de base tecnológica.

Desde la academia se debe analizar la forma como actualmente se están formando a los profesionales y a los estudiantes de los programas relacionados con el desarrollo de software, para que aprendan a gestionar adecuadamente su proceso personal y en equipo, aplicando técnicas de medición, estimación, planificación y seguimiento de tareas. Esta gestión del trabajo personal y en equipo del desarrollo de software permitirá al ingeniero administrar un proceso definido, medido y controlado con criterios de calidad.

Por lo anterior, se realiza una encuesta a estudiantes y a personas que trabajan en la industria del software, para que comparta sus conocimientos que tiene en la aplicación de prácticas y técnicas de desarrollo de software, ya sea en trabajos o proyectos de programación. Se espera que mediante sus apreciaciones se identifique cuales son los aspectos que caracterizan las prácticas de desarrollo de software individual y de trabajo en equipo.

La información que se recolecte será para efectos de investigación y tendrá carácter confidencial.

Instrucciones para responder la encuesta

Las preguntas están clasificadas con relación a: administración del tiempo, manejo de los defectos, tamaño de los programas, trabajo individual, gestión del proyecto y proceso de desarrollo y trabajo en equipo. Lo que se recomienda será responder el cuestionario basado su experiencia.
A la derecha de cada pregunta, hay tres posibles respuestas: Sí, No y No Sé.

Responda **Sí** cuando:

- La práctica está bien establecida y es consistentemente desempeñada.

Responda **No** cuando:

- La práctica no está bien establecida o es desempeñada inconsistentemente.

Responda **No Sé** cuándo:

- No se tiene certeza de cómo responder la pregunta.

Información general

Semestre	
Asignatura de Programación o Ingeniería de Software que cursa	
Número de asignaturas de programación cursadas	

Información adicional	**Si**	**No**
¿ Actualmente trabaja en una empresa de desarrollo de software? En caso de responder Sí, Indique el nombre de la empresa _____		

Con relación a la administración del tiempo dedicado a sus trabajos o proyectos, usted:

Preguntas	**Si**	**No**	**No sé**
Registra el tiempo que invierte en la realización de un trabajo o proyecto?			
Identifica los factores que interrumpen su actividad cuando realiza un trabajo o un proyecto?			
Registra el tiempo de las interrupciones cuanto está realizando un trabajo o un proyecto?			
Estima el tiempo a invertir cuando va a realizar un trabajo o un proyecto?			
Distribuye el tiempo de un nuevo proyecto basado en la distribución del tiempo de los trabajos o proyectos anteriores?			
Analiza la cantidad de tiempo que pasa en cada fase (planificación, diseño, implementación, pruebas, etc) del proceso de desarrollo?			
Excede generalmente el tiempo establecido para realizar un trabajo o proyecto?			

Con relación al manejo de los defectos (aspectos que reducen la capacidad de los trabajos o proyectos para cumplir completa y efectivamente las necesidades de los usuarios), usted:

Preguntas	**Si**	**No**	**No sé**
Registra y analiza los defectos que ha introducido al realizar un trabajo o proyecto?			
Entiende los defectos que ha introducido al realizar un trabajo o proyecto?			

Contabiliza el tiempo que tarda en encontrar y corregir cada defecto al realizar un trabajo o proyecto?			
Aplica algún método para encontrar y corregir defectos cuando realiza un trabajo o un proyecto?			
Cuenta la cantidad de defectos que encuentra cuando realiza un trabajo o proyecto?			

Con relación al manejo del tamaño (con relación a la cantidad de líneas de código que se generan al momento de la codificación de un programa), usted:

Preguntas	Si	No	No sé
Estima el número de líneas de código necesarias para la realización de un trabajo o proyecto?			
Realiza un conteo de las líneas de código que modifica cuando está reparando un defecto?			
Determina cuantas líneas de código por hora produce en sus trabajos o proyectos?			
Planea las líneas de código por hora que va escribir en un trabajo o proyecto?			

Con relación al trabajo individual para realizar trabajos o proyectos, usted:

Preguntas	Si	No	No sé
Identifica las acciones que realizan los otros para que sus tareas o proyectos funcionen mejor?			
Aplica prácticas relacionadas con estándares de codificación?			
Documenta los principales problemas que enfrenta al momento de realizar un trabajo o proyecto?			
Utiliza diagramas o modelos cuando va a realizar un trabajo o proyecto?			

Con respecto a la planeación del proyecto y al proceso de desarrollo, usted:

Preguntas	Si	No	No sé
Planifica la forma como va a realizar un trabajo o proyecto?			
Sigue un cronograma y le hace seguimiento al trabajo o proyecto?			
Cuando realiza un trabajo o proyecto, se centra en aspectos de programación, sin tener en cuenta un proceso?			
Planifica las pruebas para verificar la funcionalidad del producto?			

Conoce y aplica las fases de proceso de desarrollo de software cuando va a realizar un trabajo o proyecto? Cuáles de esas fases realiza? _____			
Reúne información de los trabajos o proyectos para el uso y la planeación de trabajos o proyectos futuros?			

Con relación a la organización en equipo para realizar trabajos o proyectos:

Preguntas	Si	No	No sé
Se toman decisiones concertadas en equipo cuando se está realizando un trabajo o proyecto?			
Cuando trabaja en equipo, registran los defectos encontrados?			
Se identifica en los integrantes del grupo los conocimientos, aptitudes y habilidades para la conformación del equipo de trabajo?			
Se definen los roles y responsabilidades de cada miembro del equipo para realizar un trabajo o proyecto? Describa el rol_____			
Se planea ordenadamente la realización de un trabajo o proyecto cuando se trabaja en grupo?			
Se realizan revisiones en grupo sobre los defectos encontrados?			

Si considera importante adicionar otra información sobre este tema, puede hacer uso de este espacio.

ANEXO 2 - INSTRUMENTO CONOCIMIENTOS PREVIOS

Fecha: _____

Estudiante: _____ Identificación: _____

Instrucciones: Lea cuidadosamente cada una de las preguntas (1 a 9) y seleccione una sola de las opciones. De acuerdo con la opción seleccionada, de una explicación de su respuesta.

1. Cuando está llevando a cabo un trabajo de programación, ¿registra el tiempo que invierte durante la realización del mismo?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

2. Cuando está llevando a cabo un trabajo de programación y suspende esta actividad, ¿registra el tiempo que tarda en volver a retomar la tarea de programación?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

3. Cuando lleva a cabo un trabajo de programación, ¿registra cada uno de los errores que surgen durante la realización del mismo?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

4. ¿Comprende los errores de codificación que se generan cuando está realizando un trabajo de programación?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

5. Cuando se presentan errores de codificación que no sabe solucionar, ¿cuál estrategia aplica para resolverlos?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

6. ¿Tiene en cuenta los estándares de codificación cuando realiza un trabajo de programación?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

7. Cuando va a trabajar de forma individual en un trabajo de programación, ¿planifica cada una de las actividades necesarias para la realización del mismo?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

8. Cuando va a trabajar en equipo en un trabajo de programación, ¿planifica cada una de las actividades necesarias para la realización del mismo?

Siempre ( ) Algunas veces ( ) Nunca ( )

Explicación:

9. ¿Se definen las funciones y responsabilidades de cada miembro del equipo para realizar un trabajo de programación?

Siempre ( ) Algunas veces ( ) Nunca ( )

Describa las funciones_____

Explicación: