ESTUDIO EMPÍRICO DE APLICACIÓN DE PSP PARA EL DESARROLLO TRANSVERSAL DE COMPETENCIAS DE GESTION, EN ESTUDIANTES DE UN PROGRAMA DE TECNOLOGÍA EN SISTEMAS

Jhon Fredy Niño Manrique

Octubre 2012

ESTUDIO EMPÍRICO DE APLICACIÓN DE PSP PARA EL DESARROLLO TRANSVERSAL DE COMPETENCIAS DE GESTION, EN ESTUDIANTES DE UN PROGRAMA DE TECNOLOGÍA EN SISTEMAS

JHON FREDY NIÑO MANRIQUE

Trabajo de grado presentado como requisito parcial para optar al título de Magister en Ingeniería

Asesora: Raquel Anaya de Páez

MEDELLÍN
UNIVERSIDAD EAFIT
Escuela de Ingeniería. Departamento de Informática y Sistemas
Octubre de 2012

AGRADECIMIENTOS

A Dios por sus bendiciones y por la fortaleza que me ha dado en momentos de gran dificultad para poder culminar con éxito este proyecto.

A mi esposa, Saray Martínez, por su amor y comprensión, por ser mi fortaleza e inspiración cada día.

A mi asesora, Raquel Anaya, por su generosidad al compartir sus conocimientos y experiencia. Su paciencia y dedicación fueron invaluables.

A la Corporación Universitaria Adventista por su apoyo y compromiso con mi crecimiento académico.

A la Universidad EAFIT por el entorno académico que propicia un aprendizaje de alta calidad.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ANEXOS

Anexo 1. Descripción_prácticas

Anexo 2. Formatos_encuestas

ABREVIATURAS

SEI: Software Engineering Institute

PSP: Personal Software Process

TSP: Team Software Process

CMMI: Capability Maturity Model Integration

GQM: Goal/Question/Metric

PIP: Process Improvement Plan

LOC: Line of Code

PROBE: Proxy Based Estimating

HPC: High performance Computing

PMBOK: Project Management Body of Knowledge

RESUMEN

Este trabajo tuvo por objetivo determinar la viabilidad y pertinencia de aplicar el primer nivel de Personal Software Process en un programa de Tecnología en Sistemas con énfasis en desarrollo de software, a fin de desarrollar en los estudiantes, competencias técnicas y básicas de gestión de proyectos y calidad. Se diseñó y se aplicó una prueba piloto para cuatro grupos de diferentes niveles de formación.
Cada grupo realizó dos prácticas individuales, de acuerdo a su nivel de formación, de los cuales se recolectaron datos que evidenciaron el aprendizaje obtenido.

Entre los principales hallazgos están: (a) la apertura de los estudiantes de primer año aplicando PSP, en contraste con la resistencia de estudiantes del último semestre; (b) la complejidad de la herramienta de registro de tiempos y defectos que genera más trabajo al estudiante y desvían su atención del trabajo técnico; c) La relación positiva encontrada entre calidad del producto final y la adopción de PSP.

Para el diseño y ejecución del estudio empírico, este trabajo aplicó el enfoque GQM (Goal-Question-Metrics); se definieron unas metas específicas que se traducen en preguntas de investigación con unas métricas asociadas que dan respuesta a las mismas. El enfoque GQM propone unas fases para la realización de estudios empíricos, que sirvieron de referente para guiar el trabajo y definir la estructura del documento: Planeación del estudio (capítulo 3), definición del estudio (capítulo 4), análisis de resultados (capítulo 5), limitaciones y amenazas del estudio (capítulo 6).

Los principales aportes de este estudio empírico son los siguientes:

- Adecuación de los scripts de PSP según el nivel de formación de los participantes.
- Creación de material de apoyo en la plataforma virtual para capacitación, seguimiento y recolección de datos.
- La articulación de las competencias de PSP con el mapa general de competencias del programa académico en cuestión.
- La aplicación de una metodología rigurosa para definir los objetivos, preguntas de investigación y métricas del estudio.
- El esquema de aplicación del estudio de forma simultánea en diferentes cursos, con diferentes niveles de formación.

- Los resultados iniciales fueron plasmados en un artículo denominado “Estudio empírico de aplicación de la metodología PSP en estudiantes de un programa de Tecnología en Sistemas con diferentes niveles de formación”; éste fue divulgado a través de una ponencia en el XX Congreso Iberoamericano De Educación Superior en Computación (CIESC), en el marco de la XXXVIII Conferencia Latinoamericana en Informática. Dicho artículo será publicado en la IEEE xplore en el mes de noviembre del presente año.
- Asimismo, este trabajo representa un punto de partida importante para la institución, pues da inicio a una línea de trabajo que tiene el propósito de articular las prácticas de PSP en el programa de Ingeniería de Sistemas de su primera cohorte de manera incremental y a lo largo del programa.

Palabras claves: Personal Software Process, Estudio empírico, Calidad, GQM, PSP en el currículum, Team Software Process, Gestión de proyectos, desarrollo de software, Ingeniería de Software, currículos basados en competencias, competencias de PSP, competencias de gestión.

.

1. INTRODUCCIÓN

1.1 MOTIVACIÓN

Buena parte de los programas de formación relacionados con la informática, inician la formación del estudiante en principios de lógica, programación y algoritmia, y los estudiantes empiezan a desarrollar programas sin aplicar principios básicos de análisis y solución de problemas y de gestión de sus actividades, y sin realizar una reflexión acerca de la calidad de sus productos. Cuando en los cursos de Ingeniería de Software, ubicados en semestres 3 y 4 se les presenta a los estudiantes los principios de ingeniería en los que se apoya el desarrollo de software y la importancia de la aplicación de buenas prácticas durante el desarrollo del proyecto, ya el estudiante ha adquirido el mal hábito de enfrentar el problema yendo directamente a la codificación de la solución a través de la prueba y el error, sin seguir un proceso planificado y consciente.

Se presentan entonces diferentes escenarios:

- Estudiantes con capacidades excepcionales de lógica que se "enamoraron" de la programación y que los principios presentados por la Ingeniería de Software (la gestión, el proceso, el modelado, el trabajo en equipo, la documentación, entre otros) les parece una pérdida de tiempo.
- Estudiantes que ya vienen frustrados con la programación y que encuentran en la ingeniería de software una vía de escape para no programar; son estos estudiantes que se dedican a "llenar los formatos" solicitados en esta materia, sin hacer una conexión consciente entre actividades y artefactos del espacio del problema y su mapeo al espacio de la solución.
- Afortunadamente, un tercer grupo de estudiantes apropia las prácticas y técnicas que suministra la ingeniería de software para abordar problemas de complejidad mayor, a un mayor nivel de abstracción (modelos antes que código) bajo una relación costo beneficio adecuada.

Cuando se trata específicamente de programas de nivel tecnológico, es decir, programas académicos cortos de seis semestres en los cuales se propende por "el hacer" más que por "el saber", el fenómeno anterior es aun más marcado: ir directo a la construcción de aplicaciones ignorando actividades como el diseño de la solución, la planeación del trabajo, la gestión del avance del proyecto y la detección de defectos para garantizar la calidad del producto final.

Los problemas mencionados están relacionados con aspectos de gestión de los proyectos y de calidad en un nivel individual. Dichos problemas son típicos y generalizados, dado que los programas académicos en

general, tienden a fraccionar el conocimiento sacrificando la asimilación de conceptos y el desarrollo de habilidades de tipo transversal como planeación, gestión y garantía de calidad. El programa de Tecnología en Sistemas de la Corporación Universitaria Adventista, no es ajeno a dicha problemática, puesto que a través de los cursos de Ingeniería de Software 1 y 2 (tercer y cuarto semestre), y Modelos y estándares de calidad (sexto semestre) se busca formar al estudiante en la aplicación de procesos de desarrollo de software sin un resultado real en el corto plazo. Esta dificultad en la asimilación del proceso se evidencia en los problemas mencionados por los asesores de prácticas y en cursos de último semestre donde ante una asignación de desarrollo de software, el estudiante en general va directamente a las IDEs para desarrollar su proyecto, desconociendo fases como planeación, diseño o pruebas. Las asignaciones en mención, son diseñadas para un trabajo individual y para trabajos grupales, en los que independientemente de la modalidad se obvian las fases necesarias para desarrollarlas, notándose con mayor razón a nivel individual.

Por otro lado, no existen mecanismos o protocolos claros para el docente y la administración del programa, que le permitan identificar los tipos de defectos que los estudiantes inyectan en sus proyectos. Dicha información facilitaría el diseño de estrategias de mejoramiento continuo y acompañamiento al estudiante, a fin de disminuir los defectos que éste pueda venir inyectando en las diferentes fases del desarrollo de sus proyectos.

Algunos de los problemas adicionales que se han podido detectar de parte de la industria en los egresados de programas relacionados en el área de desarrollo de software son los siguientes [1]:

- Debilidad en la apreciación de los márgenes de utilidad de las empresas y la necesidad de entregar código correcto a tiempo y dentro del presupuesto.
- Debilidad en la experiencia de estimar costos, incluyendo los tiempos requeridos para completar tareas.
- Productividad personal variada

En este mismo trabajo [1], se mencionan los problemas manifestados por estudiantes de dichos programas de último semestre:

- Pocas habilidades de cronometraje o registro de tiempos
- Percepción de estar sobre cargado
- Percepción de insuficientes recursos de laboratorio
- Debilidad en la apreciación de pruebas

- Competencias de diseño inadecuadas

Igualmente, se mencionan los problemas encontrados por los docentes en su trabajo de enseñanza [1]:

- Dificultad para monitorear el desempeño de los estudiantes
- Dificultad para dar una retroalimentación inmediata y relevante a los estudiantes
- Dificultad para ajustar las asignaciones prácticas a las habilidades reales de los estudiantes

Ante los problemas mencionados, surge una opción de solución para el desarrollo de competencias relacionadas con planeación, gestión y calidad: PSP. Personal Software Process fue diseñado precisamente para ayudar a los desarrolladores a hacer bien su trabajo de manera que puedan planearlo, estimarlo y seguir lo planeado para tener un conocimiento de su rendimiento personal [2] [3]. Esta es una metodología que busca potenciar las capacidades individuales de los desarrolladores de software referentes a la autogestión y reflexión de la calidad y productividad de su trabajo. Por medio de la adopción de PSP, el desarrollador puede llegar a conocer sus tiempos de desarrollo (rendimiento) y a partir de ello estimar el tiempo necesario para ejecutar un proyecto específico bajo ciertas características gracias al registro histórico de sus tiempos reales en proyectos anteriores. De igual forma, la adopción de PSP permite formar la habilidad en el desarrollador de reconocer los defectos típicos que él mismo inyecta en las diferentes etapas de un proyecto y gracias a ello proponer planes de mejoramiento personal. Finalmente, PSP busca que el estudiante adquiera la capacidad de definir las fases y actividades necesarias en un nivel de granularidad adecuado para realizar un trabajo, de tal manera que le facilite la estimación de tiempos y control de avance del proyecto.

El objetivo de este trabajo es contrastar los resultados de aplicar PSP0 simultáneamente en diferentes cursos de la Tecnología en Sistemas, lo cual permitirá analizar los resultados de aplicación de la metodología en estudiantes que están empezando el proceso de formación en lógica y programación y estudiantes que ya han superado esa etapa inicial de formación. El trabajo sigue el protocolo de un estudio experimental aplicando las fases del proceso GQM [4]

1.2 OBJETIVOS

1.2.1 Objetivo general

Analizar la aplicación de metodología PSP nivel 0 con el propósito de evaluar su pertinencia de aplicación de manera transversal con respecto al desarrollo de competencias de gestión y control desde el punto de vista de docentes e investigadores en el contexto de estudiantes del programa de Tecnología en Sistemas de la Corporación Universitaria Adventista.

1.2.2 Objetivos específicos

- Seleccionar, categorizar y analizar trabajos de aplicación de PSP en la Academia.
- Definir la estrategia adecuada para introducir PSP dentro del programa de forma transversal.
- Realizar análisis de competencias que la aplicación de PSP ayuda a formar en los estudiantes del programa en cuestión.
- Diseñar y llevar a cabo el experimento de aplicación de PSP 0 bajo los parámetros del enfoque GQM

Para el último objetivo se generarán otros objetivos específicos adicionales conforme a lo establecido por el enfoque GQM usado en el presente estudio.

1.3 METODOLOGÍA DE TRABAJO

Se aplicó la metodología GQM propuesta por Basili [5], pero antes se realizó un estudio del estado del arte de Personal Software Process principalmente, tanto de los aspectos conceptuales como su aplicación en ámbitos académicos y los resultados obtenidos de dichos estudios.

Al mismo tiempo se empezó a trabajar en la definición del estudio experimental de acuerdo a las pautas determinadas en la metodología GQM [4]. En dicha etapa se estableció el contexto del experimento, la especificación del mismo, la población involucrada, y los instrumentos de recolección de datos.

En la etapa de Definición del estudio, se establecieron el objetivo general y los objetivos específicos del estudio, junto con las preguntas de investigación asociadas y las métricas relacionadas. A continuación se aplicó en experimento en la población seleccionado utilizando los instrumentos diseñados en la etapa anterior, con el fin de recopilar los datos necesarios para aplicar las métricas y dar respuesta a las preguntas de investigación definidas previamente. La aplicación del experimento se llevó a cabo con la totalidad de los

estudiantes del programa de Tecnología en Sistemas entre los meses de septiembre y octubre de 2011. Las prácticas aplicadas en los estudiantes formaban parte de los cursos específicos pero añadiendo las prácticas de PSP nivel 0.

Una vez recopilados los datos, se procedió a tabularlos y validarlos a fin de usarlos en las métricas definidas para responder las preguntas mencionadas previamente. A partir de eso, se redactan las conclusiones y oportunidades de mejora para trabajos futuros en este tema.

Finalmente, se escribe un artículo titulado "Estudio empírico de aplicación de la metodología PSP en estudiantes de un programa de Tecnología en Sistemas con diferentes niveles de formación", el cual es aceptado para ser presentado en ponencia en el XX Congreso Iberoamericano De Educación Superior en Computación (CIESC), en el marco de la XXXVIII Conferencia Latinoamericana en Informática.

1.4 ORGANIZACIÓN DEL DOCUMENTO

El trabajo tiene la siguiente organización: el capítulo 2 contempla el marco conceptual y los trabajos relacionados con el presente trabajo. Cada uno de los siguientes capítulos presenta las fases del estudio experimental siguiendo el enfoque GQM propuesto por Basili [5], El capítulo 3 contiene la planeación del estudio y descripción de las prácticas a asignar, el capítulo 4 presenta la definición del estudio y la forma cómo serán recolectados los datos, el capítulo 5 contiene el análisis de los datos recolectados, el capítulo 6 discute las limitaciones y amenazas del estudio; finalmente el capítulo 7 contiene las conclusiones y trabajos futuros.

2. MARCO CONCEPTUAL

En el presente capítulo, se describe de forma sintética los conceptos, metodologías y demás elementos teóricos dentro de los cuales se enmarca el trabajo. Se abordan temas como Personal Software Process, Team Software Process, Capability Maturity Model Integration, la relación entres estos modelos, herramientas de apoyo a la aplicación de PSP, definición de los estudios experimentales, y trabajos relacionados.

2.1 METODOLOGÍAS DE APOYO A LA PRODUCTIVIDAD

2.1.1 PSP como apoyo a la productividad personal

Una vez que Watts Humphrey liderará el desarrollo inicial de CMM para software, decidió aplicar los mismos principios para desarrollar programas pequeños. En el desarrollo de dichos programas, aplicó prácticas de nivel 5 incluso de CMM. A partir de 1989 se dedica tiempo completo a la investigación en PSP. Durante tres años desarrolló un total de 62 programas y definió alrededor de 15 versiones de PSP. Usó Pascal, C++ entre otros lenguajes de programación para desarrollar cerca de 25000 líneas de código. A partir de esta experiencia concluye que los principios de gestión de procesos de Demming [6] y Juran [7] fueron aplicables al trabajo individual de los ingenieros de software tal como éstos fueron aplicables en otros campos de la tecnología. Humphrey procede a escribir un manuscrito que provee a varios colegas quienes planearon enseñar cursos de PSP. Entre 1993 y 1994, se impartieron los primeros cursos de PSP y basado en esas experiencias, revisó y publicó la última versión de PSP en 1994. Watts Humphrey y el SEI continuaron trabajando en la introducción y aplicación de los principios de PSP a equipos de desarrolladores, a lo cual llamaron Team Software Process (TSP). [8]

El diseño de PSP se basa en los siguientes principios de calidad y planeación: [8]

- Cada desarrollador tiene competencias diferentes, y para ser más eficientes, es necesario que cada uno planee su trabajo de acuerdo a sus propios datos personales.
- Para mejorar constantemente su rendimiento, el desarrollador debe usar procesos bien definidos y medibles.
- A fin de generar productos de calidad, los desarrolladores deben sentirse responsables personalmente de la calidad de sus productos. Los productos de alta calidad no se obtienen por error, éstos se obtienen por un esfuerzo en hacer un trabajo de calidad.
- Es menos costoso encontrar y corregir defectos en etapas tempranas del proceso.

- Es más eficaz prevenir los defectos que encontrarlos y arreglarlos.
- La forma correcta es siempre la manera más rápida y barata de hacer un trabajo.

La forma correcta de llevar a cabo un proyecto de desarrollo de software, es planear su trabajo antes de comprometerse a iniciarlo usando un proceso definido para planificarlo. El desarrollador debe medir el tiempo que gasta en cada etapa del trabajo, los defectos que inyectan y remueven, y los tamaños de los productos, a fin de comprender su desempeño personal. Para producir de forma consistente productos de calidad, los desarrolladores deben planear, medir y hacer un seguimiento de la calidad del producto; dicha calidad debe buscarse desde el comienzo del trabajo. Finalmente, deben analizar los resultados de cada trabajo y usarlos para adelantar un proceso de mejoramiento personal.

Personal Software Process ha alcanzado un nivel de madurez que a su vez, ha permitido generar un Cuerpo del Conocimiento en PSP (PSPBOK) [9]. Sus propósitos son los siguientes:

- Definir los conocimientos y habilidades que profesionales entrenados en PSP deben dominar.
- Caracterizar las prácticas estándar que los profesionales entrenados en PSP deben poseer.
- Delinear los conocimientos y habilidades que los profesionales en PSP establecen además de las prácticas de ingeniería comunes.
- Establecer una línea de base para el desarrollo, evaluación y acreditación de cursos y programas de estudio de PSP en todo el mundo académico.
- Facilitar el establecimiento de programas de certificación que están basados en un conjunto de conocimientos y habilidades estándar acordadas.
- Proveer a los empleadores una línea base para evaluar las capacidades y habilidades de los miembros de su equipo de desarrollo de producto.
- Caracterizar las prácticas disciplinadas usados por miembros de equipos TSP auto dirigidos.

PSP establece un número de técnicas y métodos para lograr las mejoras en el desempeño individual. La estructura del proceso PSP es mostrada en el Figura 2.1. A partir de una especificación de requisitos, el primer paso es la planeación. PSP presenta un script de Planeación que guía el trabajo y un Resumen del Plan para registrar los datos de planeación. Dentro de su estructura los Scripts, por medio de los cuales el desarrollador conoce las fases, y tareas propias de cada una de las etapas necesarias para desarrollar un producto: Planeación, Diseño, Revisión del diseño, Codificación, Revisión del código, Compilación, Pruebas y

Postmortem. Mientras el desarrollador hace su trabajo, va registrando los datos de tiempos y defectos en las bitácoras o logs de defectos y tiempos, miden el tamaño del programa, e introducen esos datos en el formato de Resumen del Plan. Finalmente, entregan el producto con el formato de Resumen de Plan diligenciado.

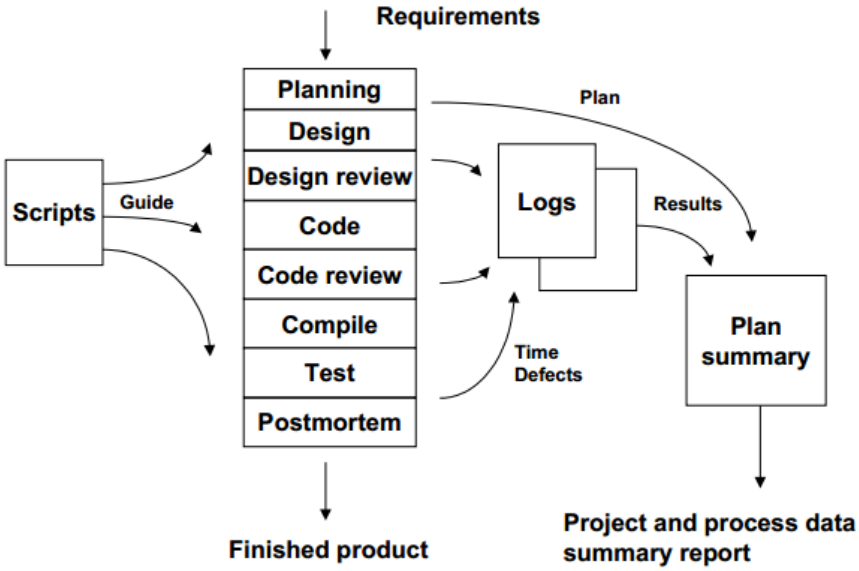


Figura 2.1 Flujo de proceso de PSP. Fuente: **[8]**

En muchos casos, la forma de introducir PSP en las Universidades ha sido por medio de un curso electivo ofrecido por el SEI [10]. En dicho curso los estudiantes escriben 10 programas y completan 5 reportes de análisis de datos. Generalmente el curso toma un semestre completo, en el cual los estudiantes siguen el proceso mostrado en la Figura 2.2 para completar 10 programas ejercicio. Se empieza con el proceso PSP0 donde se usan sus prácticas de programación actuales, registrando el tiempo que gastan en cada fase, y los defectos que encuentran. El proceso PSP va siendo mejorado a lo largo de cada versión del proceso haciendo uno o dos programas por cada versión, con el fin de que haya un aprendizaje acumulativo. Al final, en la versión PSP 3, el estudiante ha aprendido todos los métodos del proceso PSP. Este curso está dirigido normalmente para desarrolladores graduados o estudiantes de últimos semestres.

El SEI, también ha liberado material de auto estudio para el aprendiz [11] y material académico para el instructor [12] en caso de que quieran ser aplicados en ámbitos académicos.

El material de auto estudio para el aprendiz contiene los siguientes elementos:

- Student Workbook
- Scripts, formatos y estándares de PSP
- Libros de trabajo de las asignaciones (8 programas y 5 reportes)

El material académico contiene los siguientes elementos:

- Student Workbook
- Scripts, formatos y estándares de PSP
- Libros de trabajo de las asignaciones (8 programas y 5 reportes)
- Lecturas
- Guía del instructor
- Hojas de cálculo del instructor

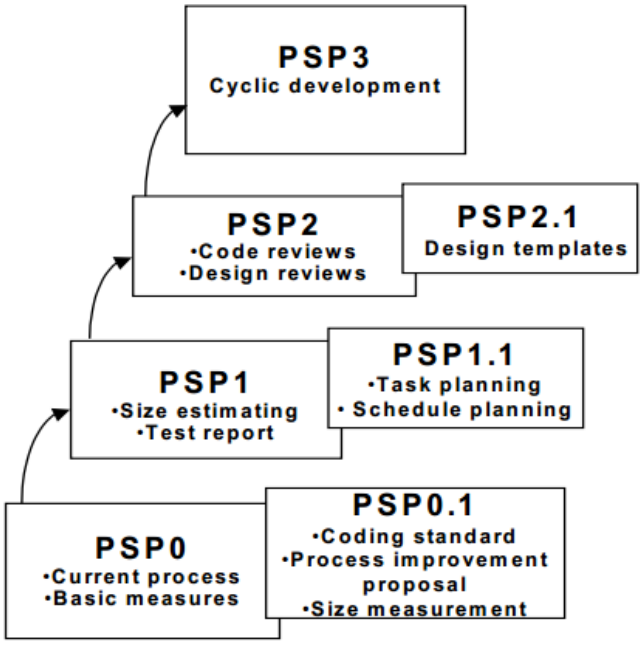


Figura 2.2 Estructura del curso de PSP por niveles propuesto por el SEI. Fuente: **[8]**

Estos niveles de mejora se explican a continuación:

- La línea base del Proceso Personal – PSP0 y PSP 0.1.

Esta línea base provee una introducción a PSP y establece una base inicial de tamaño histórico, tiempo y datos de defectos. En este nivel, se trabajan 3 programas. Se le permite a los estudiantes aplicar sus métodos actuales, pero dentro del framework de los seis pasos de la línea base mostrada en la Tabla 2.1.

Paso	Fase	Descripción
1	Planear	Planear el trabajo y documentar el plan
2	Diseñar	Diseñar el programa
3	Codificar	Implementar el diseño
4	Compilar	Compilar el programa y arreglar los defectos

		encontrados
5	Probar	Probar el programa y arreglar los defectos encontrados
6	Postmortem	Registrar el tiempo, defectos y tamaño reales

Tabla 2.1 Fases propuestas de la Línea base de PSP. Fuente: **[8]**

Se introduce una planeación y métricas básicas del proceso. En los formatos provistos por PSP se registran el tiempo de desarrollo, defectos y tamaño del programa. Se usa además un formato sencillo de Resumen del plan del proyecto usado para documentar los resultados planeados y los reales. En PSP 0.1 se entrega un formato para registrar una propuesta de mejora del proceso (PIPs). El formato de PIP provee al ingeniero una forma adecuada de registrar problemas en el proceso y soluciones propuestas.

- Gestión de proyectos personales – PSP1 y PSP 1.1

Estos niveles se enfocan en técnicas de gestión de proyectos, introduciendo estimación de esfuerzo y tamaño, programación de la planeación, y métodos de seguimiento de la programación. La estimación de tamaño y esfuerzo se obtiene usando el método PROBE (PRoxy-Based Estimating). Los proxies usados pueden ser objetos y métodos (lenguajes orientados a objetos), o funciones y procedimientos (lenguajes procedimentales). A partir de dichos proxies se hacen estimados iniciales y usando datos históricos, se convierte del tamaño relativo del proxy a Líneas de Código (LOC) y finalmente estimar el esfuerzo. Otros ejemplos de proxies pueden ser pantallas u objetos de pantalla, scripts, reportes y páginas de documentos. Usar PROBE, permite ver los objetos que deben ser desarrollados. Un breve ejemplo de aplicación del método PROBE para estimar esfuerzo en la etapa de Pruebas, es el siguiente:

Se parte de una tabla de tamaños como proxies (Ver Tabla 2.2):

Tipo de caso de prueba	Pequeño	Mediano	Grande
Interfaz de usuario	12 (pasos de prueba)	16	21
Lógica de negocio	10	15	23
Base de datos	8	17	36

Tabla 2.2 Tamaño en pasos de prueba por tipo de caso de prueba. Fuente: Adaptado Material Curso TSP Team Member Training (SEI)

En términos de productividad (Ver Tabla 2.3):

Tipo de caso de prueba	Pasos de prueba/hora
Interfaz de usuario	2.95
Lógica de negocio	8.52
Base de datos	5.68

Tabla 2.3 Productividad según Tipo de caso de prueba (CP). Fuente: Adaptado Material Curso TSP Team Member Training (SEI)

A partir de la tabla 2.2 y la tabla 2.3, se podría tener un proyecto hipotético donde se de la situación mostrada en la Tabla 2.4:

Tipo de CP	# de CP	Tamaño relativo	Tamaño estimado	Productividad	Esfuerzo estimado (horas)
Interfaz de usuario	5	M (16)	80	2.95	27.1
Interfaz de usuario	3	G (21)	63	2.95	21.4
Interfaz de usuario	2	p (12)	24	2.95	8.1
Lógica de negocio	4	G (23)	92	8.52	10.8
Lógica de negocio	6	M (15)	90	8.52	10.6
Lógica de negocio	2	P (10)	20	8.52	2.3
Base de datos	10	G (36)	360	5.68	63.4
Base de datos	20	M (17)	340	5.68	59.9
Base de datos	12	P (8)	96	5.68	16.9
TOTAL			**1165**		**220.5**

Tabla 2.4 Estimación del esfuerzo y tamaño. Fuente: Adaptado Material Curso TSP Team Member Training (SEI)

Según la tabla anterior, se tiene que el tamaño es de 1165 pasos de prueba y un esfuerzo total de 220.5 horas.

- Gestión de la calidad personal – PSP2 y PSP 2.1

En este nivel se añaden métodos de gestión de la calidad: revisiones personales de diseño y código, una notación de diseño, plantillas de diseño, técnicas de verificación del diseño, y métricas para gestionar la calidad del producto y del proceso.

La meta de la gestión de la calidad en PSP es encontrar y remover todos los defectos antes de compilar. La métrica asociada con esta meta es el Rendimiento (Yield). El rendimiento está definido como el porcentaje de defectos inyectados antes de compilar que fueron removidos antes de compilar. Se introducen dos nuevos pasos en el proceso: revisión del diseño y revisión del código. Durante la planeación debe estimarse el número de defectos que serán inyectados y removidos en cada fase. Con suficiente experiencia, el desarrollador logra eliminar entre el 60% y 70% de los defectos inyectados antes de la primera compilación. Para lograr esto, se busca que el desarrollador examine sus diseños desde diferentes perspectivas. PSP provee entonces cuatro perspectivas en el diseño: una especificación operacional, una especificación funcional, una especificación de estados y una especificación lógica.

- Proceso Cíclico Personal – PSP3

Este nivel se enfoca en la necesidad de escalar la eficiencia obtenida con PSP a proyectos más grandes sin sacrificar productividad y calidad. Para apoyar esta aproximación de desarrollo, PSP3 introduce diseño de alto nivel, revisión del diseño de alto nivel, planeación cíclica, y ciclos de desarrollo basados en los procesos de PSP2.1. Se añaden dos formatos: tamaño acumulado en un ciclo, tiempo de desarrollo y defectos para cada ciclo. Finalmente, se introduce una bitácora para registrar aspectos que puedan afectar futuros ciclos. La esencia cíclica de PSP3 puede apreciarse en la Figura 2.3.

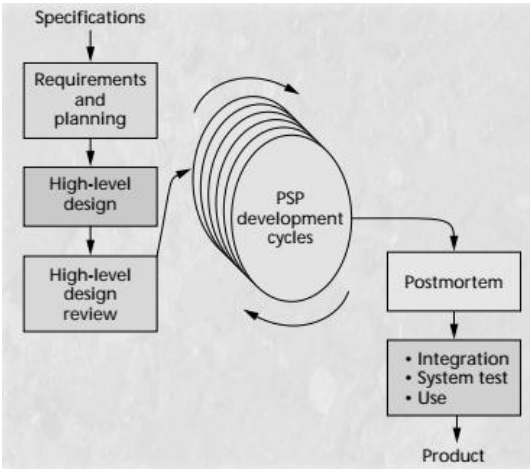


Figura 2.3 Proceso PSP3. Fuente: **[13]**

- Material y entregables por nivel de PSP

Los materiales provistos por PSP para cada nivel son diferentes de acuerdo a la Tabla 2.5 [14]:

Scripts de procesos y resúmenes	PSP0	PSP0.1	PSP1	PSP1.1	PSP2	PSP2.1
Script de Proceso PSP	X	X	X	X	X	X
Script de Planeación PSP	X	X	X	X	X	X
Script de Desarrollo PSP	X	X	X	X	X	X
Script de Revisión de Diseño PSP					X	X
Script de Revisión de código PSP					X	X
Script de Postmortem PSP	X	X	X	X	X	X
Instrucciones y Resumen del Plan de proyecto	X	X	X	X	X	X
Script de Estimación PROBE			X	X	X	X

Tabla 2.5 Scripts y resúmenes por nivel de PSP. Fuente: **[14]**

En los casos donde en cada nivel se entrega el Script de algún proceso, éste tiene algunos elementos adicionales a los del nivel anterior.

Los entregables por parte del estudiante en cada nivel son descritos en la Tabla 2.6:

Formatos, Plantillas, estándares e instrucciones	PSP0	PSP0.1	PSP1	PSP1.1	PSP2	PSP2.1
Log de Registro de tiempos	X	X	X	X	X	X
Log de Registro de defectos	X	X	X	X	X	X
PIP		X	X	X	X	X
Estándar de Codificación		X	X	X	X	X
Plantillas de reporte de Pruebas			X	X	X	X
Plantilla de estimación de Tamaño			X	X	X	X
Plantilla de Planeación de Tareas				X	X	X
Plantillas de Planeación de Cronograma				X	X	X
Lista de chequeo de Revisión de Diseño					X1	X2
Lista de chequeo de Revisión de Código					X	X
Plantilla de Especificación de Casos de Uso						X
Plantilla de Especificación Funcional						X
Plantilla de Especificación de estado						X
Plantilla de especificación lógica						X

Tabla 2.6 Entregables por versión. Fuente: **[14]**

En el caso de la lista de chequeo de Revisión de Diseño, hay algunas variaciones de PSP 2 a PSP 2.1

Para todos los niveles se entrega el estándar de tipos de defectos básicos; además se tiene una versión extendida de dicho estándar de tipos de defectos para los niveles más avanzados. Finalmente, se provee el Script de desarrollo PSP 3.0.

2.1.2 TSP como apoyo a la productividad en equipo

Por otra parte, Team Software Process (TSP), está orientado al trabajo en equipos y complementa a PSP. Usar TSP ayuda a las organizaciones a establecer una práctica de ingeniería disciplinada y madura que produce software confiable y seguro en menos tiempo y a menos costo. Ha sido aplicado en organizaciones de diverso tamaño en diferentes dominios con resultados consistentes, incluyendo mejoras en la productividad de 25% o más, reducciones en costo y variaciones en el calendario menores al 10% y reducciones en costos y tiempos de Pruebas por encima del 80% [15].

De acuerdo al TSP Body of Knowledge (TSPBOK) [15], las tecnologías PSP y TSP están basadas en la premisa de que un proceso estructurado y definido puede mejorar la eficiencia y calidad del trabajo a nivel individual. Tal como se ha mencionado previamente, PSP ayuda a los desarrolladores a definir métricas y hacer seguimiento de su trabajo para comprender mejor lo que hacen, y proveerles la información necesaria para evaluar su aprendizaje. De igual forma, TSP provee a los equipos de desarrollo de software un framework en el cual los individuos pueden combinar sus habilidades de disciplina de proceso personal con técnicas de gestión de procesos que permite trabajar con alta calidad.

TSP indica "cómo" implementar los principios de las mejores prácticas que el SEI ha promovido desde su creación. TSP incluye muchos conceptos que no han sido implementados en otros métodos de ingeniería, incluyendo la gestión de equipos auto dirigidos, sistemas de gestión de la calidad antes de Pruebas, entre otros.

2.1.3 Relación de PSP/TSP como modelos de calidad

Tal como se ha mencionado, PSP y TSP tienen una relación estrecha, toda vez que PSP promueve elementos de mejoramiento permanente a nivel individual en los desarrolladores que forman parte del equipo de trabajo y TSP aprovecha las habilidades obtenidas por la aplicación de PSP, para desarrollar trabajos en equipo. De igual forma, los modelos en mención, tienen relación con CMMI [16] tal como se presenta en la Figura 2.4:

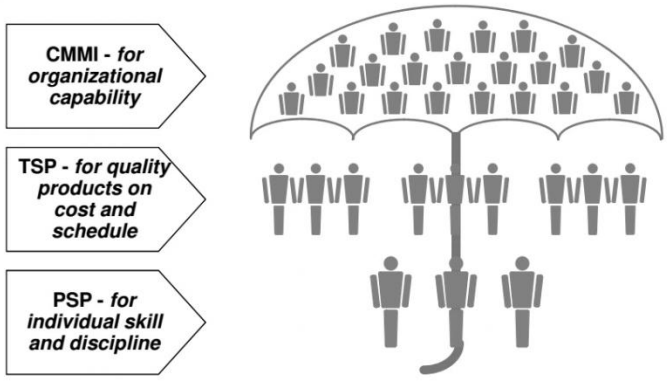


Figura 2.4 Relación de los 3 niveles operacionales de calidad. Fuente: **[16]**

PSP provee los métodos y herramientas para formar en el desarrollador la disciplina y habilidades a nivel individual, mientras TSP se enfoca en la calidad de los productos dentro del cronograma y costo previstos; finalmente CMMI se enfoca en la capacidad organizacional.

La relación con CMMI puede comprenderse a partir de una revisión de las categorías de procesos que dicho modelo define [17]:

En términos generales, TSP apoya dichas categorías de proceso según la Figura 2.5:

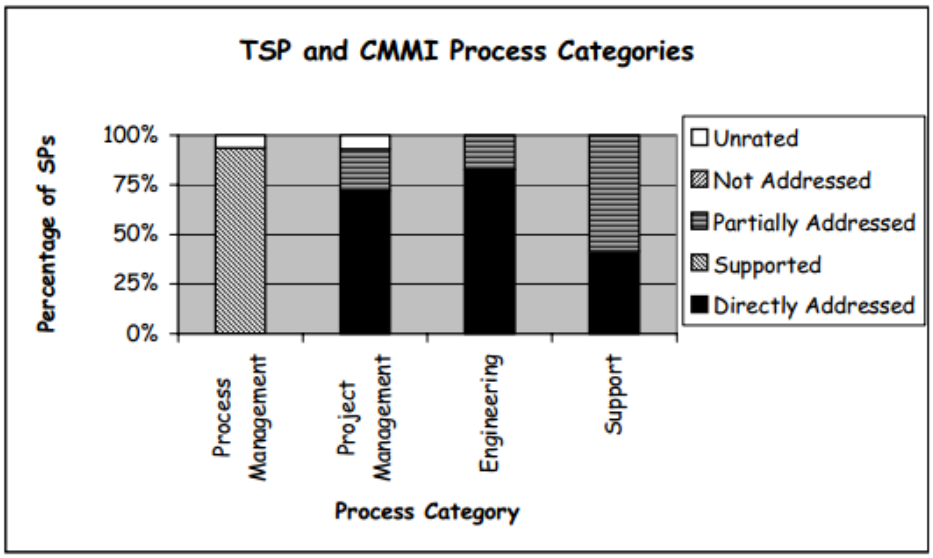


Figura 2.5 TSP y categorías de procesos de CMMI. Fuente: **[17]**

En la categoría de procesos Gestión de proyectos, más del 50% de las áreas de proceso son direccionadas marcadamente por TSP tal como lo muestra la Figura 2.6:

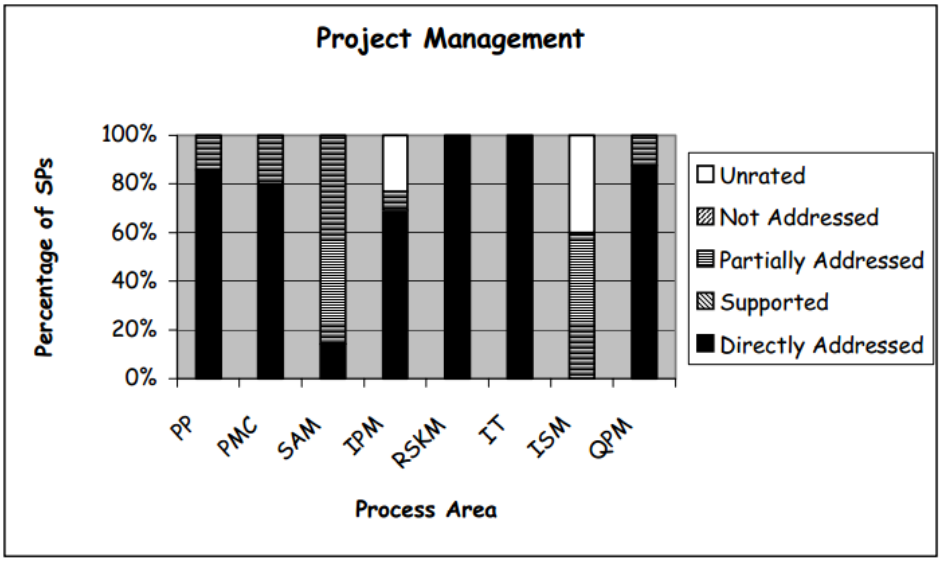


Figura 2.6 Relación TSP y Categoría de proceso Gestión de proyectos. Fuente: **[17]**

En la categoría Gestión de procesos, la mayoría de sus áreas están soportadas por TSP tal como se aprecia en la Figura 2.7:

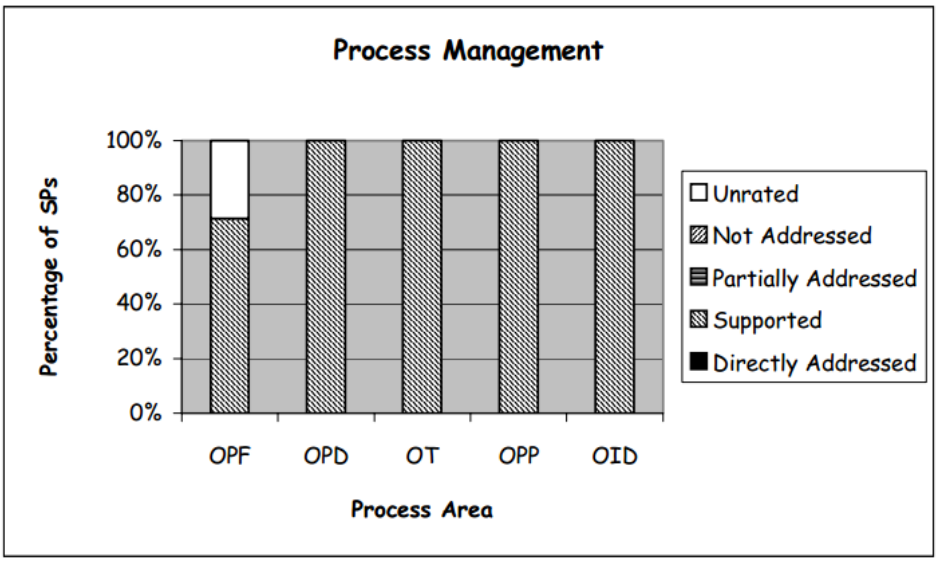


Figura 2.7 Relación TSP y Categoría Gestión de proceso. Fuente: **[17]**

En la categoría Ingeniería, la totalidad de sus áreas de proceso son directamente soportadas por TSP por encima del 70% tal como se ve en la Figura 2.8:

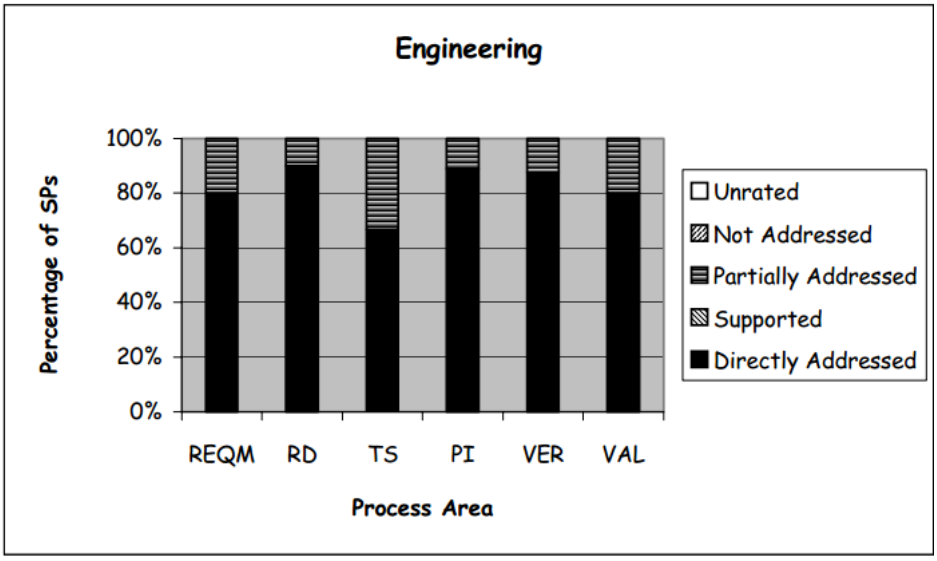


Figura 2.8 Relación TSP y Categoría Ingeniería. Fuente: **[17]**

En la categoría Soporte, no hay un patrón que se pueda observar claramente en cuanto al direccionamiento de TSP en sus áreas de proceso. Ver Figura 2.9:

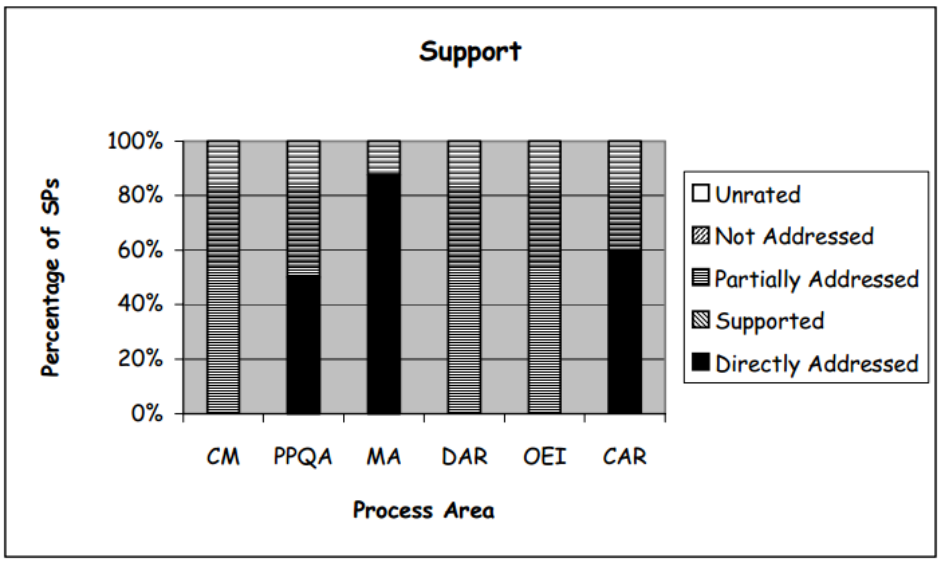


Figura 2.9 Relación TSP y Categoría Soporte. Fuente: **[17]**

En conclusión, la relación entre PSP, TSP y CMMI es lo suficientemente fuerte como para garantizar que la adopción de las mismas en las organizaciones, permite tener niveles de calidad diferenciadores.

2.2 ESTUDIOS EXPERIMENTALES

2.2.1 Principios generales

Los estudios experimentales son parte natural de muchas áreas del conocimiento como la física, biología, entre otras, puesto que son la forma de documentar los fenómenos observados. En el campo de la Ingeniería de software se viene desarrollando hasta ahora, a pesar de la madurez observada en otras áreas o campos de formación.

En términos generales, la Ingeniería de software experimental, requiere el uso científico de datos cuantitativos y cualitativos para comprender y mejorar un producto software, un proceso de desarrollo de software o gestión de software.

Basili [18] describe la evolución de la ingeniería de software experimental en las últimas cuatro décadas resaltando el estado actual de dicho campo. En nuestros días, el panorama es el siguiente:

- A partir de la construcción de cuerpos de dominio específicos, se pueden cruzar con diferentes dominios. Para cada dominio se involucran entrevistas, estudios de caso, experimentos controlados, etc. Se viene trabajando una serie de estudios con los principiantes y los profesionales que utilizan experimentos controlados (estudiantes de posgrado), los estudios observacionales (profesionales, estudiantes de posgrado), estudios de casos (proyectos de clase, los proyectos de HPC en la academia), encuestas y entrevistas (expertos en HPC).

En general, la experimentación en Ingeniería de Software vino para quedarse. Es necesario que las técnicas de Ingeniería de software sean estudiadas experimentalmente para que esta sea más que una disciplina teórica. Muchos desarrolladores de tecnología ya están haciendo estudios de viabilidad, tarea a la cual no han sido llamados pues no son experimentadores. Es clave encontrar un balance entre los teóricos y el papel de la experimentación, desarrollar un motor para la investigación experimental, desarrollar estudios donde trabajen teóricos, desarrolladores y expertos de dominio.

La dificultad en la experimentación en la Ingeniería de software radica en varios factores según [19]:

- A diferencia de otras áreas del conocimiento, la ingeniería de software trata sobre desarrollo. Esto la separa de empresas de corte manufacturero o de producción en línea.
- El software por naturaleza es complejo. Todo software tiene diferencias con otros, pues involucran muchas variables, que impactan de diferente forma. Estas variables deben ser comprendidas para entender también su efecto,
- En las empresas manufactureras normalmente se usan máquinas y son menos propensas a errores, por el contrario, la ingeniería de software se basa en humanos que pueden inyectar más fácilmente errores a lo largo de los procesos.
- Aún existen muchos aspectos por definir en la disciplina, tales como modelos que ayuden a entenderla, límites de la tecnología en diferentes contextos, entre otros.

Los paradigmas de investigación disponibles son [19]:

- Paradigma analítico: este paradigma propone un conjunto de axiomas o una teoría formal, desarrolla una teoría, consigue los resultados y, de ser posible, los verifica con observaciones empíricas.
- Paradigma experimental: este parte de observar el mundo o soluciones ya existentes; propone un modelo o una teoría del comportamiento o mejores soluciones; mide y analiza; valida o invalida las

hipótesis del modelo o teoría; repite el procedimiento llevándolo a nuestra base de conocimiento. Los paradigmas experimentales implican:

- Diseño experimental
- Observación
- Análisis cuantitativo o cualitativo
- Recolección de datos y validación del proceso o producto que está siendo estudiado.

En la relación Academia – Empresa es necesario fortalecer este tipo de estudios, puesto que, tal como mencionaba Basili, hay una brecha en la que los desarrolladores adoptan metodologías, técnicas y demás para no quedarse atrás, y la academia, que en muchas ocasiones educa a sus estudiantes en técnicas que a su vez, la industria adoptó sin ninguna validación [20].

Los elementos principales de un experimento son [21] [19]: las variables, los objetos, los participantes, el contexto del experimento, hipótesis y tipo de proyecto del experimento.

- Variables: pueden ser dependientes o independientes. Las independientes son las entradas del experimento y que afectan el resultado del experimento. Las dependientes se refieren a las salidas del experimento. Éstas variables representan el efecto de las variables independientes en el experimento
- Objetos: son herramientas usadas para verificar la relación causa-efecto de una teoría. Los objetos, sistema de medición y directrices de ejecución del experimento componen el instrumento del mismo.
- Participantes: individuos seleccionados de la población interesada en conducir el experimento. El conjunto de participantes debe ser representativo de la población de estudio. Entre más variada sea la población, la muestra debe ser más grande.
- Contexto del experimento: está definido por los siguientes elementos:
    - Clasificación del estudio [22](In vivo {se hace sobre un proyecto real} / in vitro {se hace en laboratorio con condiciones controladas} / in virtuo {objeto y ambiente modelado por computador} / in silico {sujeto, objeto y ambiente son modelados en computador)
    - Alumnos vs Profesionales: define el equipo que ejecutará el experimento
    - Problema de aula o problema real: implica el tamaño del problema.
        - Específico vs General: indica el alcance de los resultados, particulares o generales para la disciplina de Ingeniería de software.

- Hipótesis: Se tiene la hipótesis nula, que declara que no hay relación significativa entre la causa y el efecto; también están las hipótesis alternas que buscan ser validadas por el estudio.
- Proyecto de Experimentación: Define la forma con interactuarán los elementos anteriores.

El proceso de un estudio experimental obedece a la siguiente figura [19]:

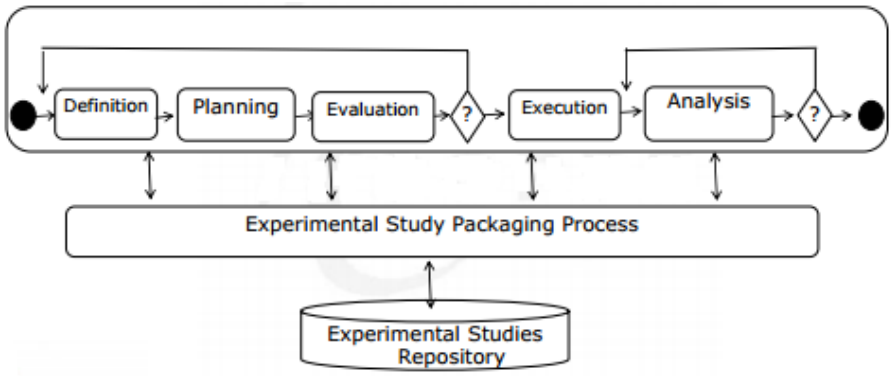


Figura 2.10 Proceso básico de un estudio experimental. Fuente: **[19]**

2.2.2 Estudios empíricos con estudiantes

En la literatura se hallan algunas consideraciones interesantes cuando se quiere realizar un estudio empírico en el área de ingeniería de software con estudiantes de pregrado especialmente. Algunas de estas consideraciones fueron [23] [24]:

- Asegurarse de que haya una adecuada integración del estudio a los temas del curso
  - En estudiantes muy avanzados es fundamental que esté articulado el tema de estudio con el tema de investigación puesto que es posible que no les interese este último.
  - El tema de la experimentación debería ser incluido en el plan de estudios.
  - Hay que considerar el uso de proyectos en equipo.
  - Determinar si el uso de un proyecto estudiantil es viable para el curso.

- Integrar el estudio al cronograma del curso
  - Normalmente, a los estudiantes que están sobre cargados de compromisos se les ve como amenaza para la validez interna, puesto que es posible que no terminen todas las tareas o tomen atajos. Sin embargo, esto también puede ser útil si el contexto de estudio se va a parecer al ambiente industrial como sería en el caso de desarrollo de software.
- Reusar de herramientas y artefactos existentes según corresponda
  Consideraciones:
  - Reusar artefactos ahorra considerablemente tiempo y esfuerzo en la creación de nuevos artefactos útiles y representativos

    - Reusar artefactos que ya han sido probados y depurados ahorra incluso más esfuerzo, y reduce el riesgo de problemas inesperados de validez.
- Redactar un protocolo y revisarlo
    - La revisión del protocolo debe ser hecha con suficiente tiempo de anticipación para corregir cualquier problema
    - Al diseñar el estudio, los investigadores e instructores deberían considerar si sirve hacerlo para trabajos en grupo.
    - Investigadores e instructores deben reflexionar sobre la conveniencia de incluir un proyecto de desarrollo.
- Obtener el permiso de los sujetos de estudio para su participación en el estudio
  Consideraciones:
    - El formulario de consentimiento es un acuerdo escrito entre docente/investigador y los estudiantes donde se explique cómo los resultados del estudio empírico influirá en sus calificaciones.
    - Un posible problema de requerir dicho consentimiento es que puede ocasionar que solo algunos estudiantes quieran participar en el estudio. Es bueno avisar a los estudiantes que sus datos permanecerán anónimos.
    - Si los estudiantes son apropiadamente informados, ellos se sentirán más cómodos de participar en el estudio y no como ratas de laboratorio, sino como participantes activos.
- Establezca las expectativas
    - Los estudiantes se sienten más cómodos cuando se tiene en cuenta el tiempo, el esfuerzo necesario y la forma de calificar, de formar que hay menos abandonos y resultados perdidos o de mala calidad.
    - Al explicar los objetivos del estudio, el investigador no debe revelar información que podría sesgar el estudio.
- Documentar información detallada sobre el contexto experimental
    - Interpretar adecuadamente los resultados y compararlos con los resultados de otros estudios. Los investigadores normalmente recolectan información de otros estudios sobre temas similares. En los estudios empíricos de ingeniería de software, los instructores deben proveer información sobre las metas del curso, los temas cubiertos, y los métodos de enseñanza usados.
    - La interacción con organizaciones profesionales le permite a los investigadores entender mejor la diferencia entre el entorno académico y el profesional y qué clase de información de trasfondo puede ser útil.
- Implementar políticas para controlar/monitorear las variables del experimento.
    - El valor educativo del experimento debe ser igual para todos los participantes del estudio. Por ejemplo, si se va a evaluar una técnica nueva, no es conveniente que solo lo aprenda un grupo y el otro no; en tal caso, es bueno que se intercambien los temas entre los dos grupos de forma que todos aprendan lo mismo.
    - Si se trabaja en grupo, se puede incurrir en problemas puesto que se organizan grupos entre los que se sienten más cómodos entre ellos, aspecto que en la industria no sucede

puesto que se obedece a necesidades corporativas. Por lo tanto se deben armar grupos aleatorios o bajo otro criterio.

- Actividades de seguimiento
    - Las actividades de seguimiento provee retroalimentación muy importante para los investigadores. Los cuestionarios y entrevistas hacen más fácil obtener información de los estudiantes sin que se sientan incómodos. Pueden hacerse primero cuestionarios, luego entrevistas y finalmente discusión en clase; ésta última técnica puede dar a conocer aspectos que pueden afectar la validez de los datos, y que las entrevistas y cuestionarios no revelan por ellas mismas.
    - Las actividades de seguimiento son una oportunidad de enseñanza fundamental, tanto para el objeto de estudio en los Estudios empíricos en Ingeniería de Software como de la misma técnica de experimentación.
    - La sesión de retroalimentación proporciona al investigador la oportunidad de elaborar detales importantes del estudio que se mantuvieron en secreto durante todo el estudio. En el caso de los investigadores les ayuda a garantizar que las conclusiones son coherentes con lo que realmente ocurrió durante el estudio
- Crear o actualizar un paquete de laboratorio
    - Los paquetes de laboratorio también deben ser vistos como un mecanismo para apoyar la comunicación entre los investigadores. A menudo, un paquete de laboratorio es el único medio para que otros investigadores puedan examinar los protocolos y artefactos en detalle, de forma que puedan proporcionar críticas bien argumentadas.

2.2.3 El método GQM

El método GQM (Goal/Question/Metric) [25] fue desarrollado inicialmente por V. Basili y D. Weiss. Más adelante fue complementado con otros conceptos de D. Rombach.

El método GQM contiene 4 fases:

- Fase de planeación: en la cual se selecciona, define, caracteriza y planifica un proyecto de aplicación de medición, lo cual genera un plan de proyecto.
- Fase de Definición: en ésta, el programa de medición es definido y documentado (se definen metas, preguntas, métricas e hipótesis).
- Fase de Recolección de datos: en la cual se recolectan los datos reales a partir de lo definido en la fase previa.
- Fase de Interpretación: en esta fase, los datos recolectados son procesados de acuerdo a las métricas definidas para responder las preguntas del estudio y de esta forma evaluar el logro de los objetivos planteados.

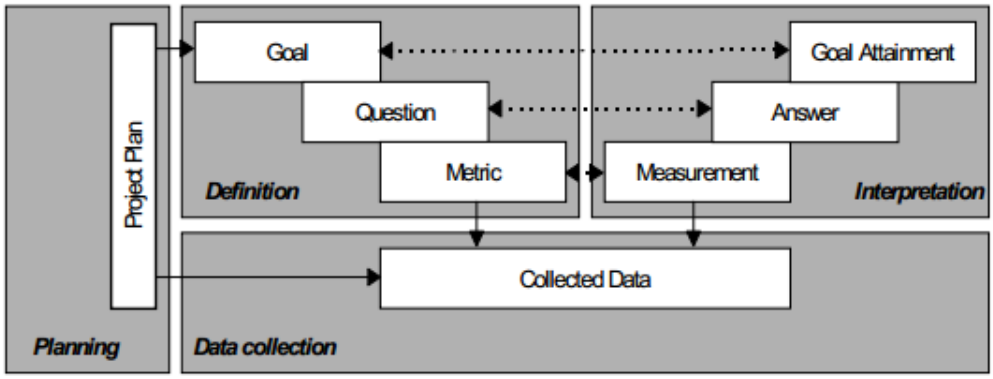


Figura 2.11 Las cuatro fases del método GQM. Fuente: **[25]**

El principio tras el método GQM [5] es que la medición debe ser orientada a la meta. De este modo, cuando se plantean este tipo de proyectos en las organizaciones, deben definirse las mediciones de las metas a partir de las metas corporativas y transformar esas metas en actividades que puedan ser medidas. Los datos recolectados pueden ser usados en otras áreas de la organización o como parte de un programa de mejoramiento continuo.

Tal como se muestra en la siguiente figura, el método GQM define unas métricas a partir de una perspectiva top-down, y analiza e interpreta los datos de medición desde una perspectiva bottom-up:

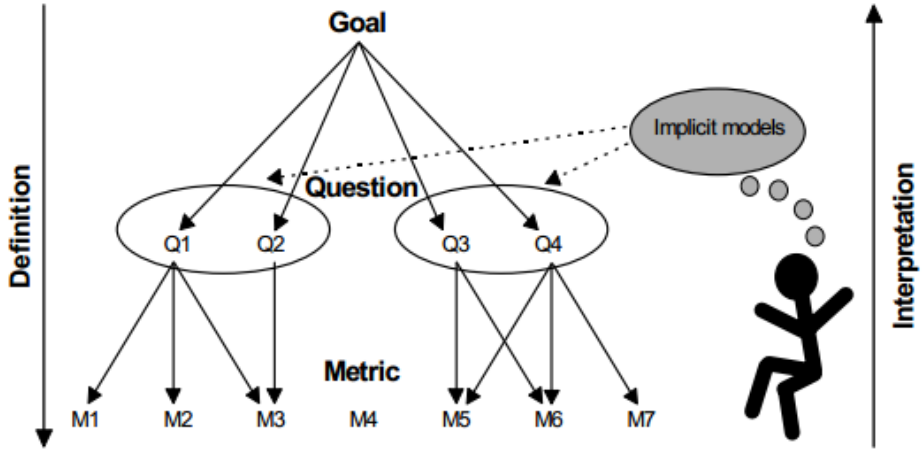


Figura 2.12 Paradigma GQM. Fuente: **[25]**

El modelo de medición que resulta de aplicar el paradigma GQM tiene tres niveles [26]:

- Nivel conceptual (Meta): Una meta se define por un objeto, por una variedad de razones, con respecto a varios modelos de calidad, a partir de varios puntos de vista, relativo a un entorno particular. Los objetos que pueden ser medidos son Productos (artefactos, entregables y

documentos producidos en el ciclo de vida del sistema), Procesos (actividades relacionadas con software que involucran tiempo como pruebas, especificación, etc) y Recursos (elementos usados en los procesos para producir resultados: personal, hardware, software, etc)

- Nivel operacional (Pregunta): se usa un conjunto de preguntas para caracterizar la forma en que una meta especifica esta siendo o no alcanzada. Las preguntas tratan de caracterizar los objetos de medición (Productos, Procesos, Recursos) de acuerdo a los aspectos de calidad seleccionado y determinar su calidad desde el punto de vista elegido.
- Nivel cuantitativo (Métrica): un conjunto de datos se asocia con cada pregunta para responderla de forma cuantitativa. Esos datos pueden ser Objetivos (no dependen de ningún punto de vista: tamaño de un programa, tiempo en horas gastado en una tarea, etc) o Subjetivos (depende del punto de vista de quien lo toma: legibilidad de un documento, nivel de satisfacción, etc)

En la siguiente tabla se amplía la información de cada una de las fases mencionadas previamente:

Fase	Pasos	Aspectos sobresalientes
1. Planeación	• Establecer un equipo GQM • Seleccionar el área de mejoramiento • Seleccionar el proyecto de aplicación y establecer el equipo de proyecto • Crear el plan de proyecto • Promocionar y entrenar	Es fundamental que los miembros del equipo de GQM estén comprometidos con el proyecto. Cuando se ha seleccionado el área de mejoramiento, se identifican: los procesos o productos involucrados, influencias (tecnológicas, organizacionales, etc), personas involucradas, la experiencia de estas personas en GQM y medición. El plan de proyecto debe incluir: resumen, introducción, caracterización, cronograma, organización, proceso de gestión, promoción y entrenamiento.
2. Diseño	• Definir las metas de medición • Revisar o definir modelos de procesos de software • Conducir las entrevistas GQM • Definir las preguntas y las hipótesis • Revisar las preguntas e hipótesis • Definir las métricas • Verificar consistencia y completitud de las métricas • Producir plan GQM • Producir plan de medición • Producir plan de análisis • Revisión de los planes anteriores	Se usa la plantilla de definición de metas que contiene los siguientes elementos: • **Analizar:** (el objeto bajo medición) • **Con el propósito de:** (comprender, controlar o mejorar el objeto) • **Con respecto a:** el enfoque de calidad del objeto en el que se centra la medición • **Desde el punto de vista de:** las personas que miden el objeto • **En el contexto de:** el ambiente en el cual se hará la medición Se crean las preguntas y sus hipótesis asociadas. Con las respuestas a dichas preguntas se establece si se cumple un objetivo determinado. De igual forma, las hipótesis planteadas son las posibles respuestas a las preguntas.

		Las métricas se definen de forma que proporcionen información cuantitativa para responder las preguntas planteadas.
3. Recolección de datos	• Mantener período de prueba • Mantener sesión Kick-off • Crear base de métricas • Recopilar y revisar los formatos de recogida de datos • Almacenar los datos de medición en la base de métricas. • Definir hojas de análisis y presentación de diapositivas	Uno de los resultados es la creación de un sistema de soporte de mediciones (MSS). Dicho sistema tiene una base de datos para el almacenamiento de métricas base, una parte para el análisis de datos, y otra parte para las presentaciones de los datos.
4. Interpretación	• Preparar una sesión de retroalimentación • Organizar y ejecutar sesión de retroalimentación • Reporte de los resultados de medición	El equipo de GQM prepara el material necesario para el informe. Se debaten los resultados de la medición y se generan conclusiones. Se desarrolla un informe con las observaciones, interpretaciones, conclusiones y acciones a realizar.

Tabla 2.7 Tabla de Fases de GQM. Fuente: Elaboración propia

2.3 TRABAJOS RELACIONADOS

2.3.1 Aplicación de PSP en la industria

Los esfuerzos de adopción de PSP en la industria de software, se articulan con TSP como enfoque de trabajo en equipo. Es generalmente aceptado que el uso de PSP/TSP complementado con otros modelos de calidad como CMMI [27] [28] [29], acelera los ciclos de mejora de las organizaciones, debido a que estos modelos (PSP/TSP) proveen formas de trabajo (cómo hacer) para los individuos y los equipos que complementan el enfoque descriptivo de CMMI (qué hacer) [30] [31] [32]. Desafortunadamente, no se encuentran muchos estudios formales sobre el impacto de aplicación de PSP/TSP en la industria, lo cual puede darse por dos razones principales: (a) Las organizaciones que han logrado o están realizando una integración consistente de PSP/TSP (generalmente para lograr ciclos de alta madurez), no están interesadas en divulgar su experiencia, pues esta representa una ventaja competitiva; (b) Las organizaciones que inician la adopción de PSP/TSP lo realizan sin una verdadera planificación y sin evaluar el impacto costo beneficio de adopción de este enfoque y por lo tanto no obtienen los resultados esperados.

2.3.2 Aplicación de PSP en la academia

Existen diversos trabajos alrededor de la adopción de PSP es su aplicación en contextos académicos como herramienta de formación. Estos trabajos presentan diferentes escenarios: estudiantes que ya han recibido formación en lógica y programación o estudiantes que hasta ahora están iniciando los cursos de formación en los primeros semestres.

Vale la pena mencionar algunos resultados de un estudio empírico realizado a ingenieros de software en 1997 [33], toda vez que da un punto de partida a la aplicación de dicho tipo de estudio en la Academia. En dicho estudio, se adaptó el curso estándar del SEI para PSP y se aplicó a 298 ingenieros, de los cuales se obtuvieron 170 conjuntos de datos completos para ser analizados. En resumen, fueron más de 300000 líneas de código escritas en alrededor de 15000 horas por 298 ingenieros, y se encontraron y removieron 22000 defectos. Se buscaba evaluar el cambio en el rendimiento individual de un nivel a otro del curso. Como conclusiones del estudio, se obtuvieron las siguientes:

- Se encontraron mejoras en cuatro dimensiones: precisión en la estimación del tamaño, precisión en la estimación del esfuerzo, calidad del producto y calidad del proceso.
- No se observaron cambios estadísticamente significativos en la productividad, sin embardo, esto quiere decir que hubo mejoras en los aspectos anteriores sin afectar la productividad.

Con el contexto anterior de aplicación en graduados, se procede a analizar algunos de los primeros trabajos relacionados con la aplicación de PSP en la Academia, como son:

- En 1998 [34], se condujo un experimento en la Escuela de Ciencias de la Computación de la Universidad Carnegie Mellon, con estudiantes de un curso de CS1 obteniendo como resultado los beneficios de aplicar PSP incluso en estudiantes novatos y no solamente para estudiantes avanzados o profesionales. Los estudiantes se hicieron conscientes de la necesidad de tener una disciplina y una planeación formal para desarrollar sus proyectos.
- En 1997 [35], en la Embry-Riddle Aeronautical University se diseñaron algunas estrategias para la integración de PSP a través del currículo a partir del proyecto DQW/PSP. En dicho artículo se plantean varias situaciones que separan la realidad de la industria del software y el mundo académico. Entre ella la asignación de proyectos más cercanos a la realidad, que no han demostrado ser efectivos puesto que los estudiantes van aprendiendo técnicas y métodos al mismo tiempo que desarrollan dichos proyectos, reciben escasa formación en planeación, trabajo en equipo, coordinación, y finalmente, un semestre es demasiado corto para aprender todos los aspectos de un

ciclo de vida. Para dar solución a los problemas mencionados previamente, desarrollaron algunas estrategias y se pueden rescatar las siguientes apreciaciones:

- Se introdujeron conceptos de ingeniería de software en los cursos CS1 y CS2. En CS1 se combinaron los temas del curso como tal y elementos de gestión de tiempo, programación de tareas, y planeación, usando materiales como: logs de tiempo, programación de tareas y documentos de informe de planeación. En las encuestas hechas al final, muchos estudiantes se quejaron por tener que aprender lo relacionado directamente con el curso, al mismo tiempo de los materiales de PSP. Se cree que esta inconformidad obedece a la inmadurez tanto mental como técnica de los novatos, la dificultad para observar los beneficios de la metodología y problemas de orientación vocacional.
- En CS2, se cubren los temas de estructuras de datos e introducen los aspectos de gestión de defectos de PSP, además de los cubiertos en CS1. De esto se obtuvo una amplia información para los docentes.
- Es necesario simplificar algunos formatos de PSP, especialmente en los cursos dictados a los más novatos.

- En este mismo año, 1997, Thomas B. Hilburn [36] plantea una propuesta para la educación en Ingeniería de Software en la cual tiene en cuenta la triada: Personas, Tecnología y Proceso. En este último elemento de la triada incluye PSP, TSP y CMMI como metodologías y modelos que facilitan el desarrollo de habilidades en planeación, gestión y calidad a nivel individual, de equipo y de organización. Dicho modelo se resume en la siguiente figura:

A SOFTWARE ENGINEERING CURRICULUM MODEL

Calendar	Topic	Courses
Year 1	Computer Science	Programming, Data Structures & Algorithms I
	SE Lab	Individual Programming, PSP
	General Education	Calculus I and II, Discrete Math, Physical Sciences I, and two from History, Literature, Art, Music, and so on
	Communications	Written and Oral Communication
Summer 1	SE Lab	Small Team Project
Year 2	Computer Science	Data Structures & Algorithms II, Computer Organization, Data Management, Programming Languages
	SE Lab	Individual Programming, PSP
	General Education	Probability and Statistics, Differential Equations, Physical Sciences II, and one from History, Literature, Art, Music, and so on
	Communications	Written and Oral Communication
Summer 2	SE Lab	Industry Internship
Year 3	Computer Science	Operating Systems, Seminar, one elective
	Software Engineering	Planning, Requirements Engineering, Formal Methods, Quality Assurance
	SE Lab	Team Project, TSP
	General Education	Physical Sciences II, Human Factors, and two from History, Literature, Art, Music, and so on
Summer 3	SE Lab	Industry Internship
Year 4	Computer Science	Computer Architecture, Seminar, and one elective
	Software Engineering	Project Management and Business Practices, Software Design, Ethics and Professionalism
	SE Lab	Team Project, TSP
	General Education	Economics and three from Mathematics, Science, History, Literature, Art, Music, and so on

Figura 2.13 Modelo curricular de Ingeniería de Software. Fuente: Tomado de **[36]**

Tal como se puede apreciar, se plantea la integración de PSP y TSP a lo largo del programa, con talleres de aplicación de dichas metodologías en los cursos regulares y en los veranos. En el año tres y cuatro se introducen en el tema de Ingeniería de software, conceptos de planeación, gestión de proyectos y aseguramiento de calidad, pero con la particularidad de haber tratado dichos temas desde el primer año a través de los laboratorios de ingeniería de software al aplicar PSP.

La realización de estudios empíricos en Ingeniería de Software, en los cuales participen estudiantes novatos, ha sido ampliamente debatida tal como se menciona en el estudio hecho por Per Runeson [37] para efectos de generalización. En dicho estudio se comparó el desempeño en estudiantes de primer año, estudiantes de postgrado y estudios en el ámbito industrial (de difícil acceso). Se pudo concluir que:

- Las tendencias de mejora pueden ser identificadas en los tres grupos, aunque la dispersión es mayor en el grupo de novatos (primer año). En todos los casos la precisión en la estimación y detección de defectos presentan mejoras.
- Los estudiantes de posgrado hacen las tareas en menor tiempo que los estudiantes de primer año.
- Los resultados, aunque no son determinantes en el tema en debate, planteó un punto de partida para estudios como el actual.

En la tabla 2.8 se presenta una síntesis de algunos estudios de referencia más recientes relacionados con el presente trabajo:

Año - Institución	Objetivos	Aspectos sobresalientes y Resultados
2008 - National Taipei University of Technology [38]	Definir estrategias y herramientas para medir el desempeño de los estudiantes de cursos de Programación.	Proponen varias estadísticas del proceso como: estadísticas del tiempo, familiaridad con la sintaxis, defectos complicados para ajustar, defectos frecuentes, productividad y densidad de defectos. A partir de estas estadísticas se pueden trazar las estrategias para mejorar la calidad de la enseñanza. Se crea una herramienta de apoyo llamada PDAT.
2001 - Birla Institute of Technology and Science [39]	Experimentar en un curso de Ingeniería de software la integración de algunos elementos de PSP con el fin de crear buenos hábitos en el desarrollo de software de forma temprana. Esta prueba se aplicó a un grupo de Ingeniería de Software de 36 estudiantes que no habían tenido ningún contacto con PSP.	Se asignaron varias prácticas cuyo peso en el curso fue del 25% para motivarlos. Se hizo lectura por parte de los estudiantes de 10 documentos de PSP a lo largo del curso. A medida que hacían sus prácticas iban registrando datos propios de PSP. Los estudiantes sintieron que el curso les ayudó a entender la forma en que se debe trabajar para mejorar la productividad, pero a la mayoría no les gustó tener que llenar los formatos y hacer la recolección de datos (tiempos y defectos) Se encontró que la aplicación de PSP es viable y útil con un impacto positivo y medible. Algunos problemas de enseñar PSP en este curso es mantener a los estudiantes motivados en esta metodología más allá de los detalles de la misma, de manera que lo más difícil es mantener una suficiente auto disciplina y encontrar una herramienta de soporte apropiada.
2000 - University of Memphis [40]	Es una aproximación a la incorporación de PSP en dos cursos de Ingeniería de software tradicional de la Universidad de Memphis.	De las 14 semanas destinadas al curso de Ingeniería de Software, se dedicaron 5 para enseñar PSP. Al comienzo, se usó un formato PSP0 y lo aplicaron varias veces mientras adquirían experiencia. Luego se usó un formato de resumen similar al especificado para PSP1. Se fijaron en la calidad de los datos recolectados advirtiendo a los estudiantes de la importancia de la precisión para poder aprobar el curso. Se tuvieron 4 poblaciones de estudiantes, desde aquellos que tenían entre 1 y 2 años académicos y otros que tenían 10 años entre academia y trabajo en la industria. Los estudiantes aventajados (con alguna experiencia en el sector industrial) de los cursos donde se aplicó PSP fueron los primeros en ver los beneficios de dicha metodología. Los menos aventajados tuvieron problemas en comprender las ventajas a largo plazo y lo consideraron un esfuerzo innecesario. La gran mayoría estuvieron de acuerdo en que el Resumen del Plan les ayudó a comprender el proceso de desarrollo de software y tratar de seguirlo. De igual forma,

		vieron las ventajas de registrar tiempos, tamaños y defectos a fin de lograr un mejoramiento en su desempeño.
2004 - University of New Mexico [41]	Se buscó la integración de PSP a un curso introductorio de Bases de datos en un plan de estudios de MIS.	En cuanto a percepciones de los estudiantes, se manifestó el problema de tener que contar manualmente las SQL LOC puesto que no hay herramientas que lo hagan. El tiempo gastado en las lecturas de PSP les quitó tiempo para lo concerniente al aspecto técnico del curso. Por su parte los docentes tuvieron algunas observaciones como la necesidad adicional de capacitación en PSP y el tiempo adicional necesario para preparar su curso de Bases de datos integrando PSP. Normalmente los cursos de bases de datos son débiles en integrar los conceptos de Ingeniería de software, y PSP se presenta como una oportunidad de integrarlos. Tuvieron que hacer adaptaciones de los scripts, del "lenguaje" a usar en términos de estandarizar la codificación y tipos de defectos ya fuera usando las vistas de diseño de MS Access o SQL por líneas de comando. También se adaptaron las asignaciones o trabajos de forma que hubiera un acercamiento cíclico tal como es inherente en PSP, es decir, se partieron los proyectos de bases de datos a fin de proveer varios ciclos de desarrollo en el semestre para que PSP pudiera ser aplicado.
2011 – Universidad ORT Uruguay [42]	Se creó un curso independiente de los correspondientes a su plan de estudios donde se adaptaron los principales elementos de PSP. Para dicho curso los estudiantes ya dominaban lenguajes de programación y entornos de desarrollo.	Se adaptaron las técnicas y herramientas a las circunstancias del curso de manera que el estudiante fuera capaz de seleccionar cuáles prácticas incluir en sus procesos. Las asignaciones que propone PSP para el entrenamiento no fueron tenidas en cuenta, por lo tanto se crearon prácticas más pertinentes y que mantuvieran el interés de los estudiantes. Las asignaciones debían ser desarrolladas en lenguajes que ya conociera el estudiante a fin de que el enfoque fuera aprender la metodología antes que un lenguaje específico. Se encontró que la adaptación del curso de PSP no afectó el desempeño promedio obtenido en comparación con los cursos tradicionales de PSP. No fueron efectivos en medir la efectivdad del curso en términos de defectos. Hubo una buena recepción en general del curso.
1997 – 2000 Purdue University [43]	Se aplicó a varios grupos empezando desde los cursos básicos a la par de los temas propios de programación. Se usaron diferentes métodos de	Uno de los problemas principales fueron los métodos de recolección, puesto que inicialmente usaron plantillas de Word para registrar tiempos, tareas y defectos. En el siguiente curso se usaron hojas de cálculo para tal fin. En dichas hojas de cálculo registraron los logs de varios proyectos, lo cual facilitaba al instructor hacer una revisión del

	recolección de datos. Todo con el fin de evaluar la viabilidad de aplicar PSP en los cursos de programación.	progreso del estudiante. Otro problema fue estudiar PSP al mismo tiempo que lo relacionado con programación, además de ser opcional. Luego se hizo a la par y obligatorio. Se presentaron muchos errores en el registro de tiempos y defectos. No se tuvo acogida por parte de los estudiantes y se pensó en dejar dichos temas como opcionales.
2005 - Curtin University of Technology [44]	Este estudio fue aplicado a estudiantes de un curso de ingeniería de software de tercer año. Se pretende evaluar los esfuerzos gastados en actividades no relacionadas con la asignación, a fin de determinar el patrón de evaluación y lo apropiado del alcance de la asignación.	Se obtuvieron algunos aspectos a tener en cuenta en los contenidos del curso: • Enseñarle a los estudiantes a hacer seguimiento del tiempo para que aprendan a gestionarlo mejor • Acondicionar las experiencias de aprendizaje según los datos de esfuerzo recolectados para que sean alcanzables. • Clasificar en sub grupos de acuerdo al esfuerzo requerido para hacer las mismas asignaciones y tomar medidas de acuerdo a ello. • Enseñar desde muy temprano en el programa lo relacionado con la gestión del tiempo
2000 - University of Karlsruhe [45]	Estudio aplicado a estudiantes de maestría. Se trabajó con un grupo que habían pasado por un entrenamiento previo en PSP y otro grupo que no conocía la metodología.	Los principales hallazgos fueron los siguientes: • Los programas hechos por el grupo entrenado en PSP fueron más robustos ante entradas inusuales. • El grupo entrenado hizo mejores estimaciones de productividad en líneas de código por hora. • El tiempo total de desarrollo del grupo entrenado fue mayor que el reportado por el otro grupo. Se considera que se debe al tiempo adicional de revisión que finalmente condujo a una mayor robustez de sus productos. • En general, no hubo mayor diferencia en la productividad.Debido quizás a que muchos de los participantes del grupo entrenado no lo aplican en sus proyectos actuales.
2010/2011 - Software Institute of Nanjing University [46]	En los cursos de verano de 2010 y 2011 se tuvieron los cursos de PSP. Entre el primero y el segundo hubo mejoras en la forma de desarrollar dicha capacitación.	Se incorporan conceptos de Programación en pareja a fin de facilitar el aprendizaje de la metodología. En el curso de 2010 no se obtuvieron significativas mejoras en el desempeño. Para el curso de 2011 se rediseñaron los entornos de laboratorio, métodos de evaluación, selección de estudiantes y nuevas métricas. El trabajo en parejas demostró que trae mejores resultados en términos de aprendizaje en comparación con el aprendizaje individual en un curso de PSP

Tabla 2.8 Síntesis de Trabajos relacionados. Fuente: Elaboración propia

Todos los estudios referenciados en la Tabla 2.8, fueron desarrollados en el marco de programas en el área de Ingeniería de Sistemas (seis profesionales y dos de posgrado). En general, estas experiencias fueron orientadas a la adquisición de las habilidades que PSP propone por parte de los estudiantes, integrando la metodología PSP a cursos dentro del plan de estudios o creando un curso dedicado completamente a PSP. Es interesante notar que en el estudio referenciado de la Universidad de Karlsruhe [45], los beneficios de PSP en cuanto a productividad, no pueden ser conseguidos completamente si el individuo no aplica dicha metodología en sus proyectos de forma consistente. Este último aspecto lleva a pensar en la necesidad de formar a los futuros desarrolladores desde el comienzo y a lo largo de sus estudios en esta metodología, a fin de ayudar a crear disciplina y obtener los beneficios de PSP.

En 2002 se realizaron otros estudios de aplicación en diferentes universidades [47] cuyos resultados y lecciones aprendidas están sintetizados en la Tabla 2.9:

Universidad	Entorno	Alcance	Herramienta de Soporte	Aspectos sobresalientes y Resultados
Umea University	Curso de estudiantes de segundo año en C++ Curso de Ingeniería de Software de segundo y tercer año	Solo PSP1 modificado	Herramienta desarrollada localmente	Se aplicó PSP de manera opcional, lo cual no fue efectivo. Se aplicó un proyecto de equipo (desarrollo de una herramienta para PSP). Fue una buena experiencia de aprendizaje.
University of Utah	CS1 / CS2 Ingeniería de Software Senior	Todo PSP	Herramienta desarrollada localmente	No significó mucha carga sobre los estudiantes y equipo de docentes. Los estudiantes que trabajaron en parejas superaron a los que trabajaron solos.
Montana Tech	CS1	PSP-Lite (Introducción a PSP)	Hojas de cálculo	Lo relacionado con PSP fue adicional al curso CS1 y los ejercicios asignados no estaban correlacionados. Los estudiantes se resistieron al comienzo a la recolección de datos, pero finalmente entendieron las ventajas de hacerlo.
	Junior CS/SE	PSP completo		
Drexel University	Estudiantes de maestría en Ingeniería de Software	PSP completo hasta PSP2	Formatos estándar	Fue una buena experiencia de aprendizaje aunque significó un arduo trabajo para estudiantes e instructores.

Tabla 2.9 Lecciones aprendidas en cinco universidades. Fuente: Adaptado de **[47]**

En general, hubo una buena recepción de la metodología aunque el gran obstáculo fue la recolección de datos y los instrumentos usados para ello.

2.3.3 Herramientas usadas para registro en PSP

Existen diversas herramientas de apoyo para la aplicación de PSP, cada una de ellas con diversas ventajas y desventajas. Algunas de ellas son: PSP Studio [48], PSP Time Tracker [49], Process Dashboard [50], Hackystat [51], Jasmine [52], PSP Student Workbook (incluido en el material de auto estudio del SEI), entre otras [53].

La Tabla 2.10 presenta un estudio comparativo de algunas de las herramientas mencionadas previamente [54]:

Herramienta	Registro de tiempo	Registro de defectos en compilación	Conteo de LOC	Reportes	Estimación	Otras características
Time Track	Manual	-	-	Reportes relacionados con el tiempo	-	Registra datos en XML
PSP Log Control	Manual	Manual	Manual	Solo archivos de registro	Asistente de planeación (no soporta la estimación automatizada)	-
PSP Studio	Manual	Manual	Manual	Resumen de Plan de Proyecto		Soporta PSP0 a PSP3
PSP 2000	Auto	Manual	Manual	-	-	Versión piloto para Palm
Process Dashboard	Auto	Manual	Auto (pero fuera de línea)	Resumen de Plan de proyecto, Gráficos de estimaciones y defectos	Estimación de tamaño automático	-
Hackystat	Auto	Auto	Auto	Resúmenes y gráficos	-	Datos recolectados por herramientas de software separadas

Tabla 2.10 Comparación de algunas herramientas de apoyo a PSP. Fuente: Adaptado de **[54]**

En la siguiente tabla se puede observar otro estudio de algunas herramientas de soporte de PSP [55]:

Ítem evaluado	PSP Studio	Dashboard	Hackystat	PSPA	JASMINE	PSP.Net
Entorno	Solo Windows	Windows, Unix, Linux, Macintosh, etc	Eclipse, Visual Studio, Jbuilder	Open Source IDE	Eclipse, Microsoft Office, Jbuilder	Windows y Unix
Scripting	No se indica	Java	Java	Java, C	Java, XML	PHP

Recopilación de datos	Automático y manual	Automático y manual	Automático	Automático	Automático	Automático y Manual
Interfaz	Sencilla y fácil de aprender	GUI	GUI	GUI y Ventanas emergentes	GUI	GUI y Ventanas emergentes
Asistente de planeación	Plantilla de PSP	Plantilla de Plan de proyecto	Plantilla de Plan de proyecto	Plantilla de cronograma y planeación	Plantilla de Plan de proyecto	Plantilla de cronograma y planeación
Contador de LOC	No	Automático (pero offline)	Automático (Basado en sensores)	Automático (Basado en sensores)	Automático (Basado en sensores)	Automático
Defect Sharing	No	No	No	Si (automático)	No	Si (automático)
Usuarios que acceden	Solo desarrollador	Solo desarrollador	Solo desarrollador	Director de proyecto y desarrollador	Solo desarrollador	Solo desarrollador
Privacidad del usuario	No	No	No	Si (Login)	No	Si (Login)
Reportes	Reporte de análisis gráfico	Resumen de Plan de proyecto, Tablas y gráficas (defectos y estimación)	Resúmenes y gráficas	Resumen de Plan de Proyecto, Gráfico (Ranking y clasificación de defectos)	Resumen del Informe de Pruebas	Gráficas y resumen de Informe
Agente de Interfaz	No	No	No	No	No	No
Características particulares			Caja de herramientas	Portal de Gestión de conocimiento		Basado en la web

Tabla 2.11 Estudio comparativo de herramientas 2. Fuente: **[55]**

En otro estudio [56], Jouni Lappalainen hace varios hallazgos interesantes. Entre ellos un análisis comparativo de herramientas de apoyo para PSP y un listado de características deseables en una futura herramienta para PSP.

Dichos hallazgos fueron resultado de aplicar la metodología en cinco cursos en la Universidad de Oulu (años: 2000, 2002, 2003 y 2004) y la Copenhagen Business School (año: 2001). En promedio se tuvieron 24 participantes por curso.

Los resultados de la evaluación de las 10 herramientas analizadas fueron los siguientes, en cuanto a la aceptación de parte de docentes y estudiantes:

Herramienta	Aceptación %
Documentos plantilla	39%(docente) / 39% (estudiante)
Student – authored tools	36$ / 30%
PPogControl	61% /58%
EvalPPLog	42% / 36%
OpenStat	39% / 39%
Logfile parsers	39% / 36%
LEAP	76% / 58%
Hackystat	64% / 55%
PSPStudio	73% / 67%
Process Dashboard	97% / 91%

Tabla 2.12 Niveles de aceptación de las herramientas de apoyo a PSP. Fuente: **[56]**

Tal como puede observarse en la Tabla 2.12, la herramienta con mayor aceptación de parte de docentes y estudiantes fue Process Dashboard.

En el proyecto actual se trabajó con la herramienta que proporciona el SEI como parte del material de auto estudio (PSP for Eng Student V4.1). Dicha herramienta se denomina PSP Student Workbook.mde (aplicación en Access). Esta herramienta no se integra a ninguna plataforma de desarrollo específica y por supuesto, es completamente compatible con las necesidades de PSP; para efectos del presente estudio, es conveniente que sea así, puesto que en dicho estudio participaron estudiantes del curso de Lógica de programación, en el cual no se trabaja con ningún lenguaje de programación específico y solo se abordan los pseudocódigos como forma de representación de algoritmos. Por otra parte, el principal objetivo era tener un acercamiento a la definición de fases, actividades y tareas para el desarrollo de las prácticas, la identificación de defectos y el registro de tiempos y defectos.

3. PLANEACIÓN DEL ESTUDIO

3.1 DEFINICIÓN DEL CONTEXTO

El programa académico en el cual se aplicó el estudio empírico es el programa de Tecnología en Sistemas de la Corporación Universitaria Adventista. Dicho programa es de nivel tecnológico constituido por seis semestres. El perfil principal de este programa está orientado al desarrollo de software. Dada la naturaleza de esta Tecnología, está orientada más al "Hacer" que al "Saber", a diferencia de programas profesionales.
Los cursos relacionados directa o indirectamente con el área de desarrollo de software y sus áreas correspondientes se pueden observar en la Tabla 3.1:

Área	Semestre 1	Semestre 2	Semestre 3	Semestre 4	Semestre 5	Semestre 6
Algoritmia y programación	Lógica de Programación (Algoritmos y pseudocódigo)	Programación 1 (POO - JAVA)	Programación 2 (POO - JAVA)	Programación 3 (Desarrollo web)	Programación 4 (Programación móvil)	
		Estructura de datos				
Ingeniería de Software			Ingeniería de Software 1	Ingeniería de Software 2		Modelos y estándares de calidad
				Bases de datos		

Tabla 3.1 Cursos del área de desarrollo de software del programa de Tecnología en Sistemas UNAC. Fuente: Elaboración propia

Los cursos resaltados en gris, fueron los elegidos para aplicar el estudio empírico objeto de este trabajo.

3.1.1 Mapa de competencias del programa y apoyo de PSP

A partir de los cursos mostrados en la tabla anterior, se busca desarrollar las competencias técnicas necesarias para que los futuros tecnólogos en sistemas puedan desempeñarse principalmente en el área de desarrollo de software. Uno de los intereses es formar futuros tecnólogos que además de estas habilidades técnicas relacionadas con lenguajes de programación, entornos de programación, entre otros, tengan otro tipo de habilidades que las empresas de software requieren. Dichas habilidades o competencias blandas corresponden a aspectos transversales como autogestión, disciplina, mejoramiento permanente, entre otros, que le garanticen al individuo un mejor desempeño y calidad en sus productos.

El programa de Tecnología en Sistemas de la Corporación Universitaria Adventista, en el cual fue aplicado el estudio presentado en este trabajo de grado, obedece a un acercamiento a un diseño curricular basado en competencias. Dicho acercamiento dirigió también el diseño del programa de Ingeniería de Sistemas que iniciará en febrero de 2014. En éste último, se aplicaría también la integración de PSP en los contenidos técnicos del programa.

Carlos Ibañez Bernal [57], menciona que el propósito de un programa de educación superior es formar individuos competentes, entendiendo como competente, que el profesional sea capaz de desempeñarse con efectividad y responsabilidad ante situaciones problémicas. En este mismo aspecto, el perfil profesional es un esquema de lo que se espera que el profesional sea capaz de aplicar para solucionar problemas con eficiencia, eficacia y elementos éticos de su área. Esta misma definición puede aplicarse a un *perfil de competencias profesionales,* en el sentido que se entiende como el modelo de competencias que debería tener un individuo en una disciplina específica. A partir de lo anterior surge la planeación de currículo basado en competencias profesionales.

Como punto de partida para dicho diseño, se tienen las problemáticas sociales a las que se enfrentaría el profesional. Esto genera a su vez, las competencias específicas, que constituyen las tareas propias de una profesión específica.

Carlos Ibañez, plantea también la necesidad de evaluar dichas competencias por medio de talleres y otras estrategias. En el presente trabajo se buscó evaluar algunas de las competencias de interés relacionados con gestión y calidad.

Competencias de Gestión de proyectos

El área de interés de este estudio es la formación en competencias de gestión de proyectos a nivel individual, en este sentido, el programa responde a esta necesidad inicial partiendo de las competencias establecidas por la Guía del PMBOK [58], que a pesar de ser diseñado para proyectos que involucren más de un individuo, permiten establecer algunas competencias a nivel personal que luego serán de utilidad en su trabajo en equipos de desarrollo.

PMBOK establece nueve áreas de conocimiento:

- Integración
- Alcance
- Tiempo
- Costo
- Calidad
- Recursos Humanos

- Comunicaciones
- Riesgos
- Subcontrataciones

Las áreas de mayor interés en este trabajo son las de Tiempo y Calidad.

Área de conocimiento: Gestión del Tiempo de proyecto [58]

Los procesos y competencias que fueron tenidas en cuenta son las siguientes:

- Definición de las actividades: Identifica y define las actividades específicas del cronograma que deben ser realizadas para producir los diferentes productos entregables del proyecto.
- Establecimiento dela Secuencia de las Actividades: identifica y documenta las dependencias entre las actividades del cronograma.
- Estimación de recursos de las actividades: estima el tipo y las cantidades de recursos necesarios para realizar cada actividad del cronograma.
- Estimación de la duración de las actividades: estima la cantidad de períodos laborables que serán necesarios para completar cada actividad del cronograma.
- Desarrollo del cronograma: analiza las secuencias de las actividades, la duración de las actividades, los requisitos de recursos y las restricciones del cronograma para crear el cronograma del proyecto.
- Control del cronograma: controla los cambios del cronograma del proyecto

Área de conocimiento: Gestión de los Recursos Humanos del Proyecto [58]

Los procesos y competencias que fueron tenidas en cuenta son las siguientes:

- Desarrollar el Equipo del Proyecto: mejorar las competencias y la interacción de los miembros del equipo para lograr un mejor rendimiento del proyecto.
- Gestionar el Equipo del Proyecto: hacer un seguimiento del rendimiento delos miembros del equipo, proporcionar retroalimentación, resolver polémicas y coordinar cambios a fin de mejorar el rendimiento del proyecto.

Área de conocimiento: Gestión de la calidad del proyecto [58]

Los procesos y competencias que fueron tenidas en cuenta son las siguientes:

- Planificación de Calidad: identificar qué normas de calidad son relevantes para el proyecto y determinando cómo satisfacerlas.
- Realizar Aseguramiento de Calidad: aplicar las actividades planificadas y sistemáticas relativas a la calidad, para asegurar que el proyecto utilice todos los procesos necesarios para cumplir con los requisitos.
- Realizar Control de Calidad: supervisar los resultados específicos del proyecto, para determinar si cumplen con las normas de calidad relevantes e identificar modos de eliminar las causas de un rendimiento insatisfactorio.

Las competencias mencionadas previamente son especificadas en la Tabla 3.2 junto con las competencias de PSP que ayudan a formar esas competencias a nivel de proyectos individuales. Algunas competencias fueron fusionadas con competencias netamente técnicas de las sub áreas de programación y pruebas, este es el caso de algunas relacionadas con Gestión de calidad del proyecto. En dicha tabla, se especifican las competencias que PSP contribuye a formar indirectamente en los estudiantes, desde el punto de vista técnico en el área de Desarrollo de software y Gestión de pruebas.

El interés central de articular PSP en este programa es evaluar de qué manera PSP puede contribuir al desarrollo de las competencias definidas; por esa razón en la Tabla 3.2 se identifican las competencias que se espera soportar con PSP, y la forma como esto se logrará.

Área	Sub área de competencia	Competencia específica	Competencia PSP	Indicador de logro	Tarea de PSP relacionada en el estudio
Desarrollo de software	**Programar**	Dominio de lenguajes de programación	Identifica los tipos de errores y a partir de ellos ejerce acciones correctivas	ECD, PCD, Recurrencia de errores	Registro de defectos según estándar PSP/Revisión al final de cada fase
		Capacidad para verificar la calidad del software desarrollado.	Revisión en búsqueda de defectos al final de cada fase. Hace uso de datos de defectos recolectados para orientar nuevas revisiones.	ECD, PCD, Recurrencia de errores	Registro de defectos según estándar PSP/Revisión al final de cada fase
		Capacidad para definir estándares de calidad del software.	Hace uso de estándares de codificación, conteo y defectos	ECD, PCD, Recurrencia de errores	Registro de defectos según estándar PSP/Revisión al final de cada fase
	Pruebas básicas de lo realizado	Capacidad para verificar la calidad del software desarrollado.	Revisión en búsqueda de defectos al final de cada fase/Hace uso de datos de defectos recolectados para orientar las nuevas revisiones.	ECD, PCD, Recurrencia de errores	Registro de defectos según estándar PSP/Revisión al final de cada fase
Gestión de pruebas	**Pruebas**	Habilidad para analizar errores propios y de los demás.	Conoce los tipos de defectos y los estandariza	ECD, PCD, Recurrencia de errores	Registro de defectos según estándar PSP/Revisión al final de cada fase
		Capacidad para aplicar y ejecutar de estándares de calidad	Gestionar la calidad de los productos producidos/Establece estándares para productos y procesos	ECD, PCD, Recurrencia de errores	Registro de defectos/Revisión al final de cada fase
		Habilidad para reportar y analizar errores	Identifica los tipos de errores y los registra describiéndolos	ECD, PCD, Recurrencia de errores	Registro de defectos/Revisión al final de cada fase
Gestión de proyectos	**Gestión Integración del proyecto**	Habilidad para apoyar el desarrollo del plan del proyecto.	Monitorear la ejecución de sus procesos y tomar decisiones para el próximo paso/Sigue su propio proceso definido/Sigue un plan definido	PTC	Sigue el proceso definido en el FPET y se ve reflejado en la pestaña Time Log del Student Workbook.mde

	Gestión alcance del proyecto	Habilidad para describir los procesos requeridos en el proyecto	Definición de procesos personales a partir de los dados por PSP/Adecuación o particularización de procesos a proyectos/	PPA, PET	Particularización los procesos de PSP a un proyecto específico usando la plantilla FPET
		Habilidad para documentar el plan del proyecto	Definición de procesos personales a partir de los dados por PSP/Adecuación o particularización de procesos a proyectos/	PPA, PET	Particularización los procesos de PSP a un proyecto específico usando la plantilla FPET. Registro de tiempos en el StudentWorkbook.mde
	Gestión tiempos del proyecto	Capacidad para definir las actividades a tener en cuenta	Logran descomponer un proyecto grande en partes pequeñas para lograr una mejor estimación	PPA, PET	Definir las fases, actividades y tareas necesarias para desarrollar el proyecto. Establecer el orden de las mismas y estimar el tiempo necesario para desarrollarlas. Se usa el FPET
		Habilidad para documentar las actividades.	Logran descomponer un proyecto grande en partes pequeñas para lograr una mejor estimación	PPA, PET	Define las fases, actividades y tareas al nivel en el cual puede describir de forma concreta lo que se realizará. Se usa el FPET
		Habilidad para realizar seguimiento de cronogramas	Monitorear la ejecución de sus procesos y tomar decisiones para el próximo paso/Sigue su propio proceso definido/Sigue un plan definido	PTC	Sigue el proceso definido a partir de los scripts de PSP y las particularidades del proyecto en el orden que estableció. Se usa el Student Workbook.mde en la pestaña Time Log para registrar los tiempos y tareas que se van realizando
	Gestión de calidad del proyecto	Habilidad para planear, asegurar y controlar la calidad del proyecto	Planear, Definir y Ejecutar tareas de Revisión al final de cada fase en búsqueda de defectos para su eliminación, estimando el número de éstos inyectados en cada fase.	PPA, PTC	Sigue el proceso definido. Específicamente en lo relacionado con la revisión al final de cada fase. Estima el número de defectos inyectados y eliminados en cada fase.
	Gestión de los Recursos humanos del proyecto	Habilidad para conocer las deficiencias y habilidades del personal adscrito al proyecto	Registrar los tiempos de ejecución de las actividades planeadas y registrar los tipos de defectos que se inyectan para plantear planes de mejoramiento.	Recurrencia de errores, Tipos de defectos inyectados	Registra el tiempo usado en cada actividad y fase. Registrar los tipos de defectos inyectados. Registro de desempeño. Diseño de plan de mejoramiento posterior al postmortem

Tabla 3.2 Mapping de competencias del programa de Tecnología en Sistemas y PSP. Fuente: Elaboración propia

Adicional a las competencias técnicas (competencias duras) visualizadas previamente, es importante mencionar que cada vez más, las organizaciones requieren empleados cuyas *competencias blandas*, hayan sido desarrolladas adecuadamente, y la industria del software no es la excepción [59] [60]. En todo programa

académico, se desarrollan actividades extracurriculares, pero éstas no siempre son suficientes o no generan interés en los estudiantes. La implementación de Personal Software Process en el currículo podría facilitar el desarrollo de dichas competencias. Algunas de ellas son:

- Fuerte ética del trabajo. PSP insta al desarrollador a hacer su mejor trabajo, a buscar la calidad de sus procesos y productos.
- Buenas habilidades de comunicación. PSP le permite al desarrollador estar completamente enterado en qué punto del proyecto se encuentra y transmitir de esa forma, de manera clara, el avance de un proyecto a sus clientes y compañeros de trabajo.
- Habilidades de gestión de tiempo: La aplicación de PSP por medio del registro de tiempos facilita al desarrollador ser consciente del tiempo dedicado a cada tarea, el tiempo gastado en otro tipo de tareas y dar prioridad a cierto tipo de tareas.
- Jugar en equipo. El desarrollador se hace consciente con PSP y TSP, de su papel como miembro de un equipo, toda vez que sus asignaciones terminarán impactando positiva o negativamente al equipo de desarrollo, en cuanto siga las tareas planeadas o respete los tiempos estimados a través del método PROBE.
- Confianza en sí mismo: PSP, además de ser una metodología para la calidad, le facilita al desarrollador tener un conocimiento claro de sus capacidades y alcances, en términos de desempeño, basándose en sus registros históricos.
- Habilidad para aceptar y aprender a partir de las críticas: Uno de los aspectos que la metodología en cuestión ayuda a formar, es la capacidad de detectar los tipos de errores particulares y a partir de ellos plantear planes de mejoramiento.
- Capacidad para trabajar independientemente: PSP forma en la capacidad de definir su propio plan de ruta en un proyecto específico, respetar dicho plan y gestionarlo.

3.2 ESPECIFICACIÓN DE LAS PRÁCTICAS

Como ya fue mencionado, para la realización de este estudio se seleccionaron cuatro cursos de diferentes niveles que se encuentran en diferentes semestres del programa (Ver tabla 3.1). Ninguno de los cursos había tenido algún acercamiento práctico o teórico a la metodología Personal Software Process.

En cada curso se realizaron dos prácticas consecutivas separadas por una semana, con un tiempo límite de dos semanas para cada práctica. En consonancia con el foco de PSP, las prácticas fueron individuales.

Los grupos que participaron en el estudio fueron:

- Lógica de Programación (C1): Semestre 1
- Programación 1 (C2): Semestre 2
- Programación 3 (C3): Semestre 4
- Modelos y estándares de calidad (C4): Semestre 6

En los cursos C1, C2 y C3, las prácticas estaban relacionadas con los temas del curso específico. Estas prácticas formaron parte del trabajo obligatorio del estudiante en la materia, a fin de evitar que fuera algo adicional a su trabajo normal que implicara sobre esfuerzo de los estudiantes. En el curso C4 las prácticas estaban relacionadas con temas ya vistos por los estudiantes en semestres anteriores con el objetivo de reforzar los conceptos de diseño y desarrollo; a la vez, en este curso, la metodología PSP sí hace parte de temática de la materia. Las prácticas aplicadas para cada curso fueron de diferente naturaleza y alcance entre sí, excepto las de los cursos C3 y C4.

En la siguiente tabla se relacionan brevemente los cursos en los cuales se aplicó el estudio, y las prácticas o problemas asignados; para cada práctica se presenta de manera sintetizada el objetivo de aprendizaje, el problema a resolver y la descripción de la práctica:

Curso	Descripción del curso	Descripción Práctica 1	Descripción Práctica 2
C1 Lógica de Programación	Número de estudiantes: 12 Temática del curso: Estructuras básicas de programación, Representación de Algoritmos: Pseudocódigo. Semestre: 1	C1-P1 Objetivo de aprendizaje: Aplicación de los conceptos de archivos, estructuras de control y sumadores. Problema: Manejo de archivo de censo electoral y tres consultas sobre el archivo.	C1-P2 Objetivo de aprendizaje: Uso de funciones y procedimientos. Problema: Construir una calculadora trigonométrica.
C 2 Programación 1	Número de estudiantes: 9 Temática del curso: Introducción a la Programación Orientada a	C2-P1 Objetivo de aprendizaje: Aplicación de los primeros conceptos de orientación a objetos (clases, métodos y atributos), interacción con el usuario en línea de	C2-P2 Objetivo de aprendizaje: Uso de arreglos bidimensionales. Problema: Construir un Juego: Triqui, de

	Objetos usando JAVA. Semestre: 2	comandos, usar diagrama de clases para diseño. Problema: Desarrollar el juego "¿Quién quiere ser millonario?".	tres en línea con dos jugadores o con un jugador vs máquina.
C3 PROGRAMACIÓN 3	Número de estudiantes: 12 Temática del curso: Desarrollo de aplicaciones web: J2EE y Oracle. Semestre: 4	C3-P1 Objetivo de aprendizaje: aplicación de matrices usando javascript. Problema: Desarrollar el juego: "Concéntrese", armar parejas en un tablero de 6x6 con banderas; uso de JavaScript, Servlets y Jsps.	C3-P2 Objetivo de aprendizaje: interacción con bases de datos y manejo de perfiles y seguridad. Problema: Desarrollar una aplicación web para encuestas de docentes, considerando diversos roles (administrador, profesor, soporte); manejo de privilegios según rol; uso de base de datos.
C4 Modelos y estándares de calidad	Número de estudiantes: 13 Temática del curso: Introducción a modelos y estándares de calidad como ISO/IEC 9126, CMMI, PSP, TSP. Semestre: 6	C4-P1 Objetivo de aprendizaje: aplicación de matrices usando javascript. Problema: Desarrollar el juego: "Concéntrese", armar parejas en un tablero de 6x6 con banderas; uso de JavaScript, Servlets y Jsps.	C4-P2 Objetivo de aprendizaje: interacción con bases de datos y manejo de perfiles y seguridad. Problema: Desarrollar una aplicación web para encuestas de docentes, considerando diversos roles (administrador, profesor, soporte); manejo de privilegios según rol; uso de base de datos.

Tabla 3.3 Descripción de Prácticas por curso. Fuente: Elaboración propia

Las competencias a formar en cada práctica a partir de la aplicación de PSP son las mencionadas en la Tabla 3.2. "Mapping de competencias del programa de Tecnología en Sistemas y PSP". Al interior de cada práctica se definen las competencias de tipo técnico y las directamente relacionadas con el tema de estudio del curso específico. Dichas competencias pueden ser observadas en el Anexo 1. Descripción_prácticas

El nivel o paso de PSP seleccionado para este estudio es PSP 0, puesto que el objetivo fue hacer una prueba piloto inicial donde se pudieran observar de manera preliminar las tendencias en cuanto a la planeación, estimación de tareas, control de avance y reconocimiento y corrección de defectos.

Ninguno de los cursos había tenido algún acercamiento a la metodología. Antes de empezar las prácticas se realizó una capacitación de 5 horas en los conceptos básicos de PSP, las herramientas y formatos a usar.

Estas herramientas fueron las siguientes: formato de planeación y estimación de tareas, aplicativo para registro de tiempos y defectos del SEI, encuestas aplicables al final de cada práctica y una encuesta general al final de la experiencia.

Inicialmente se tuvo una población participante de 46 estudiantes, sin embargo, al final los resultados se analizaron sobre una población de 23 estudiantes por diferentes razones tal como se muestra en la Tabla 3.4. Los resultados parciales de los estudiantes que cancelaron o no entregaron algunos de los instrumentos no se tuvieron en cuenta para no afectar la consistencia del análisis de los datos obtenidos.

Motivos para no incluir	C1	C2	C3	C4	Total	%
Cancelación de materia	2	3	2	0	7	15,2
No entregó la práctica 1	0	1	0	1	2	4,3
No entregó la práctica 2	1	0	2	0	3	6,5
No entregó ninguna práctica	0	0	1	0	1	2,2
No contestó las encuestas	0	1	1	5	7	15,2
No entregó el FPET de la práctica 2	1	0	0	0	1	2,2
No diligenció PSP Student Workbook.mde de práctica 2	0	0	1	0	1	2,2
No diligenció ningún PSP Student Workbook.mde	1	0	0	0	1	2,2

Tabla 3.4 Motivos de no inclusión en el estudio. Fuente: Elaboración propia

Las principales causas para nos ser tenido en cuenta en el estudio fueron la cancelación académica del curso (15,2%) y no haber diligenciado las encuestas al final de cada práctica (15,2%). Se puede observar, que fueron los estudiantes más avanzados los que estuvieron menos comprometidos con la experiencia dado que cinco de ellos no contestaron las encuestas. Algunos de estos estudiantes manifestaron personalmente que estaban más interesados en aprender los aspectos técnicos del curso, que en adoptar una metodología que de alguna forma les interrumpía en este proceso de aprendizaje.

3.2.1 Paquetes de prácticas

A través del curso creado en la plataforma virtual de la Institución (moodle), se hizo entrega a cada grupo o curso un paquete que incluía el Formato de Planeación y Estimación de Tareas (FPET), la especificación de la práctica y el Student WorkBook.

La descripción detallada de cada práctica se encuentra en el Anexo1.Descripción Prácticas

3.3 DESCRIPCIÓN Y ADAPTACIÓN DE LOS INSTRUMENTOS DE CAPACITACIÓN, REGISTRO Y RECOLECCIÓN DE DATOS

Los scripts de PSP 0 fueron adaptados para el curso de Lógica de Programación buscando que las fases de Compilación y Pruebas Unitarias tuviesen una connotación diferente que la definida en PSP 0, puesto en este curso los estudiantes aun no trabajan con un lenguaje de programación específico, la fase de compilación fue adaptada para realizar una revisión manual en búsqueda de defectos de sintaxis principalmente, y en la fase de Pruebas unitarias se aplicaron Pruebas de escritorio con unos parámetros de entrada para realizarlas.

Aprovechando la misma plataforma virtual de Institución (moodle), se preparó un espacio de comunicación en el cual fueron inscritos todos los estudiantes que participaron del estudio. Dentro de dicho espacio, se alojaron los materiales de estudio generales que incluían video tutoriales, teoría sobre la metodología, Student Workbook, scripts de PSP y material de estudio adicional. Adicionalmente, se establecieron en dicho espacio virtual los medios de descarga de las prácticas y envío de los instrumentos de registro de PSP con las soluciones de sus prácticas individuales.

A continuación se describen los materiales usados en la capacitación y auto estudio de PSP:

3.3.1 Video tutoriales

Se realizaron video tutoriales explicando particularmente el uso del Formato de Planeación y Estimación de Tareas (FPET) y el aplicativo Student WokBook.mde. Cada uno de estos videos tutoriales fueron alojados en Youtube y enlazados en el curso.

Se crearon dos versiones de los video tutoriales, una de ellas para los estudiantes de Lógica de programación y otra para el resto del curso, dado que hubo la necesidad de adaptar los scripts y conceptos de PSP 0 al nivel de estudiantes de primer semestre.

Los enlaces a los videos tutoriales en youtube son los siguientes:

- Lógica de programación y Programación 1
    - Parte 1: http://www.youtube.com/watch?v=_rL5kesycdU
    - Parte 2: http://www.youtube.com/watch?v=tYMITWcmLJ8&feature=relmfu

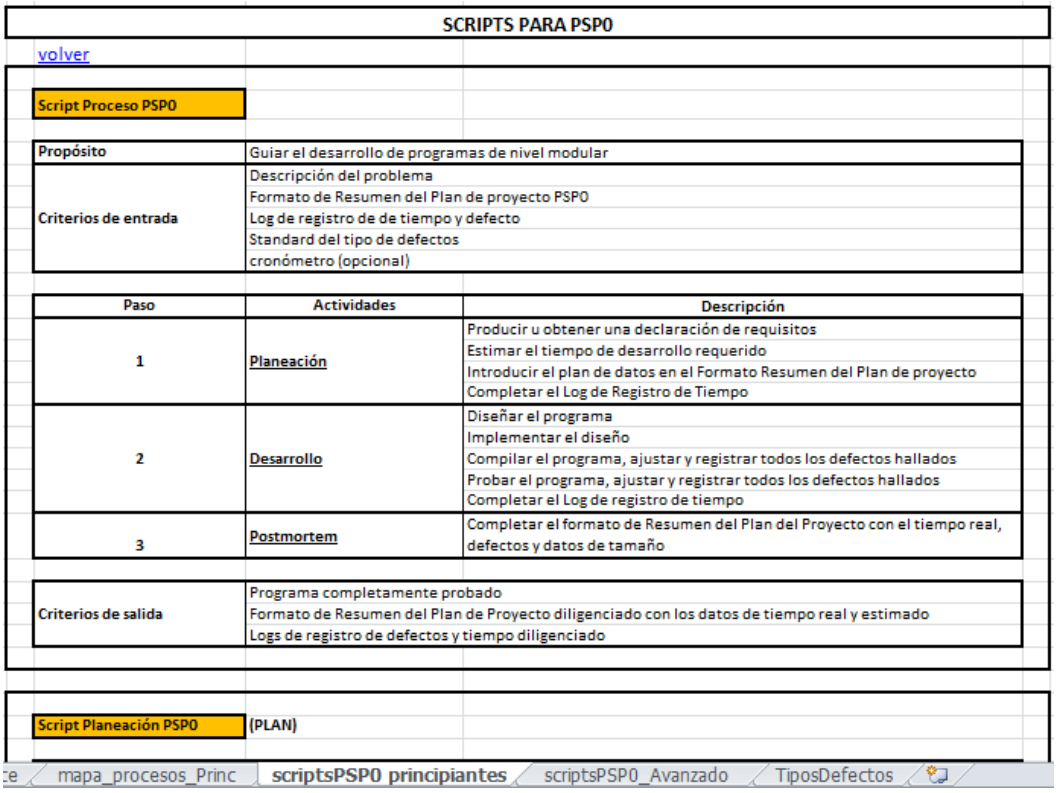
- Cursos avanzados (Programación 3 y Modelos y estándares de calidad)
    - Parte 1: http://www.youtube.com/watch?v=YO379aq6Fxs&feature=relmfu
    - Parte 2: http://www.youtube.com/watch?v=B3-4J0PlNS4&feature=relmfu

3.3.2 Conceptos de PSP y Uso del PSP 0

Se realizaron dos presentaciones donde se resumen los aspectos fundamentales de la metodología en términos generales como sus objetivos y métricas básicas, al igual que los scripts relacionados específicamente con PSP 0.

3.3.3 Scripts de PSP 0

- De acuerdo a las especificaciones de PSP 0, se especifican unos guiones o scripts para la ejecución de proyectos de desarrollo de software individual. Este material se realizó en Excel y tiene la siguiente estructura:
- Índice: Facilita el acceso a la información dentro del documento
- Mapa de procesos principal: Permite al estudiante ver de manera conceptual las fases principales de la metodología, los artefactos insumos y resultantes de cada fase.
- Scripts PSP 0 para principiantes: Los scripts de PSP 0 tuvieron que ser adaptados para los estudiantes de primer semestre del curso de Lógica de programación. Conceptos como Pruebas unitarias tuvieron que ser adaptados a sus conocimientos de manera que desarrollaran tareas similares a éstas. PSP fue diseñado para estudiantes de cursos de programación a partir de su primer acercamiento a la Programación Orientada a Objetos.
- Scripts PSP 0 para Avanzados: Estos Scripts son directamente la traducción de los descritos en el material del SEI sobre PSP y están orientados a los estudiantes de Programación 1, Programación 3 y Modelos y estándares de calidad.
- Tipos de defectos: En esta sección se explica la categorización de los defectos que los estudiantes registrarán en el Student WorkBook.



SCRIPTS PARA PSP0

volver

Script Proceso PSP0

Propósito	Guiar el desarrollo de programas de nivel modular
Criterios de entrada	Descripción del problema Formato de Resumen del Plan de proyecto PSP0 Log de registro de de tiempo y defecto Standard del tipo de defectos cronómetro (opcional)

Paso	Actividades	Descripción
1	Planeación	Producir u obtener una declaración de requisitos Estimar el tiempo de desarrollo requerido Introducir el plan de datos en el Formato Resumen del Plan de proyecto Completar el Log de Registro de Tiempo
2	Desarrollo	Diseñar el programa Implementar el diseño Compilar el programa, ajustar y registrar todos los defectos hallados Probar el programa, ajustar y registrar todos los defectos hallados Completar el Log de registro de tiempo
3	Postmortem	Completar el formato de Resumen del Plan del Proyecto con el tiempo real, defectos y datos de tamaño

Criterios de salida	Programa completamente probado Formato de Resumen del Plan de Proyecto diligenciado con los datos de tiempo real y estimado Logs de registro de defectos y tiempo diligenciado

Script Planeación PSP0 (PLAN)

mapa_procesos_Princ | scriptsPSP0 principiantes | scriptsPSP0_Avanzado | TiposDefectos

Figura 3.1 Vista material de capacitación en PSP0 (Scripts). Fuente: Elaboración propia

Cada uno de los elementos anteriores estuvieron acompañados de una capacitación por cada grupo de cinco horas en total, además del acompañamiento presencial en las clases donde podían hacer preguntas adicionales mientras iban resolviendo cada práctica dentro de los tiempos determinados. De la misma forma, el acompañamiento se hizo de manera virtual a través del Foro de preguntas sobre la metodología que estuvo alojado en el curso virtual.

La distribución del espacio o curso virtual con los elementos de capacitación pueden ser observados en la siguiente figura:

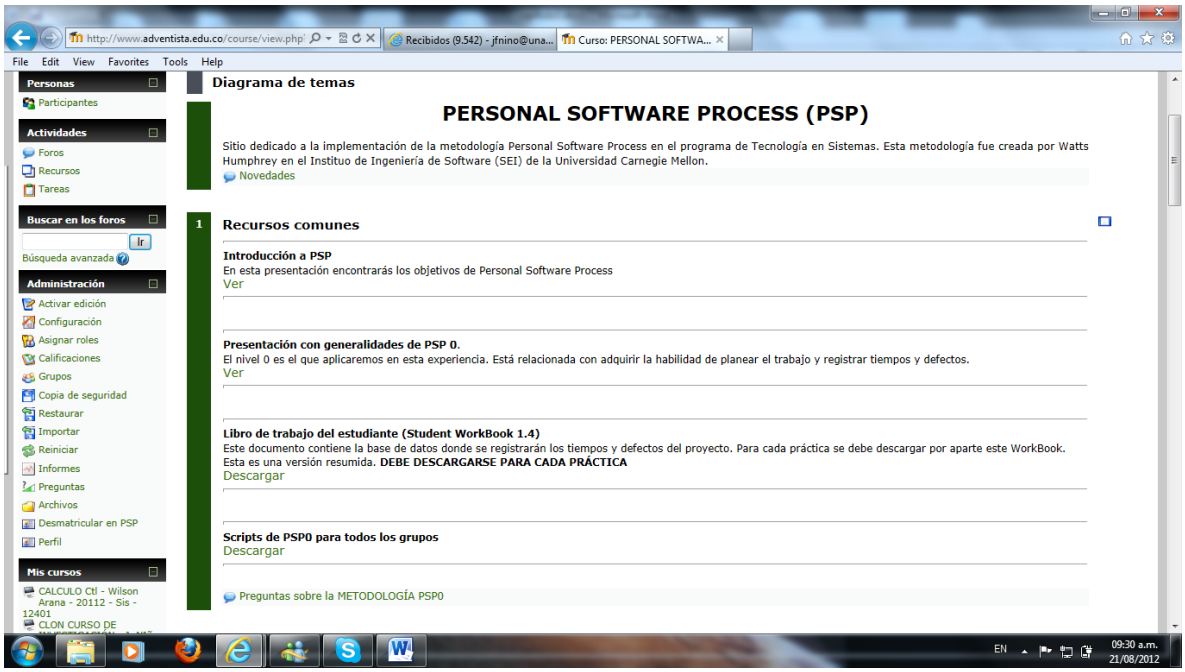


Figura 3.2 Vista de Curso virtual – Sección materiales de capacitación. Fuente: Elaboración propia

Los medios de recolección de las prácticas y registros de tiempos y defectos se diseñaron de forma que fueran fácilmente diligenciados, independientemente del nivel de conocimiento técnico de los estudiantes. Las herramientas usadas fueron las siguientes:

3.3.4 Formato de Planeación y Estimación de tareas (FPET)

Este es un formato de hoja de cálculo en el cual el estudiante debe registrar las tareas estimadas para cada práctica, defectos inyectados estimados y removidos en cada tarea. Tal como se puede observar en el formato, se prestablecen las fases y actividades descritas en el script PSP 0.

A través de este formato cada estudiante registró también las tareas al interior de cada actividad de acuerdo a lo que consideró desde su experiencia personal que debían ser realizadas para cada una de estas actividades.

3	Nombre:	Melissa Pérez Cadavid						Son tomados de la BD	
4	Práctica 1								
5	**Fase**	**Actividad**	**Tareas**	**Tiempo estimado (min)**	**Tiempo real (min)**	**Defectos inyectados estimados**	**Defectos inyectados reales**	**Defectos removidos estimados**	**Defectos removidos reales**
6	**Planeación (PLAN)**		Definición de tareas	10	9,4				
7			Estimación de tiempos	5	3,7				
8			Definición de requisitos	50	44,2		1		
9	**Desarrollo**	**Diseño (DLD)**	Diseñar las preguntas	20	52,6				
10			Hacer Diagrama de clases	50	63,6	5	1	5	
11		**Codificar (CODE)**	Codificación de clases	90	148,5	3	3	3	4
12			Comentarios clases	15	5,2				
13			Codificación clase prinicipal	200	156,6	5	6	5	4
14			Comentarios clase principal	15	9,9				
15		**Compilar (COMPILE)**	Depuración Funcional	35	197,9	2	2	2	5
16			Compilación	170	183,6	5		5	
17		**Probar (UT)**	Prueba funcional	50	59,8	4		4	
18	**Postmortem (PM)**		Presentación	20					
19				**730**	**935**	**24**	**13**	**24**	**13**
20			**Reales**						
21		**Requisitos totales**	6						
22		**Clases totales**	4						
23		**Líneas de Código**	394						
24									

Figura 3.3 Formato de Planeación y Estimación de tareas (FPET) diligenciado por estudiante de Programación 1. Fuente: Elaboración propia

La única diferencia entre el FPET de Lógica de programación y el resto de cursos es el punto relacionado con el número total de clases, puesto que en Lógica de programación se trabaja la programación estructurada; la POO se trabaja a partir de Programación 1.

3.3.5 Aplicativo para registro de tiempos y defectos del SEI en Access (Student Workbook.mde)

- Este aplicativo está contenido en el paquete PSP Student Workbook preparado por el SEI. Dicho aplicativo tiene las siguientes secciones principales:
- Student Profile: en esta sección el estudiante consigna sus datos principales, especialmente los relacionados con su trasfondo técnico en programación.
- Project Directory: en esta sección se crea el proyecto nuevo especificando que se usará el proceso PSP 0.
- Plan Summary: en esta sección el estudiante registra el tiempo estimado de duración de su proyecto y allí mismo podrá ver los tiempos, defectos inyectados y defectos removidos en cada fase a medida que va siendo usada la aplicación en el desarrollo de su práctica.
- Time Recording Log: en este formulario el estudiante va registrando cada una de las tareas que va ejecutando. La aplicación va registrando el tiempo automáticamente. Estos tiempos se van viendo reflejados en el Plan Summary.

- Defects Recording Log: en este formulario se registran los defectos inyectados, su naturaleza y descripción, entre otros aspectos.
- Encuestas aplicables al final de cada práctica y una encuesta general al final de la experiencia. Ambos tipos de encuestas son formatos de hoja de cálculo desarrolladas por el investigador.

4. DEFINICIÓN DEL ESTUDIO EXPERIMENTAL

4.1 OBJETIVO GENERAL DEL ESTUDIO

Evaluar el impacto de aplicación de PSP 0 en la formación de competencias de gestión de proyectos y calidad en estudiantes de tecnología. Este objetivo se puede redefinir siguiendo la estructura general del enfoque GQM [25]:

Analizar la aplicación de metodología PSP nivel 0 *con el propósito de* evaluar su pertinencia de aplicación de manera transversal *con respecto* al desarrollo de competencias de gestión y control *desde el punto de vista de* docentes e investigadores *en el contexto de* estudiantes del programa de Tecnología en Sistemas de la Corporación Universitaria Adventista.

4.2 OBJETIVOS ESPECÍFICOS DEL ESTUDIO

- OE1: Determinar la mejora en las habilidades de planeación expresadas a través de la definición de fases, actividades y tareas con el nivel de granularidad adecuado, así como la estimación de su duración.
- OE2: Determinar la mejora en las habilidades de controlar el avance de un proyecto, expresadas a través del registro del tiempo real invertido en cada tarea planeada.
- OE3: Determinar la mejora en las habilidades de evaluar la calidad del software, expresadas a través de la capacidad para el reconocimiento, categorización y corrección de defectos de los artefactos generados en las diferentes fases del proyecto.
- OE4: Determinar los tipos de errores más frecuentes en los estudiantes como estrategia para tomar acciones correctivas y de acompañamiento más asertivas.
- OE5: Determinar la percepción por parte de los estudiantes sobre la metodología aplicada
- OE6: Determinar la viabilidad de implementar la metodología PSP al interior del currículo de manera transversal e incremental
- OE7: Determinar relación entre aplicación de PSP y la precisión en la estimación de tiempo y la calidad del trabajo final en términos de las notas obtenidas en las prácticas

4.3 PREGUNTAS DE INVESTIGACIÓN Y MÉTRICAS ASOCIADAS

Por medio del estudio, se buscó medir las mejoras en competencias relacionadas con la gestión y control, de igual manera, se buscó medir las percepciones de los estudiantes en cuanto a la metodología. En términos generales, las macro competencias mencionadas son las siguientes:

- Planeación de actividades
- Control de avance
- Reconocimiento y corrección de defectos

Se plantearon 7 preguntas de investigación con sus métricas correspondientes, a partir de las cuales se midió la mejora en las competencias mencionadas previamente, el impacto de la metodología en la calidad de los trabajos entregados y la viabilidad de aplicar nuevamente dicha metodología.

La matriz de correspondencia entre las preguntas de investigación y los objetivos específicos que se buscan en este estudio es la siguiente:

Objetivo	Pregunta de investigación	Métrica(s) asociada(s) y/o Resultado
OE1	RQ1	PPA, PET
OE2	RQ2	PTC
OE3	RQ3	ECD, PCD, Recurrencia de errores
OE4	RQ4	Frecuencia en el tipo de errores hallados por el docente
OE5	RQ5	Frecuencia de beneficios Frecuencia de problemas
OE6	RQ6	Viabilidad de aplicar en un futuro la metodología
OE7	RQ7	Porcentaje de desfase entre tiempo estimado y tiempo real Relación entre nota obtenida y porcentaje de desfase entre Tiempo estimado y Tiempo real

Tabla 4.1 Relación Objetivos – Preguntas de investigación y métricas. Fuente: Elaboración propia

Las preguntas y sus métricas son las siguientes:

4.3.1 RQ1

¿En qué medida la adopción de la metodología PSP permite mejorar las habilidades de gestión de un proyecto en términos de definir tareas adecuadamente granulares y estimar el tiempo necesario para desarrollarlas?

De acuerdo a la metodología PSP, el desarrollador debe ser capaz de atomizar de manera adecuada las tareas necesarias para desarrollar un proyecto. Dicha atomización debe facilitar también la estimación de tiempo para la ejecución de las mismas. Para medir el desarrollo de dichas competencias, se plantean dos métricas: PPA y PET.

- Porcentaje de Precisión en la Atomización (PPA).

Se calcula a partir de la siguiente ecuación:

$$PPA = (NTPE * 100) / NTED$$

PPA: porcentaje de precisión de atomización (/100)
NTPE: número de tareas planeadas por el estudiante
NTED: número total de tareas establecidas por el docente como ideales
El valor ideal a obtener es 100.

Los valores para esta métrica se obtuvieron del formato FPET que cada estudiante diligenció en cada una de las prácticas realizadas y las tareas especificadas por el docente como las ideales para atomizar de manera adecuada el proyecto. Por ejemplo, la planeación realizada por uno de los docentes para la segunda práctica del curso Programación 3 es la siguiente:

Fase	Actividad	Tareas
Planeación (PLAN)		Hacer la planeación de tareas
		Hacer la estimación de tiempos y defectos
		Definición de los requisitos del proyecto
		Revisión final de los requisitos
Desarrollo	**Diseño (DLD)**	Diseñar las clases a usar
		Definir los atributos y métodos de cada clase (Diagrama de clases)
		Diseñar diagrama de secuencias

		Diseñar el modelo fisico de la BD (Diagrama E-R)
	Codificar (CODE)	Elaborar página de login
		Elaborar página de la encuesta para el profesor
		Elaborar página del perfil administrador
		Elaborar página del perfil soporte
		Elaborar controlador (servlet) para el login de usuarios
		Elaborar método que valide el ingreso del usuario
		Elaborar método para activar la cuenta del administrador
		Elaborar controlador (servlet) para almacenar las encuestas diligenciadas
		Elaborar controlador (Servlet) para las consultas del administrador
		Elaborar método que consulta el % de profesores adventistas o no
		Elaborar método que consulta el % de profesores adventistas o no por facultad
		Elaborar página de resultados de la encuesta
		Elaborar página de error
	Compilar (COMPILE)	Revisar página de login
		Revisar página de la encuesta para el profesor
		Revisar página del perfil administrador
		Revisar página del perfil soporte
		Revisar controlador (servlet) para el login de usuarios
		Revisar método que valide el ingreso del usuario
		Revisar método para activar la cuenta del administrador
		Revisar controlador (servlet) para almacenar las encuestas diligenciadas
		Revisar controlador (Servlet) para las consultas del administrador
		Revisar método que consulta el % de profesores adventistas o no
		Revisar método que consulta el % de profesores adventistas o no por facultad
		Revisar página de resultados de la encuesta
		Revisar página de error
	Probar (UT)	Probar funcionalidades del perfil administrador
		Probar funcionalidades de perfil soporte
		Probar funcionalidades del perfil profesor
		Probar resultados de la encuesta
Postmortem (PM)		Revisión de Log de tiempos y Log de defectos
		Revisión general y encuesta en clase

Tabla 4.2 Planeación ideal del docente para la Práctica 2 del curso C3. Fuente: Elaboración propia

Esta planeación permite revisar la atomización de cada una de las fases. Esta atomización facilitaría que el estudiante estime el tiempo que necesita para completar cada fase y finalmente, todo el proyecto.

Para efectos de aplicación de la métrica se considera el número de tareas especificadas por los estudiantes. Para ello se crea una tabla que sirve de referencia sobre el número de tareas por fases y totales. Por ejemplo, para el curso de Lógica de Programación se definió la siguiente tabla:

LÓGICA DE PROGRAMACIÓN			
PRÁCTICA 1 DOCENTE		PRÁCTICA 2 DOCENTE	
FASE/ACTIV	# TAREAS	FASE/ACTIV	# TAREAS
PLAN	4	PLAN	4
DLD	1	DLD	1
CODE	5	CODE	7
COMPILE	6	COMPILE	8
UT	3	UT	2
PM	1	PM	2
	20		**24**

Tabla 4.3 Número de tareas por fase y práctica del Curso C1. Fuente: Elaboración propia

- Porcentaje de Estimación de Tareas (PET)

Se calcula a partir de la siguiente ecuación:

$$PET = (NTE \times 100) / NTP$$

PET: porcentaje de estimación (/100)
NTE: número de tareas a las que el estudiante le estimó el tiempo
NTP: número total de tareas planeadas por el estudiante
El valor ideal a obtener es 100.

Se busca establecer a través de esta métrica, qué tan acuciosos fueron los estudiantes para diligenciar el FPET en lo que concierne a los tiempos que consideran necesario desde su experiencia para realizar sus tareas. El desarrollador debe hacer el ejercicio de aprender a estimar estos tiempos para poder tener un control más técnico en cuanto a la gestión de sus proyectos. En este caso, se parte de la experiencia como estudiantes de primer o último semestre del programa.

4.3.2 RQ2

¿En qué medida mejoró la disciplina de registrar los tiempos usados en cada tarea?

Para poder establecer qué tan conscientes estaban de la tarea y la fase correspondiente que está realizando, el estudiante debe usar el Student Workbook.mde para cronometrar el tiempo real usado en cada una de las tareas que definió en el FPET. Esta competencia está relacionada con el control de los tiempos y ejecución de las tareas planeadas. La métrica usada para medir es PTC.

- Porcentaje de Tareas Cronometradas (PTC)

Se calcula a partir de las tareas que se cronometraron en el Student Workbook.mde en cada práctica.

$$PTC = (NTC \times 100) / NTPE$$

PTC: Porcentaje de Tareas cronometradas (/100)
NTPE: número de tareas planeadas por el estudiante
NTC: número total de tareas cronometradas por el estudiante en el PSP Student WorkBook.mde
El valor ideal es 100.

Si el estudiante no tuvo en cuenta algunas tareas para ser cronometradas, quiere decir que no tuvo un control claro del avance del proyecto y los tiempos totales del mismo no están ajustados a la realidad.

4.3.3 RQ3

¿En qué medida la adopción de la metodología PSP contribuye a mejorar las habilidades de detección y categorización de defectos, corrección de defectos y menor recurrencia de errores a fin de generar productos de mejor calidad?

A fin de garantizar la calidad de los productos software, es necesario que el desarrollador identifique claramente los defectos en los artefactos resultantes de cada fase del proyecto, los corrija y haga conciencia de mejorar en los aspectos en los cuales tiene falencias en el sentido de inyectar más defectos al producto.

Para medir esta evolución en los estudiantes que participaron en el estudio, se usan las siguientes métricas: ECD, PCD y Recurrencia de errores.

- Eficacia en la Corrección de Defectos (ECD)

$$ECD = E / (E + D)$$

ECD: Eficacia en la Corrección de Defectos

E: número de defectos registrados por el estudiante

D: número de errores encontrados por el docente

El valor ideal es 1

Esta métrica se calcula a partir de los defectos registrados por los estudiantes en el WorkBook.mde en el módulo Defects Recording Log y el número de errores que el docente encontró en el momento de evaluar la práctica.

El informe de errores hallados por el docente en cada práctica tienen la siguiente estructura, ver ejemplo:

Código del Estudiante	Descripción defecto	Tipo de defecto
20111004527	[Código] Case opción off, le sobra una f	Sintaxis
	[Código] No lee el lugar de votación del ciudadano	Función
	[Código] Falta cerrar un If en la opción 2.	Sintaxis
	[Código] Mal hallado el porcentaje de hombres o mujeres	Función
	[Código] Error al imprimir en la opción 3	Sintaxis

Tabla 4.4 Retroalimentación errores hallados por docente en práctica de un estudiante. Fuente: Elaboración propia

Esta evaluación la hace el docente para que el estudiante comprenda cuáles errores tuvo, el tipo de error hallado y la fase en la cual lo inyectó.

- Precisión en la Categorización de Defectos (PCD)

$$PCD = (DBC \times 100) / D$$

PCD: Porcentaje de Precisión en la categorización de defectos

D: número de defectos registrados por el estudiante

DBC: número de defectos bien categorizados

El valor ideal es 100

A partir de los defectos registrados en el módulo Defects Recording Log, se evalúan los comentarios hechos en el registro de cada defecto para determinar si fue bien categorizado. La adecuada categorización le permite al desarrollador establecer planes de mejoramiento una vez pase a implementar niveles superiores de PSP. De igual forma, le sirve para llevar un registro histórico de las fases en las cuales típicamente introduce defectos y la categoría de los mismos. Sin mencionar que le permite al docente ver en qué aspectos hay que hacer refuerzos adicionales en cada estudiante y la tendencia del grupo.

- Recurrencia de errores:

Será hallada a partir de los errores encontrados por el docente en cada práctica. Esta métrica permite revisar la frecuencia de las categorías de defectos definidos por PSP en cada práctica. Se observará el comportamiento en la frecuencia por tipos de defectos a fin de determinar si hubo mejoras en la detección y eliminación de defectos por categoría. Se hace un comparativo entre prácticas a nivel general del estudio.

4.3.4 RQ4

¿Cuáles son los tipos de errores más frecuentes de los estudiantes en las prácticas?

Al igual que en el punto de Recurrencia de errores, los datos para responder a esta pregunta de investigación son tomados de la corrección de las prácticas hechas por el docente en cada práctica. Se hará un comparativo por cada uno de los cuatro grupos.

4.3.5 RQ5

¿Cuáles son los principales beneficios y problemas encontrados por los estudiantes en la metodología aplicada?

Las encuestas específicas que debían ser contestadas por los estudiantes al final de cada práctica, cubrían los siguientes ítems:

- La aplicación de los Scritps de PSP 0 fue sencilla
- El instrumento de Planeación y estimación (FPET) es fácilmente diligenciable.
- El instrumento PSP Student WorkBook.mde es de fácil uso
- La aplicación de la metodología PSP 0 en general es fácil
- La categorización de los defectos es sencilla

Para las respuestas se utilizó la escala de Likert, la cual tiene los siguientes valores:

- Totalmente de acuerdo
- Parcialmente de acuerdo
- Ni de acuerdo Ni en desacuerdo
- Parcialmente en desacuerdo
- Totalmente en desacuerdo

Adicionalmente a los ítems mencionados, se incluyeron dos preguntas abiertas:

- Enumere las principales utilidades que le pudo encontrar a la metodología PSP
- Enumere los principales problemas encontrados al aplicar la metodología PSP

A partir de las dos últimas preguntas abiertas:

- Enumere las principales utilidades que le pudo encontrar a la metodología PSP
- Enumere los principales problemas encontrados al aplicar la metodología PSP

Se hace una categorización del tipo de beneficios y problemas a nivel de cada grupo y en general.
Se hace un cuadro comparativo a partir de las respuestas encontradas en cada una de las prácticas, a fin de comprender de qué manera los beneficios fueron más fácilmente identificados por los estudiantes, tanto como la disminución en los problemas encontrados.

4.3.6 RQ6

¿Cuáles son las posibilidades de aplicar PSP en futuros cursos del programa de Tecnología en Sistemas?

En la última práctica los estudiantes contestaron una encuesta adicional que busca evaluar la prueba piloto en varios aspectos, entre los cuales se evalúa la viabilidad de aplicar la metodología en los cursos del área técnica del programa de Tecnología en Sistemas; de igual forma evaluar qué mejoras deben hacerse para su posterior aplicación en aspectos como la capacitación, instrumentos de recolección de datos, tiempo para las prácticas, entre otros. Las preguntas relacionadas con este aspecto son las siguientes:

- La aplicación de PSP me ayudó a no cometer los mismos errores de una práctica a la otra.
- Estimar de una manera más adecuada
- Aplicar un proceso para desarrollar un programa
- Recomendaría esta metodología para cursos futuros

- Los instrumentos y material de estudio de PSP entregados fueron adecuados
- La capacitación dada sobre la metodología fue adecuada
- Aplicaría esta metodología en mi vida laboral.
- El tiempo dado para la realización de las prácticas fue adecuado
- El acompañamiento del docente en cuanto a la metodología fue idóneo

La escala de Likert usada es la siguiente:

- Totalmente de acuerdo
- Parcialmente de acuerdo
- Ni de acuerdo Ni en desacuerdo
- Parcialmente en desacuerdo
- Totalmente en desacuerdo

Igualmente, se hicieron dos preguntas directas en las encuestas realizadas al final de cada práctica:

- Me sentí motivado para aplicar la metodología
- Estoy interesado en aplicar PSP en proyectos futuros.

La escala usada es igual a la descrita anteriormente.
Adicionalmente se dispuso de un espacio para observaciones por parte de los alumnos.
Los formatos de encuestas pueden ser revisadas en el Anexo2: Formatos_Encuestas.

4.3.7 RQ7

¿De qué manera impactó la aplicación de la metodología en la estimación de tiempos y la calidad del trabajo final?

- Porcentaje de desfase entre tiempo estimado y tiempo real

El porcentaje de desfase entre el tiempo estimado y el tiempo real usado en cada práctica será obtenido a partir de Resumen de Plan de Proyecto del Student Workbook de cada individuo. Se hará una comparación entre prácticas por cada curso y en general.

- Relación entre nota obtenida y porcentaje de desfase entre Tiempo estimado y Tiempo real

Dicha relación será obtenida a partir de las notas de las prácticas de los estudiantes y el Resumen de Plan de Proyecto del Student Workbook.

5. ANÁLISIS DE LOS RESULTADOS

Una vez aplicada la metodología a los cuatro cursos se procedió a analizar los datos recolectados en cada una de las prácticas a través de los formatos de planeación y estimación de tareas, el Student WorkBook, y las encuestas al final de cada práctica y al final de la prueba piloto. Cada uno de estos instrumentos diligenciados fueron entregados a través del campus virtual de la Institución.

5.1 PLANEACIÓN DE PROYECTOS POR MEDIO DE LA DEFINICIÓN DE FASES, ACTIVIDADES Y TAREAS

Los resultados obtenidos en esta área se analizan a partir de los PPA y PET.

5.1.1 Porcentaje de Precisión en la Atomización (PPA)

Los valores ideales establecidos por los docentes para cada práctica se encuentran en la siguiente tabla:

	C1-P1	C1-P2	C2-P1	C2-P2	C3-P1	C3-P2
FASE/ACTIV	**# TAREAS**	**# TAREAS**	**# TAREAS**	**# TAREAS**	**# TAREAS**	**# TAREAS**
PLAN	4	4	4	4	4	4
DLD	1	1	3	2	3	4
CODE	5	7	8	6	9	13
COMPILE	6	8	8	6	9	13
UT	3	2	3	2	1	4
PM	1	2	2	2	2	2
Total tareas	**20**	**24**	**28**	**22**	**28**	**40**

Tabla 5.1 Número de tareas ideales por práctica. Fuente: Elaboración propia

Las prácticas 1 y 2 para el curso C4 son las mismas del curso C3.

Se tabuló por cada estudiante el número de tareas por cada fase en cada una de sus prácticas, lo cual permitió comprender en qué fases tienen más problemas los estudiantes en definir las tareas para estimarlas. Estos resultados se comentaron con cada estudiante a manera de retroalimentación personal y privada.

En términos generales se pudo observar una mejoría en los cuatro cursos de la primera a la segunda práctica tal como se refleja en la siguiente figura:

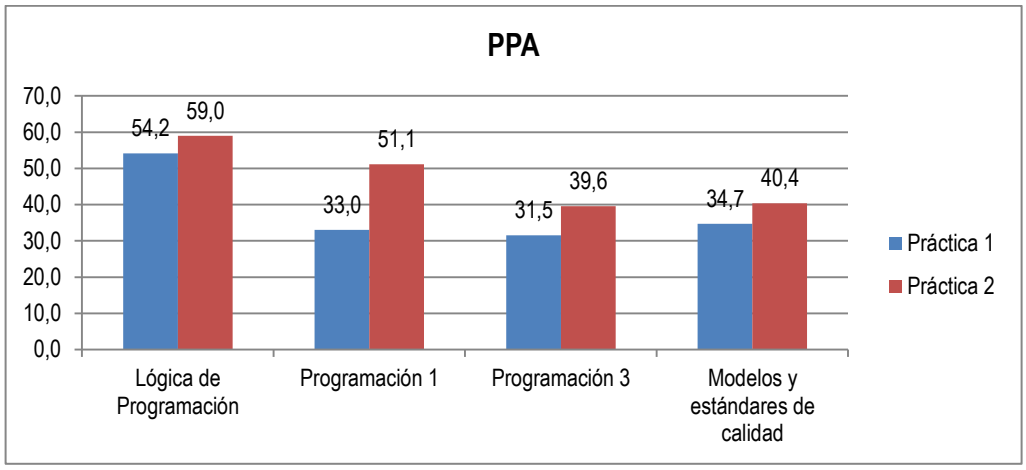


Figura 5.1 Porcentaje de Precisión en la Atomización. Fuente: Elaboración propia

El curso en el cual hubo una mejora más notoria fue el de Programación 1 que pasó de un 33% de precisión a un 51.1%. El curso que menos mejora tuvo fue el de Lógica de Programación, sin embargo estuvo siempre por encima de los demás en este concepto con 54.2% y un 59% de precisión en las prácticas 1 y 2 respectivamente.

También se puede concluir que los cursos más avanzados, Programación 3 y Modelos y estándares de calidad, tuvieron menos precisión en la atomización de tareas, resultado que es afín con una suposición inicial en la cual los estudiantes más adelantados han adquirido malas prácticas en cuanto a lo que significa el proceso de desarrollo de software, asumiendo que abarca solo la programación a pesar de haber recibido cursos de Ingeniería de software 1 y 2, en los cuales se busca la conceptualización del desarrollo de software como un proceso.

En promedio, los PPA de todos los estudiantes fueron de 38.4% y de 47.5% para las prácticas 1 y 2 respectivamente, lo cual indica que hubo una mejora en este aspecto; al final de cada práctica se realizó una retroalimentación (etapa de postmortem) en dónde se recalcó a los estudiantes los beneficios de aprender a atomizar las tareas dentro de cada fase a fin de poder estimar más fácilmente los tiempos y tener claro en qué punto del proyecto se está trabajando en un momento dado; esta retroalimentación contribuyó de manera positiva a mejora la precisión en la definición de actividades.

El desfase en el nivel de atomización de tareas Por fases y grupos generó los siguientes resultados:

- En promedio para la C1-P1 la fase en la que más hubo desfase fue la de Compilación (83%), y las más acertadas fueron Diseño y Prueba unitarias (0%). Para la C1-P2, la fase de Codificación fue la más precisa (9.5% de desfase), y la más desfasada fue Compilación con un 85.4%.

- En promedio para la C2-P1, la fase en la que hubo mayor desfase fue la de Compilación con un 84.4%, y en la que menos desfase se presentó fue en Planeación con un 43.8%. En la C2-P2, el mayor desfase siguió dándose en Compilación con un 70.8%, y Diseño fue la de menos desfase con un 0%.
- En la C3-P1, las fases con mayor desfase fueron Codificación y Compilación con un 79.6% y 88.9% respectivamente; por otro lado la de menor desfase fue Pruebas Unitarias (0%). En la C3-P2, la Compilación tuve el desfase más alto (91%) y Diseño el más bajo con un 8.3%.
- En la C4-P1, la fase de Compilación presentó el mayor desfase con un 88.9%; la fase con menos desfase fue Pruebas unitarias (0%). En la C4-P2, la menos desfasada fue Diseño con un 3.6% y la más desfasada siguió siendo Compilación (84.6%).

En promedio, los desfases para todos los grupos se observa en la tabla siguiente:

Práctica	Planeación	Diseño	Codificación	Compilación	Pruebas unitarias	Postmortem
1	48	30	67	86	17	32
2	38	18	47	83	46	48

Tabla 5.2 Desfases por fase y práctica. Fuente: Elaboración propia

Tanto Compilación como Codificación, se presentan como aquellas fases en las que más imprecisión hubo de parte de los estudiantes al especificar las tareas correspondientes.

Finalmente, se puede concluir que en los cursos iniciales, C1 y C2, se desarrolló mejor la competencia en la atomización de tareas puesto que entre las prácticas 1 y 2, el curso C1 tuvo una mejora de parte del 50% de los estudiantes, mientras en C2 la mejora fue del 100%, es decir, todos los estudiantes mejoraron en la atomización de tareas. Por otra parte, los cursos más avanzados no tuvieron una mejora significativa en este sentido puesto que en C3 Y C4, solo mejoraron el 33.3% y el 42.86% de los estudiantes respectivamente. Estos resultados son consecuentes con la tendencia de que en estudiantes de cursos más avanzados se aprovecha menos o se dificulta más aprender este tipo de metodologías.

5.1.2 Porcentaje de Estimación de Tareas (PET)

En términos generales, los estudiantes fueron disciplinados al estimar el tiempo que gastarían en cada una de las tareas que planearon antes de iniciar cada una de sus prácticas, en promedio los PET de cada práctica fueron de 94.9% y 96.5% respectivamente. El grupo más disciplinado en este sentido fue el de último

semestre: Modelos y estándares de calidad con un 98,2% y 98,6% en las prácticas 1 y 2 respectivamente. Hubo una leve mejora en este indicador de alrededor de 1,76% entre las dos prácticas.

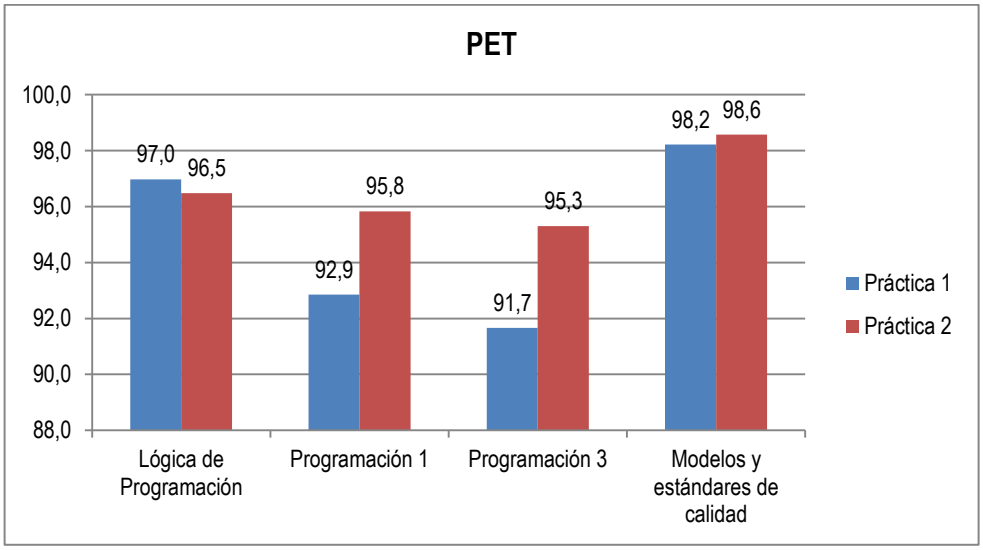


Figura 5.2 Porcentaje de estimación de tareas. Fuente: Elaboración propia

5.2 CONTROL DE AVANCE DE UN PROYECTO

En este aspecto, los cuatro cursos no mostraron mejoras excepto el curso de Programación 1 que tuvo una mejora del 5,81%, tal como se puede observar en la Figura 5.3. En general, los cursos desmejoraron en un 4,10% el control del proceso de desarrollo de software al cronometrar cada una de las tareas planeadas y estimadas, tal como se observa en la siguiente tabla.

Curso	Práctica 1	Práctica 2	% Mejora
C1	97,0	86,2	-11,14
C2	94,5	100,0	5,81
C3	80,6	78,2	-2,88
C4	88,2	81,0	-8,19
Total	90.1	86.3	-4,10

Tabla 5.3 Comportamiento del PTC por prácticas y cursos. Fuente: Elaboración propia

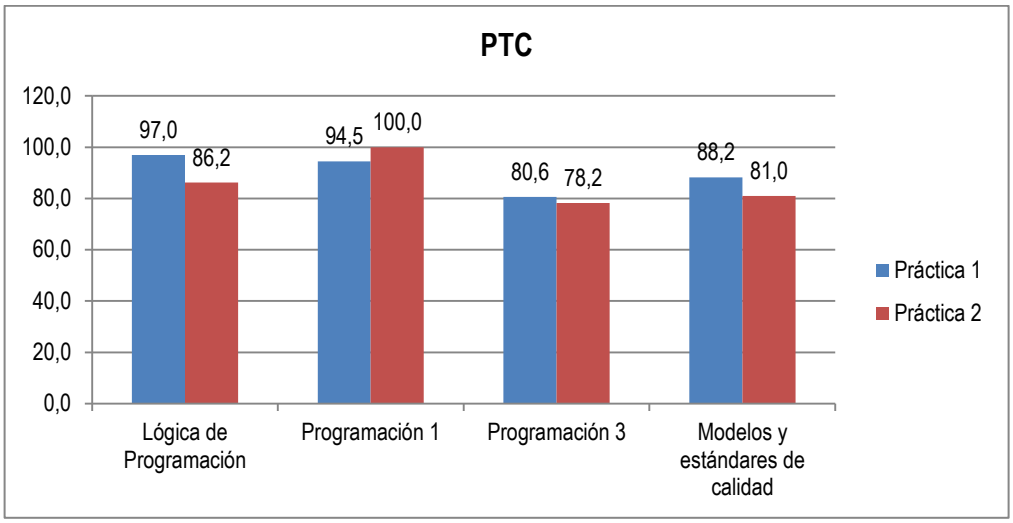


Figura 5.3 PTC por cursos y prácticas. Fuente: Elaboración propia

5.3 RECONOCIMIENTO Y CORRECCIÓN DE DEFECTOS INDIVIDUALES EN PROYECTOS DE SOFTWARE PARA GARANTIZAR LA CALIDAD DEL PRODUCTO

5.3.1 Eficacia en la Corrección de Defectos (ECD)

Según la Figura 5.4, hubo una leve mejora en el índice ECD en los cursos de Programación 1 y Modelos y estándares de Calidad. En el curso de Lógica de programación hubo una fuerte disminución en el índice en cuestión, pasó de 0,5 a 0,33, por lo tanto se les pasaron más defectos a los estudiantes de este curso.

El curso con un mejor indicador fue el de último semestre: Modelos y estándares de calidad.

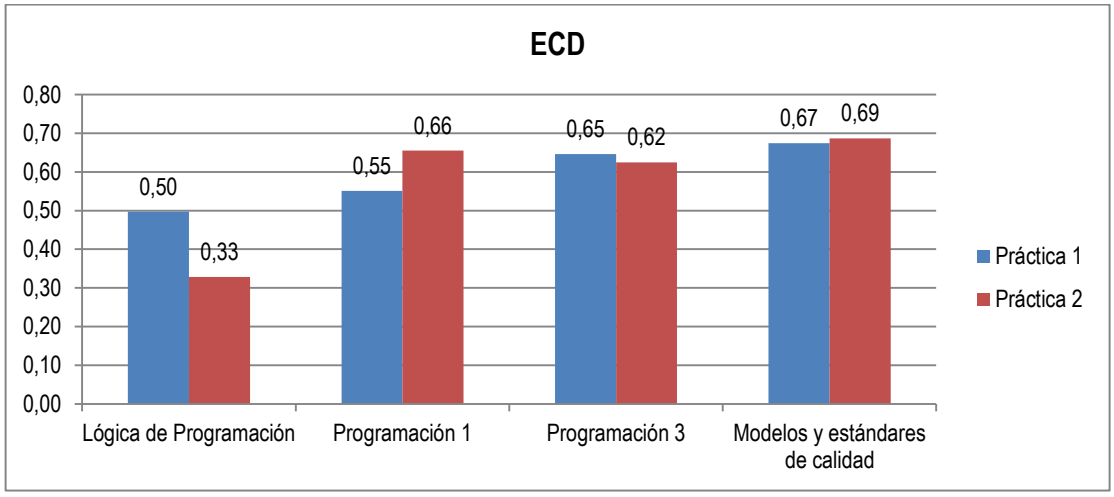


Figura 5.4 ECD por prácticas y cursos. Fuente: Elaboración propia

5.3.2 Precisión en la Categorización de Defectos (PCD)

De acuerdo a la siguiente tabla, todos los grupos mostraron un comportamiento desfavorable en cuanto a la adquisición de esta competencia. En promedio, en la práctica 1 el PCD fue de 46.5% y en la práctica 2 fue de 29.7%, es decir, hubo un descenso en el PCD del 27,1% entre las dos prácticas.

Curso	Práctica 1	Práctica 2	%Mejora
C1	41,94	31,35	-25,26
C2	78,47	27,78	-64,6
C3	32,54	30,76	-5,45
C4	33,33	29,02	-12,93
Promedio	46.57	29.73	-27,06

Tabla 5.4 Comportamiento del PCD por cursos y prácticas. Fuente: Elaboración propia

En términos generales, los grupos con menor precisión en la categorización de defectos fueron los de cuarto y sexto semestre. Estos resultados son consistentes con otros indicadores resultantes de la aplicación de este experimento, pues muestra una dificultad en la determinación del tipo de defecto registrado por los estudiantes. Esto supone la necesidad de mejorar la explicación de los tipos de defectos especificados por PSP o proponer un material de estudio más concreto con otro tipo de categorización.

5.3.3 Tipos de errores detectados por el docente en la evaluación de las prácticas

Los resultados obtenidos en esta experiencia muestran claramente el tipo de problemas que los estudiantes tuvieron en el desarrollo de las dos prácticas: Función, Entorno, Sintaxis, Interfaz, Asignación.

De acuerdo a la Tabla 5.5, el principal tipo de errores encontrados están en la categoría Función, la cual está relacionada con el aspecto de la lógica de los programas desarrollados por los estudiantes. Este tipo de errores constituyen en promedio de todos los cursos y en ambas prácticas el 59,3% de los errores. El siguiente tipo de error encontrado está relacionado con los de tipo Entorno, y más específicamente lo que tiene que ver con aspectos de diseño como especificación de requisitos, diagramas de clases y diagramas de secuencia. Este tipo de errores constituyen en promedio de todos los cursos y en ambas prácticas el 33,7% de los errores. Es importante mencionar que este tipo de errores se notaron más en los estudiantes de últimos semestre puesto que tenía mayor complejidad y más elementos en la parte de diseño.

Resultados como los mencionados previamente, le permiten a la coordinación académica de un programa, especificar estrategias de mejoramiento y de refuerzo en áreas específicas. Generalmente se tiende a calificar las prácticas de los cursos del área técnica a manera de caja negra; la aplicación de PSP le permite al docente y al estudiante tener una idea más clara de los tipos de problemas que se presentan en el desarrollo de las prácticas y tomar medidas correctivas y de mejoramiento.

Tipo de error	Práctica 1		Práctica 2		% Promedio prácticas
	Cantidad de errores	% Tipo de error	Cantidad errores	% Tipo de error	
Función	49,0	63,6	44,0	55,0	59,3
Entorno	22,0	28,6	31,0	38,8	33,7
Sintaxis	6,0	7,8	1,0	1,3	4,5
Interfaz	0,0	0,0	2,0	2,5	1,3
Asignación	0,0	0,0	2,0	2,5	1,3

Tabla 5.5 Porcentajes por tipo de error y práctica. Fuente: Elaboración propia

En la Tabla 5.6 puede observarse que las prácticas que presentaron mayores problemas en errores de tipo Función, fueron las del Curso C1 (Lógica de programación), en contraste con los estudiantes de último semestre (C4); este resultado se ve lógico si se tiene en cuenta que los estudiantes del curso C1 están recién aprendiendo las estructuras lógicas básicas para cualquier programa y el alcance de la práctica tampoco incluía aspectos de interfaz de usuario.

Práctica	Función	Entorno	Sintaxis	Interfaz	Asignación
C1-P1	22	3	4	0	0
C1-P2	26	3	0	2	2
C2-P1	16	1	2	0	0
C2-P2	4	5	1	0	0
C3-P1	6	8	0	0	0
C3-P2	10	10	0	0	0
C4-P1	5	10	0	0	0
C4-P2	4	13	0	0	0
Promedio	**11,6**	**6,6**	**0,9**	**0,3**	**0,3**

Tabla 5.6 Cantidad de errores por práctica y tipo. Fuente: Elaboración propia

En cuanto a los errores de tipo Sintaxis se puede decir que es obvio que se encuentren principalmente en el curso de Lógica de programación (C1) puesto que no tienen forma de compilar en el sentido estricto de la

palabra; esta compilación realmente fue una revisión manual de todo el pseudocódigo. Dicha revisión fue mejor ejecutada hacia la segunda práctica en la cual no se pasaron errores de sintaxis.

El curso C4, de último semestre, mostró el mejor comportamiento de todos en cuanto al número de errores de tipo Función. Dicho comportamiento corresponde, tal como se esperaría, al nivel de conocimiento que tienen estos estudiantes de último semestre.

Es válido hacer una comparación entre los cursos C3 y C4, cuyos estudiantes tuvieron que desarrollar las mismas prácticas (Ver Tabla 5.7). En ambos cursos hubo un aumento en el número de errores de tipo Entorno, los cuales están relacionados con aspectos de diseño. En cuanto a los errores de función (lógica) el desempeño de los estudiantes de último semestre fue mejor, seguramente por la experiencia que ya tienen tal como se mencionara previamente.

	Práctica 1		Práctica 2	
Curso	**Tipo de error**	**Cantidad**	**Tipo de error**	**Cantidad**
C3	Entorno	8	Entorno	10
	Función	6	Función	10
C4	Entorno	10	Entorno	13
	Función	5	Función	4

Tabla 5.7 Comparativo grupos C3 y C4. Fuente: Elaboración propia

5.3.4 Relación entre la nota obtenida y el número de errores detectados por el docente

En las Tablas 5.8 y 5.9, se resumen el comportamiento de las notas respecto del número promedio de defectos registrados por los estudiantes y los errores encontrados por los docentes.

Perfiles	**Rango notas**	**# de defectos todos estudiantes**	**# de errores hallados docentes**	**Promedio defectos**	**Promedio errores**	**% estudiantes**
Bajo	[1.4 - 1.8]	26	14	5,2	2,8	21,7
Medio	[2.1 - 2.9]	13	26	2,6	5,2	21,7
Medio Alto	[3.5 - 4.2]	46	32	7,7	5,3	26,1
Alto	[4.3 - 4.9]	51	11	7,3	1,6	30,4

Tabla 5.8 Notas, defectos y errores Práctica 1. Fuente: Elaboración propia

Perfiles	Rango notas	Número de defectos	# de errores hallados docentes	Promedio defectos	Promedio errores	% estudiantes
Bajo	[0.7 - 1.5]	32	15	5,3	2,5	26,1
Medio	[2.2 - 2.9]	10	25	2,5	6,3	17,4
Medio Alto	[3.0 - 4.1]	52	33	6,5	4,1	34,8
Alto	[4.6 - 5.0]	37	7	7,4	1,4	21,7

Tabla 5.9 Notas, defectos y errores Práctica 2. Fuente: Elaboración propia

En las tablas 5.8 y 5.9, se muestra que en la relación de las notas con los errores encontrados por el docente, se presenta un fenómeno un poco diferente; no se ve una diferencia notable en cuanto al número de errores, sin embargo se puede suponer las prácticas de los niveles Bajo y Medio no tienen tantos errores encontrados porque en muchos casos no había aspectos a calificar simplemente por la inexistencia de un artefacto o varios.

5.4 PERCEPCIÓN DE LOS ESTUDIANTES SOBRE EL USO DE LA METODOLOGÍA

5.4.1 Categorización de los beneficios encontrados por los estudiantes

Los beneficios fueron obtenidos a partir de preguntas abiertas que los estudiantes contestaron en las encuestas al final de cada una de las prácticas.

Los beneficios que los estudiantes manifestaron haber encontrado en la metodología, se categorizaron de la siguiente manera de acuerdo a su naturaleza:

- B1: Detección temprana de errores
- B2: Conocer el rendimiento personal, fortalezas y aspectos a mejorar
- B3: Administrar el tiempo, planeación y el orden del proyecto en general
- B4: Conocer metodologías de calidad
- B5: Estimar con precisión para proyectos futuros

La frecuencia de dichos beneficios se puede ver en la siguiente tabla:

Curso	Práctica 1					Práctica 2				
	B1	B2	B3	B4	B5	B1	B2	B3	B4	B5
C1	0	9	4	2	0	0	3	1	0	0
C2	1	9	5	1	1	1	2	7	1	1
C3	1	3	4	0	0	0	3	2	1	1
C4	1	2	7	0	0	1	1	7	0	1
Totales	**3**	**23**	**20**	**3**	**1**	**2**	**9**	**17**	**2**	**3**

Tabla 5.10 Frecuencia de Beneficios por curso y práctica. Fuente: Elaboración propia

El principal tipo de beneficio encontrado por los estudiantes, de acuerdo a la tabla anterior es el de Administrar el tiempo, planeación y el orden del proyecto en general, con una frecuencia de 20 y 17 en las prácticas 1 y 2 respectivamente.

En la práctica 1 el principal tipo de beneficio encontrado es el relacionado con el Conocimiento del rendimiento personal, fortalezas y aspectos a mejorar con una frecuencia de 23. En la segunda práctica la frecuencia es menor pero sigue siendo el segundo beneficio más importante hallado por los estudiantes participantes del estudio con una frecuencia de 9. Estos hallazgos muestran que los estudiantes fueron conscientes de la importancia de generar una planeación del proyecto y administrar la ejecución del mismo. Esta es precisamente una de las competencias que PSP permite fortalecer en los desarrolladores de software. La detección temprana de errores y el autoconocimiento son otros beneficios encontrados por los participantes del estudio.

En los cursos del primer año, C1 y C2 durante la práctica 1, registraron como el principal beneficio Conocer el rendimiento personal, fortalezas y aspectos a mejorar, en contraste con los cursos avanzados que no le dieron mayor relevancia a dicho aspecto. En cuanto a la Administración del tiempo y la planeación (B2), la importancia dada en todos los cursos, tanto en la práctica 1 y 2, fue importante y homogéneamente distribuida.

En términos de porcentajes, las siguientes figuras permiten visualizar la importancia de los beneficios ya mencionados previamente.

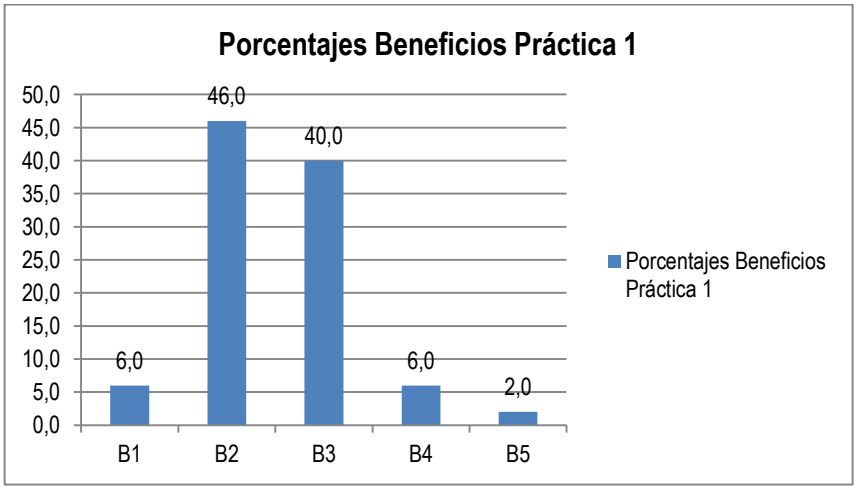


Figura 5.5 Porcentajes de los beneficios hallados en la Práctica 1. Fuente: Elaboración propia

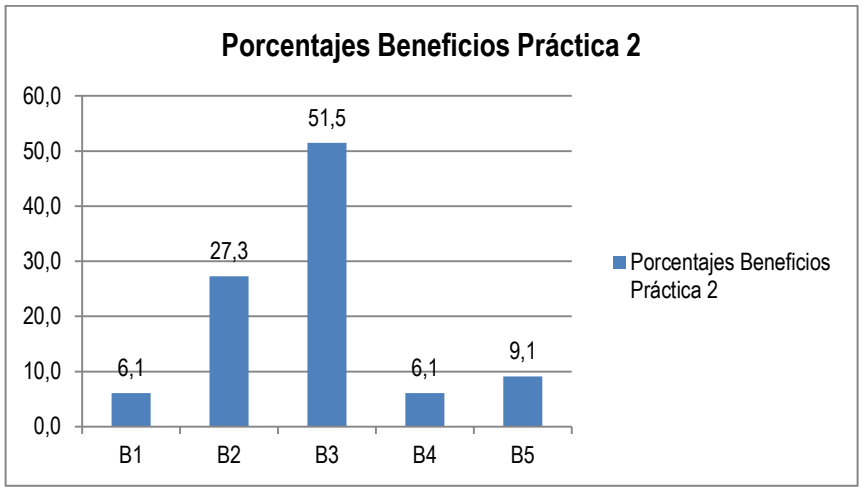


Figura 5.6 Porcentajes de los beneficios hallados en la Práctica 2. Fuente: Elaboración propia

5.4.2 Categorización de los problemas encontrados por los estudiantes

Los problemas fueron obtenidos a partir de preguntas abiertas que los estudiantes contestaron en las encuestas al final de cada una de las prácticas. Dichas respuestas se agruparon según algunas categorías definidas por el investigador.

Los problemas que los estudiantes manifestaron haber encontrado en la metodología, se categorizaron de la siguiente manera de acuerdo a su naturaleza:

- P1: Usabilidad de la Herramienta de registro de tiempos y defectos
- P2: Estimación de tiempos y defectos para cada tarea
- P3: Desconocimiento de la Metodología PSP
- P4: Categorización de defectos
- P5: Incremento de tiempo por la metodología
- P6: Falta de interés
- P7: Desarrollar de acuerdo a lo planeado

La frecuencia de dichos problemas se puede ver en la siguiente tabla:

Curso	Práctica 1							Práctica 2						
	P1	P2	P3	P4	P5	P6	P7	P1	P2	P3	P4	P5	P6	P7
C1	6	1	1	2	0	0	0	3	1	0	1	1	0	0
C2	3	0	2	0	1	1	0	2	1	1	0	0	0	0
C3	2	0	2	1	1	1	0	2	1	0	1	0	1	1
C4	2	0	2	1	1	0	0	2	0	0	2	2	0	1
Totales	**13**	**1**	**7**	**4**	**3**	**2**	**0**	**9**	**3**	**1**	**4**	**3**	**1**	**2**

Tabla 5.11 Frecuencia de problemas hallados. Fuente: Elaboración propia

Los principales problemas mencionados por los estudiantes están relacionados con la usabilidad de la Herramienta de Registro de Tiempos y Defectos (Student WorkBook), sin embargo se puede observar una disminución en la frecuencia de mención de dicho problema pasando de 13 a 9 de la práctica 1 a la práctica 2.

Otros problemas importantes mencionados están relacionados con el Desconocimiento de la Metodología PSP. En esta categoría también hay un descenso en la frecuencia al pasar de 7 a 1 de la primera práctica a la última.

Finamente, un tipo de error frecuente es el de Categorización de defectos cuyo comportamiento se mantiene estable entre la práctica 1 y la práctica 2. Este comportamiento es consistente con el indicador PCD (Porcentaje de precisión de categorización de defectos).

En términos de porcentajes, las siguientes figuras permiten visualizar la importancia de los problemas ya mencionados previamente.

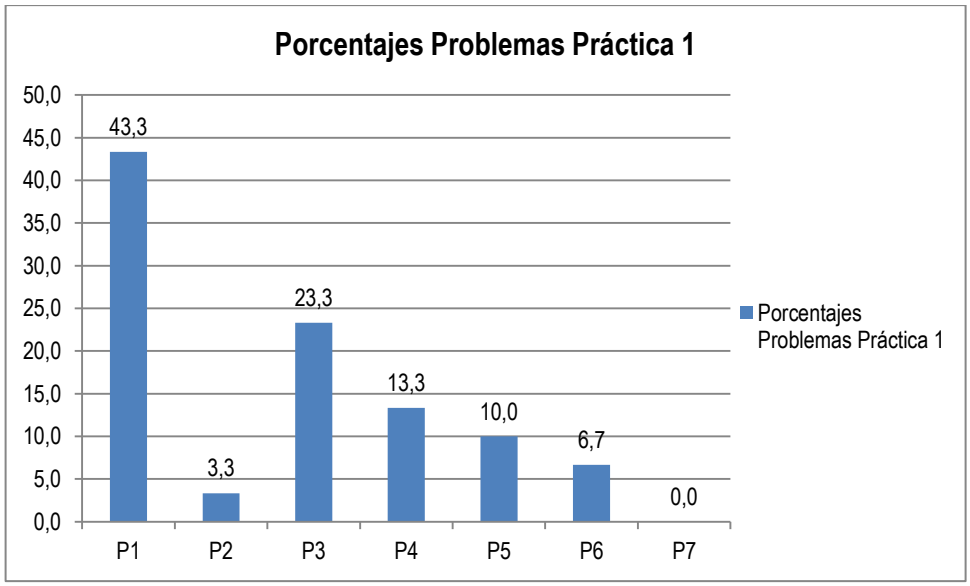


Figura 5.7 Porcentajes de los problemas hallados en la Práctica 1. Fuente: Elaboración propia

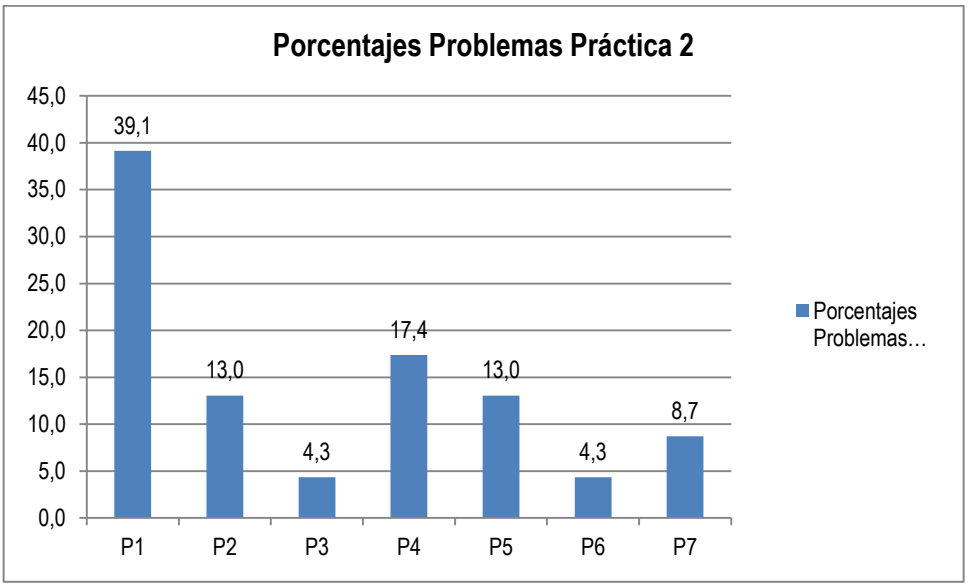


Figura 5.8 Porcentajes de los problemas hallados en la Práctica 2. Fuente: Elaboración propia

5.4.3 Motivación – Disposición a usar de nuevo la metodología

Tanto en la práctica 1 y 2, la mediana ante la pregunta: ¿Me sentí motivado para aplicar la metodología? fue de 4 (Parcialmente de acuerdo). De manera que la motivación de los estudiantes ante esta metodología es altamente favorable. Igual resultado se obtuvo ante la pregunta: ¿Estoy interesado en aplicar PSP en proyectos futuros? En el curso de Programación 3 (cuarto semestre) se notó una mejora tanto en la motivación como en el interés por aplicar nuevamente la metodología. En el curso de Lógica de programación (primer semestre) hubo por el contrario, una disminución en el primer aspecto, pasando su mediana de 5 (Totalmente de acuerdo) a 4 (Parcialmente de acuerdo).

La siguiente tabla muestra el comportamiento de la mediana en cada curso y práctica:

Pregunta	Mediana							
	C1-P1	C1-P2	C2-P1	C2-P2	C3-P1	C3-P2	C4-P1	C4-P2
P21	5	4	5	5	3,5	4	3	4
P22	4,5	4	4,5	5	3,5	4	4	4

Tabla 5.12 Motivación por prácticas y cursos. Fuente: Elaboración propia

5.4.4 Percepción de la metodología aplicada

Se hicieron dos encuestas, una al final de cada práctica. En cada una se hicieron, entre otras las siguientes preguntas:

- P16: La aplicación de los scripts de PSP 0 fue sencilla
- P17: El instrumento de Planeación y estimación.xlsx es fácilmente diligenciable
- P18: El instrumento PSP Student WorkBook.mde es de fácil uso
- P19: La aplicación de la metodología PSP0 en general es fácil
- P20: La categorización de los defectos es sencilla

Algunas de las preguntas y sus resultados pueden ser contrastados con los puntos anteriores. Los resultados para los cuatro grupos en las dos prácticas fueron los siguientes:

- P16: Hubo una mejora en la mediana, pasó de 3 (Ni de acuerdo Ni en desacuerdo) a 4 (Parcialmente de acuerdo), por lo tanto se adquirió una mejor comprensión de los estudiantes de los scripts de PSP adaptados a cada nivel de formación. En primera instancia fue de difícil aceptación puesto que no habían adquirido el hábito de tener un proceso de desarrollo de software personal.
- P17, P18 y P19: La mediana en todos los casos fue de 4 (Parcialmente de acuerdo) en ambas prácticas. No hubo variaciones pero muestra que en estas 3 preguntas hubo aceptación.
- P20: Se obtuvo que la mediana en la primera práctica fue de 3 y de la misma forma en la segunda práctica. En este aspecto no hubo mejora y se mantuvo en el nivel 3 (Ni de acuerdo Ni en desacuerdo). Este resultado es consiste con la métrica PCD vista previamente.

5.5 INFLUENCIA DE LA APLICACIÓN DE LA EXPERIENCIA (CONTRASTE ENTRE PRÁCTICAS)

5.5.1 Reducción del desfase entre tiempo estimado y real

Tal como puede verse en la siguiente figura, los estudiantes del curso C1 tuvieron un incremento en el desfase entre el tiempo estimado y el real, mientras el resto de cursos tuvieron un comportamiento un poco más favorable en este aspecto.

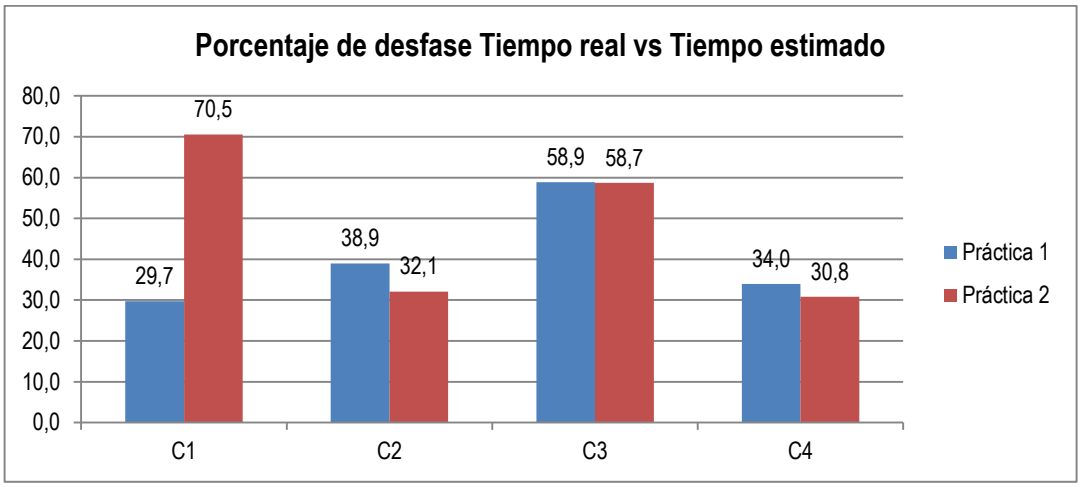


Figura 5.9 . Porcentaje de desfase entre Tiempo estimado y el Tiempo real. Fuente: Elaboración propia

A pesar de que el Porcentaje de Precisión de Atomización (PPA) mostró una mejora entre la prácticas 1 y 2, no es así con el desfase entre tiempo estimado y tiempo real puesto que en la práctica 1 el porcentaje de desfase fue de 40.4% contra un 48% de la práctica 2.

Otro aspecto interesante es que los cursos C3 y C4 que tuvieron que desarrollar las mismas prácticas (especificaciones), tuvieron un comportamiento notoriamente diferente entre ellos; es claro que los estudiantes de último semestre (C4) tuvieron un menor porcentaje de desfase, y adicional a esto tuvo una mayor disminución entre ambas prácticas, pasando de 34% a 30.8%.

En promedio, el grupo C3 fue el de mayor porcentaje de desfase.

5.5.2 Relación entre nota obtenida y porcentaje de desfase entre Tiempo estimado y Tiempo real

En la tabla resumen de la práctica 1, se encontró que las mejores notas las obtuvieron los estudiantes que fueron más precisos en la estimación del tiempo necesario para ejecutar las tareas, y las peores notas los que hicieron lo propio en cuanto a la estimación.

Perfiles	Rango porcentaje desfase tiempo	Promedio notas	Número estudiantes	% Población
Bajo	[2.9 - 5.5]	4,34	5	22,73
Medio	[14.5 - 29.4]	3,36	6	27,27
Medio Alto	[35.9 - 42.9]	3,11	6	27,70
Alto	[50.9 - 92.6]	2,58	5	22,73

Tabla 5.13 Resumen de la práctica 1. Fuente: Elaboración propia

En la Tabla resumen de la práctica 2, se puede observar que las mejores notas las obtienen en los estudiantes cuyo desfase están en el rango medio. Tanto en la práctica

Perfiles	Rango porcentaje desfase tiempo	Promedio notas	Número estudiantes	% Población
Bajo	[12 - 20.9]	2,38	6	28,57
Medio	[24.7 - 27]	3,95	4	19,05
Medio Alto	[32.7 - 39.6]	2,80	7	33,33
Alto	[42.9 - 101.1]	3,60	4	19,05

Tabla 5.14 Resumen de la práctica 1. Fuente: Elaboración propia

Tanto en las prácticas 1 y 2 las mejores notas están en los rangos de desfase medio y bajo, aunque en la práctica 2 la relación no es tan consistente puesto que las peores notas fueron obtenidas en este caso por los estudiantes cuyos desfase fue el más bajo.

6. LIMITACIONES Y AMENAZAS DEL ESTUDIO

La principal limitación encontrada para este estudio fue la disminución en la población por diferentes causas. La población inicial era de 46 estudiantes, sin embargo, la mitad de ellos no fueron tenidos en cuenta en las estadísticas por dos razones principalmente: 7 de ellos cancelaron la materia antes de terminar y el resto de ellos no diligenciaron los instrumentos completamente (Instrumento de estimación, Herramienta de registro de tiempos y defectos, y encuestas) o no entregaron alguna de las prácticas. Estos 23 estudiantes no fueron tenidos en cuenta para evitar incoherencias en los análisis realizados. Sin embargo, en las encuestas realizadas por estos estudiantes "desertores" se pudo determinar, en términos generales, una buena recepción de la metodología especialmente en los cursos del primer año y una resistencia en los estudiantes de último semestre. Es posible que el tiempo asignado de 2 semanas para cada práctica haya influido negativamente en la aplicación de PSP y será tenido en cuenta para futuras experiencias.

Se categorizaron los motivos de no inclusión en la Tabla 6.1:

Motivos para no incluir	C1	C2	C3	C4	Total
Cancelación de materia	2	3	2	0	7
No entregó la práctica 1	0	1	0	1	2
No entregó la práctica 2	1	0	2	0	3
No entregó ninguna práctica	0	0	1	0	1
No contestó las encuestas	0	1	1	5	7
No entregó el FPET de la práctica 2	1	0	0	0	1
No diligenció PSP Student Workbook.mde de práctica 2	0	0	1	0	1
No diligenció ningún PSP Student Workbook.mde	1	0	0	0	1

Tabla 6.1 Motivos de no inclusión de estudiantes en las estadísticas. Fuente: Elaboración propia

En el curso C4 cinco estudiantes no contestaron las encuestas, lo que llevaría a pensar que no se sintieron motivados por la experiencia.

No se trabajó con un grupo de control dado que el objetivo era medir la asimilación de una nueva metodología, y más aún, medir la adquisición de algunas competencias técnicas y básicas relacionadas con el control y la gestión de sus proyectos de desarrollo en estudiantes de diferentes niveles de formación. Para un trabajo futuro se podría trabajar con grupos de control a fin de comparar la mejora en la calidad al aplicar PSP.

Otro tipo de amenaza, es el relacionado con la naturaleza misma de los estudios empíricos con estudiantes de pregrado. En este sentido, este estudio tiene las siguientes amenazas:

- A pesar de haberse diseñado las prácticas a aplicar, dentro de los temas que los estudiantes venían tratando en sus clases para que no significar un esfuerzo adicional fuerte, es posible que por sus múltiples ocupaciones, los estudiantes de último semestre hayan tenido más problema para trabajar de forma adecuada en el estudio. Dichos estudiantes, se encontraban desarrollando su práctica de grado (software a la medida).
- No se creó un formulario de consentimiento escrito en donde los estudiantes fueran informados de la confidencialidad de los datos consignados en los registros de PSP. Sin embargo, se hizo de manera oral.
- Los artefactos utilizados para la descripción de las prácticas no fueron probados previamente, pero han servido de punto de partida para la definición de nuevas prácticas en los programas de Tecnología en Sistemas e Ingeniería de Sistemas de la Institución.

Estas amenazas serán consideradas para siguientes estudios empíricos a aplicar con los estudiantes de los programas académicos mencionados previamente.

7. CONCLUSIONES Y TRABAJOS FUTUROS

Este trabajo ha realizado un estudio experimental para analizar el efecto de aplicar PSP Nivel 0 en estudiantes con diferentes niveles de formación de un programa de tecnología en sistemas, orientado al desarrollo de software.

Los principales aportes de este trabajo son los siguientes:

- Adecuación de los scripts de PSP para los cursos de introducción a la programación que aún no tienen acercamiento a un lenguaje de programación específico.
- Definición de material de apoyo para la aplicación de PSP nivel 0 y la adecuación de la plataforma virtual de la institución (moodle) como espacio de interacción con los estudiantes, para efectos de retroalimentación y de recolección de datos.
- La articulación de las competencias de PSP con el mapa general de competencias del programa académico en cuestión.
- La aplicación de una metodología rigurosa para definir los objetivos, preguntas de investigación y métricas que permitieron evaluar el impacto de la aplicación de PSP nivel 0.
- El esquema de aplicación del estudio de forma simultánea en diferentes cursos, con diferentes niveles de formación.
- Los resultados iniciales de este estudio fueron plasmados en un artículo denominado "Estudio empírico de aplicación de la metodología PSP en estudiantes de un programa de Tecnología en Sistemas con diferentes niveles de formación"; éste fue divulgado a través de una ponencia en el XX Congreso Iberoamericano De Educación Superior en Computación (CIESC), en el marco de la XXXVIII Conferencia Latinoamericana en Informática. Dicho artículo será publicado en la IEEE xplore en el mes de noviembre del presente año.
- Asimismo, este trabajo representa un punto de partida importante para la institución, pues da inicio a una línea de trabajo que tiene el propósito de articular las prácticas de PSP en el programa de Ingeniería de Sistemas de su primera cohorte de manera incremental y a lo largo del programa.

7.1 CONCLUSIONES DEL ESTUDIO

Las principales conclusiones obtenidas a partir de este estudio son las siguientes:

- Los trabajos de aplicación de PSP en general han estado orientados a medir la mejora en el desempeño de los estudiantes. En este estudio se buscó medir otro tipo de competencias a fin de determinar los beneficios de esta metodología, aplicando en primera instancia el nivel 0 de la misma.

Por otra parte, en este estudio se aplica en estudiantes de diferentes niveles de formación a diferencia de otros estudios. Finalmente, este estudio aborda su aplicación en un programa de nivel tecnológico.

- Atendiendo las recomendaciones para aplicación de estudios empíricos en estudiantes, se obtuvieron resultados favorables al aplicarlo como parte de los cursos en cuanto a calificaciones y temas, y no como asignaciones adicionales.
- No se cuentan con cursos orientados a formar las competencias de planeación, control y gestión de la calidad en el proceso de desarrollo de software personal. En algunos casos se podría decir que constituyen temas dictados con una orientación teórica, más no de forma práctica que permita al estudiante interiorizar dichas competencias. La aplicación de PSP contribuye a la formación de las mismas.

Las conclusiones directamente relacionadas con el desarrollo de las competencias de gestión:

- En general se observó que la metodología sirvió para la adquisición de competencias de gestión.
- Los estudiantes de primer año obtuvieron mejores resultados en la precisión de atomización de sus prácticas en comparación con los estudiantes más avanzados, hecho que permite afirmar que es más adecuado adquirir estas competencias en etapas tempranas de la formación.
- En general, los estudiantes fueron disciplinados en la tarea de estimar los tiempos para las tareas definidas.
- El registro de tiempos de las actividades planeadas, como mecanismo para evidenciar el control de avance del proyecto mostró un comportamiento similar en todos los niveles, excepto en los estudiantes de segundo semestre que tuvieron una leve mejora en este aspecto.
- Los estudiantes de último año tuvieron una mejora entre prácticas en la detección de defectos, seguramente por su nivel de experticia
- La categorización de los defectos detectados fue uno de los aspectos en los cuales los estudiantes presentaron mayores problemas independientemente de su nivel de formación. Dicho fenómeno quizás obedezca a un pobre entendimiento de las categorías.
- Los tipos de errores que dejaron pasar los estudiantes en su mayoría fueron de tipo lógico. Es importante resaltar que los estudiantes de primer semestre mostraron una notable mejoría entre prácticas en los tipos de errores Sintaxis.

- La aplicación de la metodología le permite a la dirección académica del programa y a los docentes correspondientes conocer los tipos de errores en los que incurren los estudiantes, y a partir de ellos plantear alternativas educativas para reforzar en los temas o aspectos débiles encontrados.
- En términos generales fueron los estudiantes de primer año quienes encontraron más beneficios que problemas en la aplicación de la metodología. Los dos principales beneficios encontrados fueron los de conocer el rendimiento personal y gestionar los proyectos. Ambas competencias están dentro de las principales que se buscan desarrollar a través de PSP. En cuanto a problemas, el principal fue a nivel instrumental, es decir, la herramienta usada para el registro de tiempos y defectos, sin embargo, en cuanto a la metodología en sí, se menciona como problema pero en un plano muy secundario en comparación con la herramienta.
- Los estudiantes de último año mostraron una leve resistencia a aplicar de nuevo la metodología en un primer momento, sin embargo hacia la segunda práctica mejoró dicha percepción. En cuanto a los estudiantes de primer año, la percepción es favorable a aplicar de nuevo la metodología.
- La reducción de desfase entre el tiempo estimado y el tiempo real usado para el desarrollo de las prácticas mostró un mejor comportamiento en los estudiantes de último año en comparación con los de primer año.
- Se encontró una relación directa entre la precisión en la estimación de tiempos de las tareas y las notas obtenidas. Las mejores notas en ambas prácticas, fueron obtenidas por los estudiantes cuya precisión fue mayor.

A partir de las conclusiones anteriores, se puede decir que la metodología en su nivel 0 es útil para los propósitos de formar a los estudiantes en las competencias relacionadas con gestión de proyectos y calidad, y por lo tanto se infiere que la metodología aplicada completamente será útil y bien recibida por los estudiantes, planteándose la necesidad de que sea aplicada de forma transversal e incremental en el programa académico en cuestión. Las mejoras y adaptaciones para futuras aplicaciones son mencionadas a continuación.

7.2 TRABAJOS FUTUROS

Algunos posibles trabajos futuros a partir de este estudio son los siguientes:

- Crear o adoptar un formato o herramienta de registro de tiempos y defectos más adecuada para los estudiantes de cualquier nivel de formación, integrada a los entornos de desarrollo correspondientes;

de igual forma se espera que dicha herramienta pueda ser usada para llevar un registro histórico de los proyectos realizados por cada uno de los estudiantes desde que inician sus estudios, de manera que éstos puedan ser consultados en cualquier momento por el estudiante, docentes y administradores del programa. Esto permitiría, entre otras cosas, formar equipos de trabajo equilibrados, hacer estimaciones para proyecto futuros y conocer los perfiles de cada uno de los futuros egresados.

- Crear un material de capacitación más detallado con la categorización de defectos propuesta por PSP, puesto que este fue uno de los problemas más evidentes del estudio, de acuerdo a la métrica Precisión en la Categorización de Defectos (PCD).
- Crear scripts de evaluación de cada una de las prácticas para que el docente lo aplique y sirva de retroalimentación para el plan de mejoramiento individual de cada estudiante. De igual forma definir un protocolo para el postmortem al final de cada asignación.
- Ampliar el tiempo de capacitación y mejorar los materiales de estudio adaptados al nivel de formación de los estudiantes.
- Desarrollar un nuevo estudio en el cual el objetivo sea medir la mejora en el desempeño de los estudiantes y en la calidad del producto final, a partir de preguntas de investigación relacionadas usando grupos de control. Dicho estudio está siendo refinado para ser aplicado a grupos de Tecnología en Sistemas y de Ingeniería de Sistemas. El plan es aplicar de manera transversal e incremental a través de los cursos del área de desarrollo de software esta metodología, de manera natural en los programas académicos mencionados.

8. BIBLIOGRAFÍA

[1] A. Brown, M. Oudshoorn y K. Maciunas, *The Personal Software Process in Undergraduate Software Engineering Education,* Australia: Department of Computer Science The University of Adelaide.

[2] W. S. Humphrey, A Discipline for Software Engineering, Addison-Wesley, Enero 1995..

[3] W. S. Humphrey, Introduction to the Personal Software Process, SEI Series in Software Engineering. Addison Wesley, 1997.

[4] V. Basili, G. Caldiera y D. Rombach, «Goal Question Metric Paradigm,» *Encyclopedia of Software Engineering, John Wiley & Sons,* vol. 2, 1994.

[5] V. Basili, C. Caldiera y H. Rombach, «Goal Question Metric Paradigm,» de *Encyclopedia of Software Engineering*, John Wiley & Sons, 1994a, pp. 528-532.

[6] R. Gabay, «Deming's 14 Principles,» [En línea]. Available: http://www.ctrials.org.il/files/BARQA_Article.pdf. [Último acceso: 21 Octubre 2012].

[7] J. Juran y F. Gryna, Juran's Quality Control Handbook, New York: Edition.New York: McGraw-Hill Book Company, 1988.

[8] W. Humphrey, *The Personal Software Process (PSP),* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2000-TR-022, 2000.

[9] M. Pomeroy-Huff, R. Cannon, T. Chick, J. Mullaney y W. Nichols, *The Personal Software Process (PSP) Body of Knowledge, Version 2.0,* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Special Report CMU/SEI-2009-SR-0, 2009.

[10] SEI, «Become an SEI-Certified PSP Developer,» [En línea]. Available: http://www.sei.cmu.edu/certification/process/psp/. [Último acceso: 21 Octubre 2012].

[11] SEI, «Self-Study PSP Material,» [En línea]. Available: http://www.sei.cmu.edu/tsp/tools/student/?location=tertiary-nav&source=5876. [Último acceso: 21 Octubre 2012].

[12] SEI, «PSP Academic Material,» [En línea]. Available: http://www.sei.cmu.edu/tsp/tools/academic/?location=tertiary-nav&source=5876. [Último acceso: 21 Octubre 2012].

[13] P. Ferguson, W. S. Humphrey, S. Khajenoori, S. Macke y A. Matvya, *Results of Applying the Personal Software Process,* 1997.

[14] SEI, *PSP for Eng Public Professor V4.1,* Software Engineering Institute.

[15] W. S. Humphrey, T. A. Chick , W. Nichols y M. Pomeroy-Huff, *Team Software Process (TSP) Body of Knowledge (BOK),* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2010-TR-020, 2010.

[16] J. McHale, T. Chick, N. Davis y G. Miluk, *Accelerating CMMI Adoption with PSP/TSP,* Software Engineering Institute, Carnegie Mellon University, 2007.

[17] J. McHale y D. Wall, *Mapping TSP to CMMI,* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2004-TR-014, 2005.

[18] V. R. Basili, *The Past, Present, and Future of Experimental Software Engineering,* Maryland, USA: ACM-IEEE International Symposium on. Empirical Software Engineering - ISESE, 2007.

[19] G. H. Travassos, S. L. Pfleeger y V. R. Basili, *Experimental Software Engineering:,* 1st Experimental Software Engineering Latin American Workshop -ESELAW., 2004.

[20] D. I. K. Sjøberg, T. Dybå y M. Jørgensen, *The Future of Empirical Methods in Software Engineering Research,* IEEE, 2007.

[21] G. Travassos, D. Gurov y G. Amaral , *Introducción a la ingeniería de software experimental,* Relatório técnico PESC 590/02. COPPE/UFRJ., 2002.

[22] V. R. Basili, *CMSC 735: A Quantitative Approach to Software Management and Engineering,* University of Maryland, 1996.

[23] J. C. Carver, L. Jaccheri, S. Morasca y F. Shull, *A checklist for integrating student empirical studies with research and teaching goals,* Springer, 2009.

[24] J. Carver, L. Jaccheri, S. Morasca y F. Shull, *Issues in Using Students in Empirical Studies in Software Engineering Education,* IEEE, 2003.

[25] R. van Solingen y E. Berghout, The Goal/Question/Metric Method: a Practical Guide for Quality Improvement of Software Development, London: McGraw Hill, 1999.

[26] V. R. Basili, G. Caldiera y H. D. Rombach, *The Goal Question Metric Approach.*

[27] J. A. Calvo-Manzano, G. Cuevas, J. Mejia, M. A. Muñoz y T. San Feliu, «How is CMMI-DEV Applying when Using TSPi Project Planning,» *Electronics, Robotics and Automotive Mechanics Conference (cerma 2009),* pp. 143-148, 2009.

[28] D. Wall, J. McHale y M. Pomeroy-Huff, *Case Study: Accelerating Process Improvement by Integrating the TSP and CMMI,* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University,

Technical Report CMU/SEI-2007-TR-013, 2007.

[29] CMMI Product Team, *CMMI for Development, Version 1.3,* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2010-TR-033, 2010.

[30] K. Cedillo, *Accelerating CMMI Implementation with PSP an TSP in Small Organization,* Ciudad de México, México, 2005.

[31] A. Carleton, J. Over, J. Schwalb, D. Kellogg y T. Chick, *Extending Team Software Process (TSP) to Systems Engineering: A NAVAIR Experience Report,* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2010-TR-008, 2010.

[32] M. Morisio, «Applying the PSP in Industry,» *IEEE Software,* vol. 17, nº 6, pp. 90-95, 2000.

[33] W. Hayes y J. Over, *Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers,* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-97-TR-001, 1997.

[34] L. Hou y J. Tomayko, «Applying the Personal Software Process in CS1: An experiment,» *ACM,* 1998.

[35] M. Towhidnejad y T. Hilburn, «Integrating the Personal Software Process (PSP) across the Undergraduate Curriculum,» *IEEE - Frontiers in Education Conference,* pp. 162-168, 1997.

[36] T. B. Hillburn, *Software Engineering Education: A modest proposal,* Daytona Beach Florida: IEEE Software, 1997.

[37] P. Runeson, «Using students as experiment subjects–an analysis on graduate and freshmen student data,» *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering.–Keele University, UK,* pp. 95-102, 2003.

[38] Chien-Hung Liu 1, Shu-Ling Chen 2 y Chia-Jung Wu 1, *Applying PSP to Support Teaching of Programming Courses,* Taiwan: 1 Department of Computer Science and Information Engineering National Taipei University of Technology, 2 Department of Management and Information Technology Southern Taiwan University, 2008.

[39] K. Venkatasubramanian, S. Bastin Tony Roy y M. V Dasari, *Teaching and Using PSP in a Software Engineering course: An Experience Report,* Rajasthan, India: Computer Science and Information System Group, Birla Institute of Technology and Science Pilani, 2001.

[40] J. I. Maletic, A. Marcus y A. Howald, «Incorporating PSP into a Traditional Software Engineering Course: An Experience Report,» *14th Conference on Software Engineering Education and Training,* p. 89, 2001.

[41] W. I. Bullers, «Personal Software Process in the Database Course,» *Conferences in Research and*

Practice in Information Technology, vol. 30, 2004.

[42] T. San Feliu, M. Dymenstein, A. Etchamendi, F. Maidana y S. Matalonga, *Towards a student oriented approach to teaching PSP discipline,* Montevideo, Uruguay, 2011.

[43] S. K. Lisack, *The Personal Software Process in the Classroom: Student Reactions (An Experience Report),* W. Lafayette: Purdue University, Computer Information Systems Department, 2000.

[44] B. R. von Konsky, J. Ivins y M. Robey, *Using PSPto Evaluate Student Effort in Achieving Learning Outcomes in a Software Engineering Assignment,* Western, Australia: Department of Computing, Curtin University of Technology, 2005.

[45] L. Prechelt y B. Unger, «An Experiment Measuring the Effects of Personal Software Process (PSP) Training,» *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* vol. 27, pp. 465-472, 2000.

[46] G. Rong, H. Zhang, M. Xie y D. Shao, «Improving PSP education by pairing: An empirical study,» *2012 34th International Conference on Software Engineering (ICSE),* pp. 1245-1254, 2012.

[47] J. Börstler, D. Carrington, G. W. Hislop, S. Lisack, K. Olson y L. Williams, «Teaching the PSP: Challenges and Lessons Learned,» *IEEE Software,* vol. 19, nº 5, pp. 42-48, 2002.

[48] «Instructions and Agreement for Downloading the PSP Studio.,» [En línea]. Available: http://csciwww.etsu.edu/psp/dlpsps.html. [Último acceso: 21 Octubre 2012].

[49] Project Web Hosting - Open Source Software, «PSP Time Tracker,» [En línea]. Available: http://psptimetracker.sourceforge.net/. [Último acceso: 20 Octubre 2012].

[50] «The Software Process Dashboard Initiative,» [En línea]. Available: http://www.processdash.com/. [Último acceso: 20 Octubre 2012].

[51] «hackystat,» [En línea]. Available: http://code.google.com/p/hackystat/. [Último acceso: 20 Octubre 2012].

[52] H. Shin, H. Choi y J. Baik, *Jasmine: A PSP Supporting Tool,* Korea: Information and Comunications University, School of Engineering, 2007.

[53] P. Johnson, K. Hongbing, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen y W. Doane, *Beyond the Personal Software Process Metrics collection and analysis for the differently disciplined,* Hawai: Department of Information and Computer Sciences, University of Hawai, 2003.

[54] R. Sison, D. Diaz, E. Lam, D. Navarro y J. Navarro, *Personal Software Process (PSP) Assistant,* Manila, Philippines: College of Computer Studies, De La Salle University, 2005.

[55] Hazrina Hassan, Mohd Hairul Nizam Md. Nasir y Shukor Sanim Mohd Fauzi, *Incorporating Software Agents in Automated Personal Software Process (PSP) Tools,* Kuala Lumpur, Maylasia - Perlis, Maylasia:

Department of Software Engineering, Faculty of Computer Science & Information Technology - Faculty of Information Technology and Quantitative Science, 2009.

[56] J. Lappalainen, *Tool Support for Personal Software Process,* Oulu, Finland: Department of Information Processing Science, University of Oulu, 2005.

[57] C. Ibañez Bernal, «Diseño curricular basado en competencias profesionales: una propuesta desde la psicología interconductual,» *Revista de Educación y Desarrollo,* pp. 45-54, 2007.

[58] W. R. Duncan, *A guide to the Project Management Body of Knowledge,* Newtown Square, PA: Project Management Institute, 1996.

[59] M. Orsted, *Software Development in Microsoft - A subjective view of soft skills required,* Dublin, Ireland: ICSE, 2000.

[60] F. Ahmed, L. F. Capretz y P. Campbell, *Evaluating the Demand for Soft Skills in Software Development,* IEEE Computer Society, 2012.