# Estudio del Proceso Software Personal (PSP) en un entorno académico

**Omar S. Gómez**

**Facultad de Matemáticas, Universidad Autónoma de Yucatán,**
**Anillo Periférico Norte, Tablaje Cat. 13615,**
**Mérida, Yucatán, México**
**omar.gomez@uady.mx**

**Antonio A. Aguileta**

**Facultad de Matemáticas, Universidad Autónoma de Yucatán,**
**Anillo Periférico Norte, Tablaje Cat. 13615,**
**Mérida, Yucatán, México**
**aaguilet@uady.mx**

**Gerzon E. Gómez**

**Facultad de Matemáticas, Universidad Autónoma de Yucatán,**
**Anillo Periférico Norte, Tablaje Cat. 13615,**
**Mérida, Yucatán, México**
**gerzon.gomez@uady.mx**

**Raúl A. Aguilar**

**Facultad de Matemáticas, Universidad Autónoma de Yucatán,**
**Anillo Periférico Norte, Tablaje Cat. 13615,**
**Mérida, Yucatán, México**
**avera@uady.mx**

**Resumen: El Proceso Software Personal (En Inglés, Personal Software Process o PSP) es un proceso individual cuyo objetivo es ayudar a los ingenieros en software a medir y mejorar su productividad personal. El PSP suele enseñarse exclusivamente a profesionales, no obstante éste ha comenzado a incluirse como parte de cursos universitarios. En este trabajo se presenta un estudio sobre PSP efectuado en un entorno académico donde se analizan los siguientes**

**indicadores: precisión en las estimaciones de tamaño y esfuerzo, calidad del producto así como productividad. Los resultados aquí reportados sugieren una mejoría parcial con respecto a la precisión de las estimaciones, una reducción sustantiva en la densidad de defectos de acuerdo a un factor de 6.6 y una productividad media de 53 líneas de código fuente por hora.**

**Palabras clave: ingeniería en software; proceso software personal; PSP; medición del software; proceso de mejora continua.**

## Study of the Personal Software Process (PSP) Under an Academic Setting

**Abstract: The Personal Software Process (PSP) is an individual process where its main goal is to help software engineers to measure and to improve their own productivity. PSP courses usually are given to practitioners; nevertheless it has started to be taught in bachelor's and master's degree programs too. In this work, we present a study of the PSP under an academic setting; where the following elements are analyzed: Size and effort precision estimates, product quality and productivity. Results shown suggest a partial improvement regarding size and effort precision estimates; an important decrease, by a factor of 6.6, with respect of defect density; and an average productivity of 53 source lines of code (SLOC) per hour.**

**Keywords: Software Engineering; Personal Software Process; PSP; Software Measurement; Software Process Improvement.**

# 1. Introducción

Las empresas dedicadas a la construcción de software están en la búsqueda continua de la mejora de sus productos. Un elemento imprescindible en estas empresas son los ingenieros en software, donde cada ingeniero produce una serie de módulos o componentes que son integrados hasta conformar un

sistema software funcional. Si estas empresas desean mantener una alta competitividad, es de suma importancia mejorar la eficiencia, productividad y calidad de los productos software producidos por cada uno de sus ingenieros.

Tomando en cuenta la relevancia del trabajo individual del ingeniero en software, Watts Humphrey desarrolló un proceso de mejora personal siguiendo como referente un modelo para evaluar y mejorar la madurez de las capacidades en organizaciones dedicadas a la construcción de software[1] (Humphrey, 1988, 1989). Humphrey desarrolló este proceso en la década de 1980, en el Instituto de Ingeniería en Software (En Inglés, Software Engineering Institute o SEI). Durante algunos años, Humphrey escribió más de 60 programas donde fue refinando este proceso, posteriormente, comenzó a aplicarlo en grupos de estudiantes adscritos a la Maestría en Ingeniería en Software de la Universidad de Carnegie Mellon. Humphrey llamó a este proceso individual como: Proceso Software Personal[2] (En Inglés, Personal Software Process o PSP) (Humphrey,1995).

Se dice que PSP ayuda a los ingenieros a mejorar en sus estimaciones con respecto al tamaño y esfuerzo requeridos para construir un componente software. PSP también ayuda a los ingenieros a mejorar la calidad de sus productos reduciendo las tasas de inyección de defectos. Diversas compañías quienes han aplicado los principios del PSP han mostrado un aumento de calidad en sus productos software, así como una reducción en los tiempos de desarrollo (Hayes y Over, 1997; Ferguson et al., 1997; Kelly y Culleton, 1999).

Hasta hace poco tiempo el PSP se enseñaba exclusivamente a profesionales, no obstante, éste ha comenzado a impartirse como parte de cursos universitarios en distintas universidades alrededor del Mundo (Ferguson et al., 1997; Prechelt et al., 1997; Carrington et al., 2001; Maletic et al., 2001; Runeson, 2001; Borstler et al., 2002; Abrahamsson y Kautz, 2002a, b; Hou y Tomayko, 1998; Lisack, 2000).

En México, algunas Instituciones de Educación superior del país[3] han comenzado a incluir en sus planes de estudio cursos de PSP, sin embargo, no se han encontrado estudios donde se reporten los resultados de aplicar PSP en instituciones de educación superior mexicanas.

En este trabajo se presentan de manera detallada los resultados del estudio del PSP en un entorno académico de acuerdo a los siguientes indicadores: precisión en las estimaciones de tamaño y esfuerzo, calidad del producto así como productividad. Una versión previa de este trabajo donde se discuten las experiencias de aplicación del PSP se encuentra disponible en Gómez et al. (2014).

Aunque puede cuestionarse que los resultados aquí reportados son de valor limitado para la industria, dado que el estudio se realizó en un entorno académico, diversos autores han observado que el uso de estudiantes universitarios para el estudio de diversos aspectos de la ingeniería en software también son representantes válidos de profesionales en la industria (Jørgensen y Sjøberg, 2001; Höst et al., 2000; Carver et al., 2003; Runeson, 2003; Svahnberg et al., 2008).

El resto del artículo se estructura de la siguiente manera: en la sección 2 se presenta un panorama general del PSP. En la sección 3 se describe el contexto del estudio. En las secciones 4, 5, 6 y 7 se describen los resultados del estudio en términos de la precisión en las estimaciones de tamaño y esfuerzo, calidad del producto así como productividad. En la sección 8 se discuten los resultados encontrados. Por último, en la sección 9 se presentan las conclusiones.

# 2. El Proceso Software Personal

El PSP se desarrolló para ayudar a los ingenieros en software a hacer bien su trabajo. En éste se enseña al ingeniero a aplicar métodos avanzados de ingeniería a sus tareas diarias. Este proceso proporciona métodos detallados de planificación y estimación donde el ingeniero aprende a tener mayor control

sobre su trabajo con respecto a planes que previamente establece así como le ayuda a producir productos de calidad, reduciendo su número de defectos inyectados (Humphrey, 1995).

De acuerdo a los principios de planificación del PSP, para que los ingenieros sean eficaces deben seguir procesos definidos que sean medibles así como planificar su trabajo. Por otra parte, los principios de calidad del PSP promueven que cada ingeniero realice trabajo de calidad. Para alcanzar esta calidad los ingenieros son responsables de la calidad de los productos que producen, previniendo defectos y haciendo su trabajo de manera correcta (Nichols y Salazar, 2009). Estas mejoras se logran a través de la introducción gradual de nuevos elementos o versiones a la línea base del proceso software personal (Abrahamsson y Kautz, 2002b). La progresión de las distintas versiones del PSP se muestra en la Figura 1.

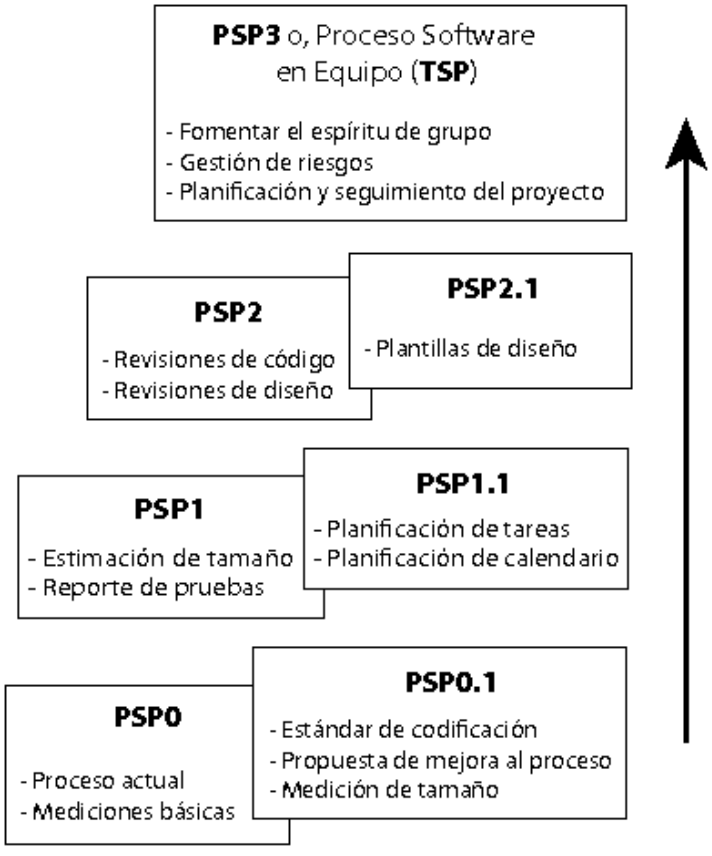


***Figura 1.*** *Versiones del proceso PSP (adaptado de Humphrey [2005])*

Durante un curso de PSP el ingeniero realiza varios ejercicios de programación así como una serie de informes. Este proceso se introduce en seis versiones ascendentes y compatibles. Los participantes inscritos en este curso escriben uno o dos programas en cada versión del proceso, donde recopilan y analizan distintas mediciones. Posteriormente, utilizan las mediciones recolectadas para realizar diversos análisis con el fin de mejorar su desempeño (Nichols y Salazar, 2009).

El ingeniero inscrito en un curso de PSP comienza con la versión inicial del proceso identificada como PSP0, es decir, usa su proceso actual de desarrollo de software y sólo registra sus tiempos y defectos de los programas que realiza. PSP0 se compone de 3 fases que son: planificación, desarrollo (que incluye el diseño, la codificación y la verificación) así como postmortem. También se establece una línea base para llevar a cabo la medición actual del proceso: tiempo dedicado a la programación, fallos inyectados y eliminados, así como el tamaño del programa medido en líneas de código fuente (LOC). La fase de planificación se usa para documentar el plan del producto a construir (ejercicio de programación). En la fase de desarrollo se realizan varias actividades como son: diseño, codificación, compilación y pruebas. En la fase de postmortem se complementa el plan realizado en la primera fase de acuerdo a las mediciones obtenidas tras construir el producto.

PSP0.1 extiende el proceso mediante la adición de un estándar de codificación y de una propuesta de mejora al proceso (En Inglés, Process Improvement Proposal o PIP) que le facilita al ingeniero identificar acciones de mejora para obtener mayor desempeño en su proceso software personal.

En las versiones PSP1 y PSP1.1 (estimación y planificación), de acuerdo a las mediciones recolectadas en PSP0 y PSP0.1, el ingeniero estima el tamaño del producto a construir así como prepara un informe con casos de prueba (PSP1). Las mediciones acumuladas en los ejercicios anteriores se emplean para estimar el tiempo requerido para construir el siguiente producto. En cada producto que

se construye se registran los tiempos de las distintas fases del proceso (planificación, desarrollo y postmortem) así como el tamaño del producto (medido en LOC). Esta información se utiliza para estimar el tamaño y esfuerzo necesarios para construir el siguiente producto. En PSP1.1 se añade a la anterior base del proceso (PSP1) la planificación y calendarización de las actividades definidas en las fases del proceso.

Por otra parte, en PSP2 y PSP2.1 (gestión de calidad y diseño) se añaden dos actividades a la fase de desarrollo: revisión de diseño y revisión de código. La prevención y eliminación de defectos es la parte central de PSP2. En esta versión del proceso, los ingenieros aprenden a evaluar y mejorar sus estimaciones, así como a mejorar la calidad de sus productos. Los ingenieros elaboran y usan listas de comprobación para revisar tanto sus diseños como sus códigos fuente. Finalmente, PSP2.1 introduce técnicas de especificación de diseño y análisis formales con el fin de reducir los defectos inyectados durante el diseño de sus productos.

# 3. Contexto del estudio

Una vez descrita de manera general la estructura del PSP, a continuación se detalla el contexto donde se realizó este estudio.

El estudio se realizó en una de las asignaturas del programa de licenciatura en Ingeniería de Software (IS) ofertado en la Facultad de Matemáticas (FMat) de la Universidad Autónoma de Yucatán (UADY).

El año académico en la UADY consta de dos semestres donde cada semestre suele dividirse en 16 semanas. El estudio se llevó a cabo durante el semestre agosto-diciembre del año 2012.

La FMat-UADY oferta el curso PSP en el programa de licenciatura en (IS) como parte de una serie de cursos optativos que complementan la formación del estudiante. Los cursos optativos se organizan en áreas de concentración, donde

en este caso el curso PSP pertenece al área de concentración definida como mejora del proceso software. Los cursos optativos se imparten en los últimos semestres de la carrera, una vez que el estudiante ha cubierto la mayoría de sus cursos obligatorios. En este estudio, el curso PSP se impartió a estudiantes de séptimo semestre de la carrera.

El curso PSP donde se realizó el estudio se dividió en 16 semanas, donde semanalmente se impartieron dos sesiones (lunes y miércoles) con una duración de dos horas por sesión. Durante el curso los estudiantes desarrollaron ocho programas así como elaboraron dos reportes que entregaron a la mitad y al final del curso. En la Tabla 1 se muestra la descripción de los ejercicios de programación empleados, así como se muestra el promedio de las líneas de código fuente codificadas así como el esfuerzo promedio medido en minutos que les llevó a los estudiantes realizar cada programa.

| Programa | Versión PSP | Descripción | Tamaño (LOC) | Esfuerzo (Mins) |
|---|---|---|---|---|
| 1 | v0 | Cálculo de promedio y desviación estándar de un conjunto de números almacenados en una lista enlazada. | 140.1 | 169.9 |
| 2 | v0.1 | Contador de líneas de código fuente. | 218.14 | 339.71 |
| 3 | v1 | Cálculo de los parámetros de regresión lineal $\beta_0$ y $\beta_1$ así como los coeficientes de correlación $r$ y $r^2$ dado un conjunto de pares de valores. | 249.36 | 283 |
| 4 | v1.1 | Cálculo de rangos, ya sea medidos en LOCs o páginas de acuerdo con los siguientes tamaños relativos: muy pequeño, pequeño, mediano, grande, muy grande. | 231.21 | 189.43 |
| 5 | v2 | Integración numérica de una función empleando la regla de Simpson. | 123.07 | 219.57 |
| 6 | v2.1 | Análisis semántico y sintáctico de un programa que funciona como línea de comandos. | 267.29 | 327.57 |
| 7 | v2.1 | Uso de operaciones sobre una tabla de símbolos, como son: inserción, asignación, búsqueda e impresión en pantalla de la tabla de símbolos. | 285.79 | 261.71 |
| 8 | v2.1 | Uso de operaciones para gestionar un árbol n-ario. | 297 | 274.93 |

***Tabla 1.*** *Listado de los programas usados en este estudio así como el promedio del tamaño (medido en LOC) y promedio del esfuerzo (medido en minutos).*

Como material para la impartición del curso se usaron dos de los libros del autor quien desarrolló este proceso (Humphrey, 1995, 2005). La meta del curso consistió en aprender todo el proceso PSP (hasta la versión 2.1).

De un total de 19 estudiantes inscritos al curso, sólo 14 lo completaron de manera satisfactoria; se tuvo una tasa de deserción del 26%. Los estudiantes inscritos en el curso iniciaban el cuarto año de la carrera (séptimo semestre).

Como herramienta de soporte para la recolección de mediciones se empleó Process Dashboard (Tuma Solutions, 1998). Se decidió en este curso usar una herramienta con el fin de reducir el esfuerzo que conlleva recolectar manualmente las mediciones. Una vez descrito el contexto del estudio en las

siguientes secciones se describen los resultados obtenidos con respecto a la precisión en las estimaciones de tamaño y esfuerzo, calidad del producto así como productividad.

# 4. Resultados con respecto a la precisión en las estimaciones de tamaño

En PSP la estimación de tamaño se realiza con el fin de obtener un estimado preciso del esfuerzo requerido para desarrollar un producto. Para medir el tamaño se emplean LOCs, ya que existe cierta evidencia que sugiere una correlación entre las LOCs y el esfuerzo de desarrollo (Humphrey, 1995, 2005).

Con respecto a esta correlación, en la Tabla 2 se muestra el valor obtenido del coeficiente de correlación[4] *r* y del coeficiente de determinación[5] $r^2$ de cada estudiante que aplicó PSP. Estos coeficientes se calcularon a partir de las mediciones que representan el total de LOCs de los ocho programas, así como el esfuerzo total que le llevó a cada estudiante desarrollar los programas.

| Estudiante | Coeficiente de correlación *r* | Coeficiente de determinación $r^2$ |
|---|---|---|
| 1 | 0.851 | 72.42 |
| 2 | 0.5342 | 28.54 |
| 3 | 0.4396 | 19.32 |
| 4 | 0.7671 | 58.84 |
| 5 | 0.7041 | 49.58 |
| 6 | 0.3713 | 13.79 |
| 7 | 0.3196 | 10.21 |
| 8 | 0.1956 | 3.82 |
| 9 | 0.7076 | 50.07 |
| 10 | 0.3507 | 12.3 |
| 11 | 0.5731 | 32.85 |
| 12 | 0.5176 | 26.79 |
| 13 | 0.4702 | 22.11 |
| 14 | -0.4122 | 16.99 |

***Tabla 2.*** *Correlaciones observadas entre el tamaño y el esfuerzo.*

Por ejemplo, las mediciones de tamaño y esfuerzo del estudiante 1, indican un coeficiente de determinación $r^2$=72.42%, es decir, que el 72.42% de la variación total en el esfuerzo de desarrollo (variable dependiente) puede ser explicada por una relación lineal entre el tamaño (variable independiente) y el esfuerzo. El otro 27.58% de variación restante permanece sin explicar. Como se observa en la Tabla 2, sólo las mediciones de los estudiantes 1, 4, 5 y 9 pueden explicar aproximadamente el 50% o más de variación entre estas dos variables.

En la Figura 2 se muestran las correlaciones de los estudiantes 1, 8, 4 y 14. Como se observa en esta figura, las mediciones de los estudiantes 1 y 4 describen una correlación positiva “fuerte”. En el caso de las mediciones del estudiante 8 no se observa una correlación. Respecto al estudiante 14 sus mediciones indican una correlación negativa.

Continuando con las estimaciones referentes al tamaño, cabe señalar que los estudiantes obtuvieron sus estimaciones a partir de las mediciones que ellos previamente recolectaron de los programas que desarrollaron. Al comienzo del proceso (PSP versión 0.1) las estimaciones tienden a variar de manera considerable, no obstante según evidencia encontrada en Hayes (1998), conforme madura el proceso (PSP versiones 2 y 2.1) esta variación tiende a estabilizarse dentro de un 25% de margen de error, es decir la precisión en las estimaciones aumenta. En la Figura 3 se muestra un diagrama de cajas[6] con la distribución de la precisión de las estimaciones referentes al tamaño de los programas desarrollados (programas 2 a 8 que representan las versiones 0.1 a 2.1 del PSP[7]).

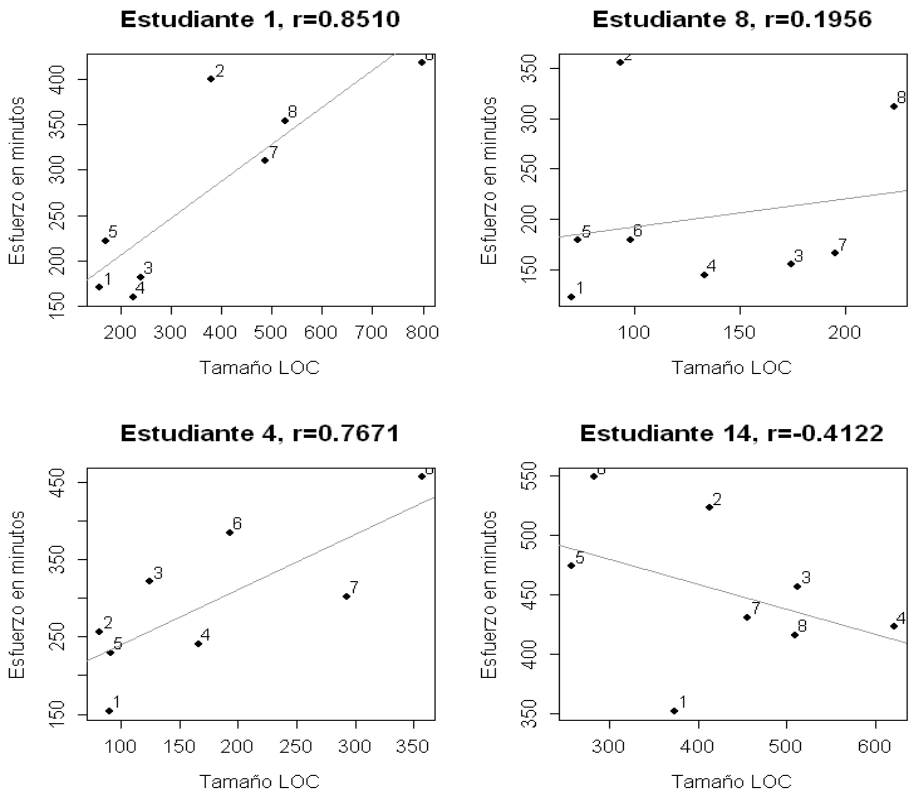


***Figura 2.*** *Gráficos de correlaciones entre tamaño y esfuerzo en cuatro estudiantes.*

Como se observa en la Figura 3, al inicio del proceso los estudiantes tienden a sobrestimar su trabajo, no obstante conforme madura el proceso, las estimaciones tienden a estabilizarse hacia cierto grado de subestimación. Tomando como valor de referencia el 25% de margen de error en las estimaciones, a excepción del programa 2 y 6 se observa que las medianas del resto de los programas se aproximan o están dentro de este valor de referencia.

Respecto a los programas 6, 7 y 8 (versión 2.1 del proceso), el 36%, 50% y 71% de los estudiantes fueron capaces de estimar dentro del 25% de margen de error mencionado en Hayes (1998).

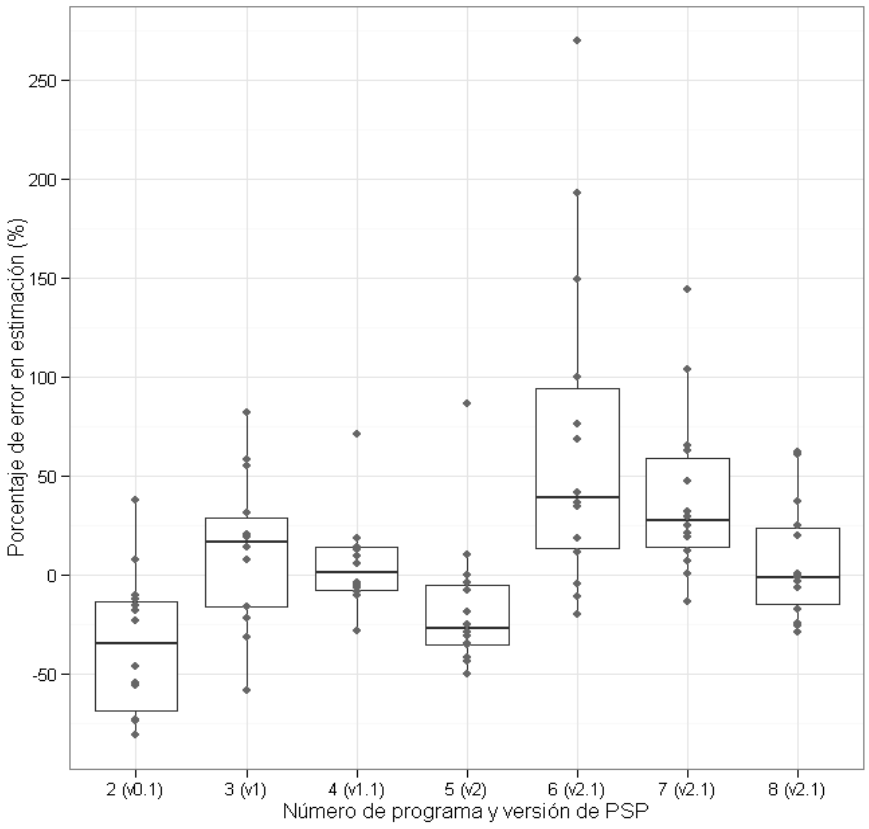


***Figura 3.*** *Precisión en estimaciones con respecto al tamaño.*

Para llevar a cabo el cálculo del tamaño, en PSP se contabilizan diferentes tipos de LOCs como son:

- Líneas de Código Fuente Base. Son aquellas LOCs que no han sufrido modificaciones y que están sujetas a futuros cambios. Por ejemplo, en un programa a desarrollar no habría LOCs base.
- Líneas de Código Añadidas. Son las LOCs nuevas que se añaden a las LOCs base. Por ejemplo, en un programa a desarrollar no habría LOCs base, sólo LOCs añadidas.
- Líneas de Código Modificadas. Es parte de las LOCs base que se han modificado.
- Líneas de Código Borradas. Se refiere a una parte de LOCs base que se han eliminado y que no serán usadas en futuras versiones del programa.

- Líneas de Código Re-utilizadas. Se mantienen sin modificar y son copiadas a un nuevo programa, por ejemplo, funciones, métodos o librerías. Si estás LOCs sufren alguna modificación por mínima que sea, dejan de contarse como re-utilizadas y se añaden a las LOCs base, mientras que las LOCs cambiadas se cuentan como modificadas.

En la Figura 4 se muestra un diagrama de barras con los distintos tipos de LOCs registrados por los estudiantes tanto en la versión inicial como final del PSP. Al inicio del proceso (PSP versión 0.1) la mayor parte de LOCs fueron añadidas (87%), seguidas de LOCs re-utilizadas (12%), sólo un 1% de las LOCs se emplearon como base. Un patrón similar se observa en la versión 2.1 del proceso, donde se observa un 82% de LOCs añadidas y un 8% de LOCs re-utilizadas. En esta versión del proceso el uso de LOCs base se incrementó en un 7%, mientras que se modificó el 2% de LOCs y se eliminó un 1% de éstas.

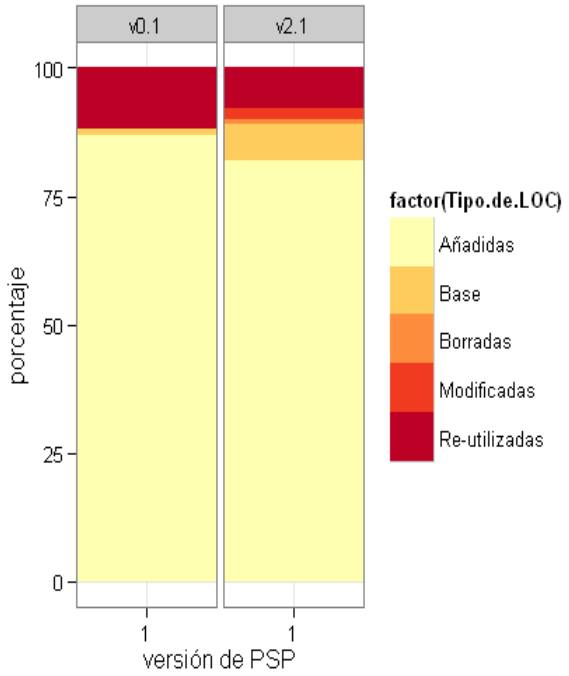


***Figura 4.*** *Tipos de LOCs empleadas en la versión inicial y final del PSP.*

# 5. Resultados con respecto a la precisión en las estimaciones del esfuerzo

PSP usa las estimaciones referentes al tamaño para calcular el esfuerzo estimado en completar un producto. En la Figura 5 se muestra un diagrama de cajas con la distribución de la precisión en las estimaciones afines al esfuerzo. De manera similar que en las estimaciones con respecto al tamaño, se observa que al inicio del proceso los estudiantes tienden a sobrestimar el esfuerzo de construcción del producto, no obstante conforme madura el proceso sus estimaciones tienden a mejorar.

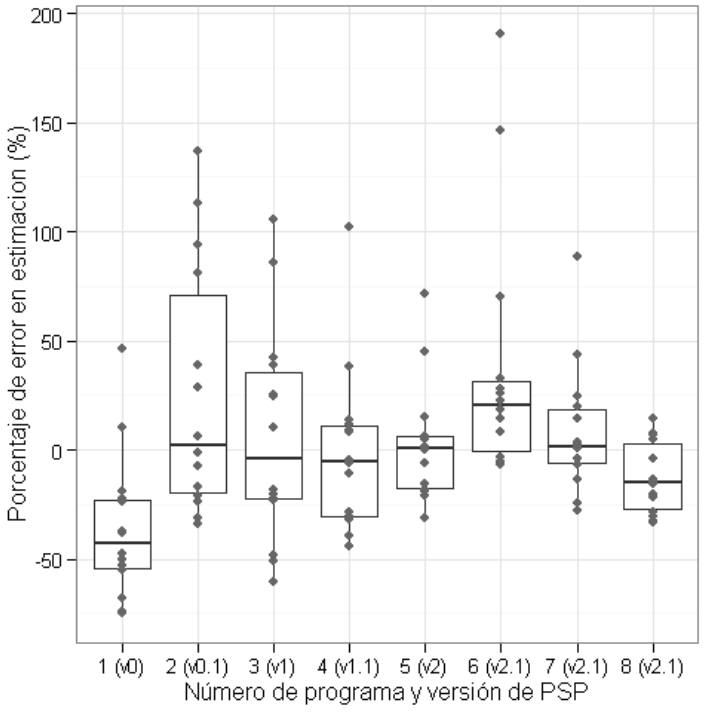


***Figura 5.*** *Error en estimación respecto al esfuerzo.*

Tomando como valor de referencia el 25% de margen de error en las estimaciones, con excepción del programa 1, se observa que las medianas del resto de los programas se encuentran dentro de este valor de referencia.

Con respecto a la versión 2.1 del proceso, el 64%, 79% y 71% de los estudiantes que desarrollaron los programas 6, 7 y 8 (respectivamente) fueron capaces de estimar dentro un 25% de margen de error.

Como se mencionó en la sección 2, cada versión del PSP se conforma de tres fases que son: planificación, desarrollo y postmortem. La fase de planificación se usa para documentar el plan del producto a realizar. En la fase de postmortem se complementa el plan realizado en la fase de planificación de acuerdo a las mediciones obtenidas tras desarrollar el producto. En la fase de Desarrollo se realizan actividades como: diseño, codificación, compilación y pruebas. A partir de la versión 2 se introducen dos nuevas actividades en la fase de desarrollo que son: revisión de diseño y revisión de código.

En la Figura 6 se muestra un diagrama de barras con la distribución del tiempo empleado en las distintas fases y actividades del PSP, tanto en su versión inicial (PSP 0.1) como final (PSP 2.1). Como se observa en esta figura, con respecto a la fase de desarrollo, aproximadamente la mitad del esfuerzo (48%) se usó en codificación, mientras que un 24% se empleó en verificar el producto, un 16% del esfuerzo se usó en el diseño y sólo un 1% en compilación[8]. En las fases de planificación y postmortem se tuvo un esfuerzo del 6% y 4%, respectivamente.

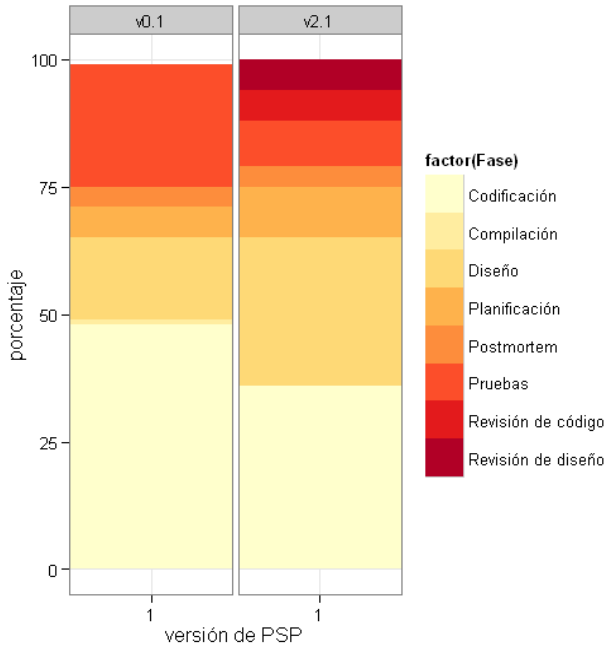


***Figura 6.*** *Esfuerzo empleado en las diferentes fases del proceso.*

Conforme los estudiantes avanzan en su proceso software se añade mayor énfasis en la calidad del producto, tal como se observa en la Figura 6. A diferencia de la versión PSP 0.1, en la versión PSP 2.1 se observa que es mayor el esfuerzo empleado en planificación (10%) y diseño (29%). Por otra parte, se observa una disminución del esfuerzo en codificación (36%) y pruebas (9%). Respecto a las revisiones de diseño y código se observó un esfuerzo del 6% en ambas actividades.

# 6. Resultados con respecto a la calidad del producto

El otro componente principal del PSP es el referente a la calidad, donde se promueve la habilidad de encontrar y remover defectos en etapas tempranas del proceso de desarrollo software. Las revisiones de diseño y de código son actividades que se incorporan en el PSP a partir de la versión 2 del proceso. Al incluir estas actividades en etapas tempranas, se espera que los defectos

encontrados en compilación y pruebas disminuyan de manera considerable, aumentando así la calidad del producto.

En la Figura 7 se muestra un diagrama de cajas con la distribución de los defectos removidos por cada mil LOCs (o KLOCs) durante la actividad de verificación o pruebas. Como se observa en esta figura, los defectos tienden a reducirse conforme madura el proceso.

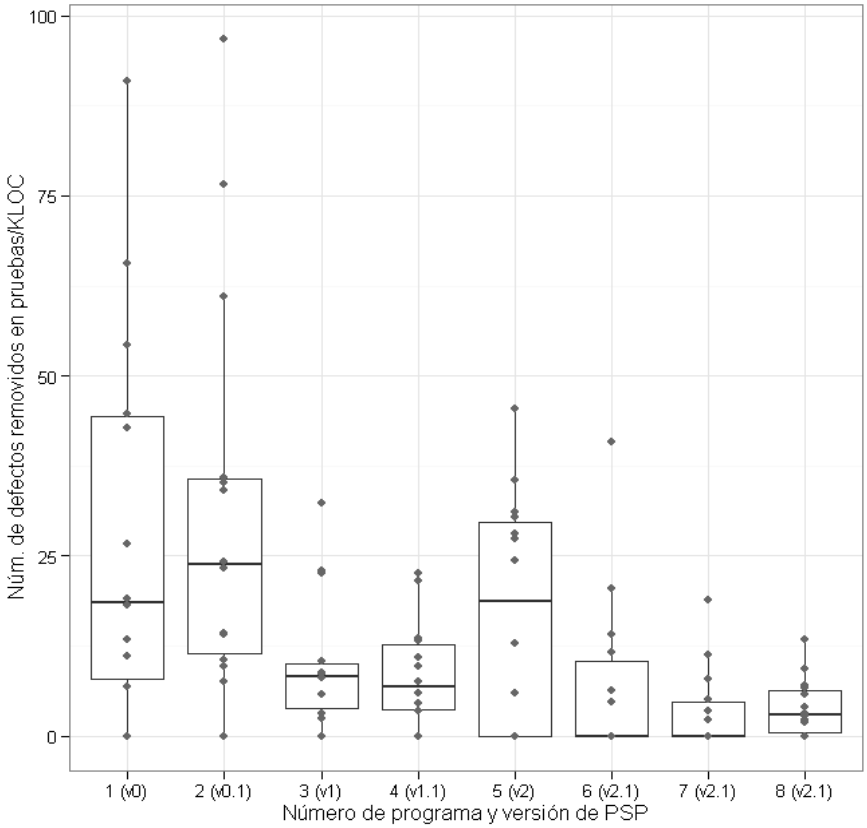


***Figura 7.*** *Defectos removidos por cada mil LOCs en la actividad de pruebas.*

De acuerdo a Humphrey (1995), los defectos pueden clasificarse en una de las siguientes categorías:

- Asignación. Declaración, nombres duplicados, ámbito, límites.
- Construcción. Librerías, control de versión.
- Comprobación. Mensajes de error, verificaciones inadecuadas.
- Datos. Estructura, contenido.
- Documentación. Comentarios, mensajes.
- Entorno. Diseño, compilación, pruebas y otros problemas que soporta el sistema.
- Función. Lógica, punteros, ciclos o bucles, recursión, computación, defectos de función.
- Interfaz. Llamadas a procedimientos y referencias, E/S, formatos de usuario.
- Sintaxis. Ortografía, puntuación, erratas, formato de las instrucciones
- Sistema. Configuración, temporización, memoria.

En total, los estudiantes removieron 886 defectos. En la Figura 8 se muestra la distribución de los distintos tipos de defectos removidos. Los tipos de defectos que más veces se eliminaron fueron aquellos de tipo entorno (273), función (240), datos (126) y sintaxis (115). Por otra parte, los tipos de defectos que menos veces se removieron fueron los de tipo construcción (2) así como defectos de tipo sistema (2).

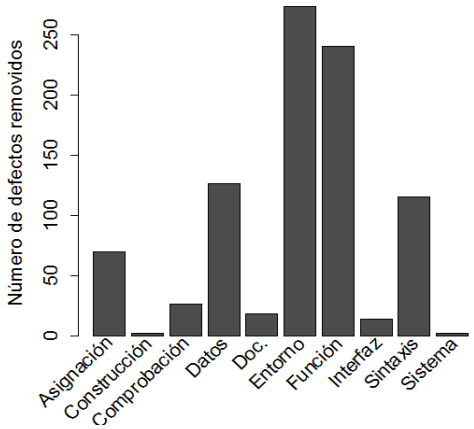


***Figura 8.*** *Frecuencia de los tipos de errores removidos.*

Una de las mediciones importantes del PSP con respecto a la calidad es el porcentaje de remoción de defectos antes de compilación. Esta medición identificada como *yield* se emplea para examinar la calidad del proceso. El *yield* indica qué tan bien funciona el proceso actual para prevenir que los defectos entren en etapas posteriores del proceso. Lo ideal es contar con un *yield* cercano al 100% lo que indica que los defectos se encuentran y remueven antes de las actividades de compilación y pruebas, aumentando de esta manera la calidad del producto resultante.

Se tiene cierta evidencia que sugiere un aumento del *yield* conforme madura el proceso (Hayes y Over, 1997) debido a que se incluyen otras actividades en la fase de desarrollo como son: revisiones al diseño y al código fuente.

En la Figura 9 se muestra un diagrama de cajas con la distribución de yields en cada uno de los programas. Como se observa en esta figura, a partir de la inclusión de las actividades de revisión en la versión 2 el *yield* aumenta de aproximadamente 55% a un 80%, es decir, se observa un incremento favorable de un 25% al añadir las revisiones de diseño y de código.

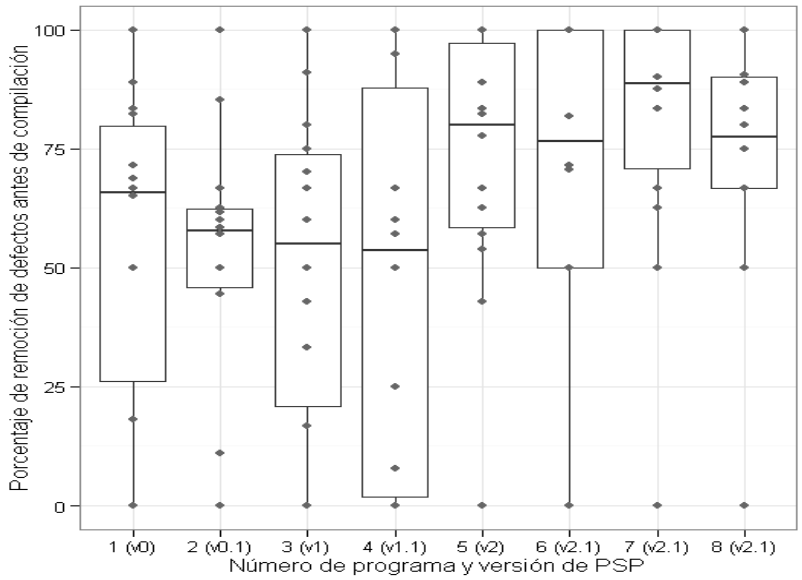


***Figura 9.*** *Porcentaje de remoción de defectos antes de compilación (yield) en los distintos programas.*

# 7. Resultados con respecto a la productividad

A nivel empresarial, la productividad es un indicador relevante para cuantificar los productos o servicios generados en términos del tiempo requerido para producirlos. En PSP la productividad se mide de acuerdo a las LOCs por hora (LOCs/Hora) que produce un ingeniero, siendo esta medición un indicador de la productividad individual. En la Figura 10 se muestra el diagrama de cajas con la distribución de la productividad registrada de los estudiantes en cada programa.

Como se observa en esta figura, las medianas presentan cierto grado de variabilidad en los ocho programas. En el programa 5 se observó la menor productividad (28 LOCs/Hora) mientras que la mayor productividad se presentó en el programa 4 (73 LOCs/Hora). De manera general, se observó una productividad aproximada de 53 LOCs/Hora.

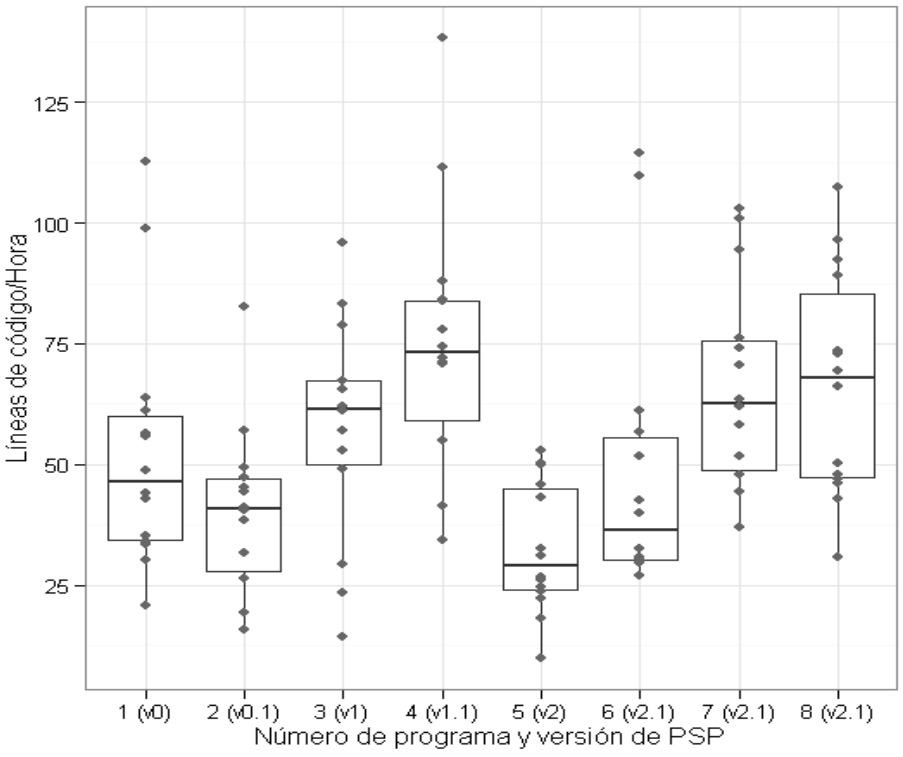


***Figura 10.*** *Productividad observada en los distintos programas*

# 8. Discusión

Los resultados observados con respecto a la precisión en las estimaciones de tamaño y esfuerzo sugieren que cierto porcentaje de estudiantes mejora en sus estimaciones durante las distintas versiones del proceso. A diferencia de las estimaciones con respecto al tamaño, las estimaciones afines al esfuerzo presentan menor variación por lo que tienden a incrementar su precisión conforme madura el proceso. Este aumento de precisión con respecto al esfuerzo se corrobora de acuerdo a los resultados reportados en Hayes (1998).

Se dice que las LOCs correlacionan razonablemente bien con el esfuerzo de desarrollo (Humphrey, 1995), no obstante de acuerdo a los resultados obtenidos esta afirmación se corrobora parcialmente en un 28%, sólo en las mediciones de cuatro de los catorce estudiantes se obtuvieron coeficientes de determinación iguales o superiores al 50%.

Con respecto a la calidad, se observa una disminución en la densidad de defectos en la fase de pruebas de aproximadamente 20 defectos por cada mil LOCs (al inicio del proceso) a aproximadamente 3 defectos por cada mil LOCs. Es decir, se observó un factor de mejora de 6.6. Por su parte, en el estudio realizado por Hayes (1998), el autor reporta un factor de mejora de 2.5, mientras que Abrahamsson y Kautz (2002a) reportan un factor de mejora de 4.2.

De acuerdo al porcentaje de remoción de defectos antes de compilación o *yield*, se observa un incremento conforme madura el proceso. Esta tendencia se corrobora de acuerdo con los resultados reportados en Hayes (1998).

En cuanto a la productividad, se observa una variabilidad en las distintas versiones del proceso, no obstante, no se observa una mejora en la productividad en las distintas versiones del proceso, es decir, no se tuvo una ganancia o pérdida sustantiva de la productividad. Este hallazgo se corrobora de acuerdo a los resultados reportados en Hayes (1998).

# 9. Conclusiones

El PSP es un proceso donde el ingeniero en software aprende a controlar y desarrollar su propio proceso de desarrollo de software. No obstante como se discute en Morisio (2000) y Cannon (1999), el clima organizacional es un factor determinante para que el ingeniero en software continúe aplicando prácticas disciplinadas como las descritas en el PSP.

En este trabajo de investigación se presentaron los resultados obtenidos de estudiar el Proceso Software Personal en un contexto académico. Este estudio se realizó durante el semestre agosto-diciembre 2012 en la Licenciatura en IS de la FMat-UADY.

Los resultados de este estudio indican una ligera mejoría con respecto a las estimaciones de tamaño y esfuerzo. Estos resultados probablemente se deban a que los estudiantes no cuentan con suficiente experiencia codificando programas informáticos. Referente a la calidad del producto, los defectos removidos en la actividad de verificación se redujeron en un factor de 6.6. Se observa que el uso de actividades preventivas como son las revisiones de diseño y de código demuestran ser efectivas.

Para finalizar, a continuación se listan algunas recomendaciones a tener en cuenta en la aplicación del PSP en un entorno académico:

- Se recomienda enseñar PSP en los últimos semestres de la carrera, una vez que el estudiante cuente con cierta experiencia en algún lenguaje de programación.
- Se recomienda que el estudiante cuente con conocimientos de probabilidad e inferencia estadística. Esto con el fin de facilitar el aprendizaje de las diferentes técnicas estadísticas en las que se apoya el PSP.
- Es recomendable que el profesor supervise la fase de planificación en clase para asegurarse que se han registrado de manera correcta las estimaciones.

- Se recomienda que al menos una parte de los ejercicios de programación asignados se trabajen en clase, de tal manera que el profesor pueda supervisar el proceso de desarrollo del estudiante.
- Es recomendable el uso de alguna herramienta que apoye en el proceso de recolección de mediciones. Esto facilita centrarse en el aprendizaje del proceso y no en el cálculo manual de las distintas mediciones del PSP.

# Notas

**[1] A este modelo se le conoce como Modelo de Madurez de Capacidades, en Inglés, Capability Maturity Model o CMM.**

**[2] Otras traducciones en castellano comunes a este término son: proceso de desarrollo de software personal, proceso de software personal y proceso personal de software.**

**[3] Tales como: Universidad Autónoma de Yucatán, el Centro de Investigación en Matemáticas, Instituto Tecnológico y de Estudios Superiores de Monterrey, la Universidad Autónoma de Zacatecas, la Universidad Autónoma de Nuevo León, entre otras.**

**[4] El coeficiente de correlación r mide el grado de asociación y dirección entre dos conjuntos de datos (que generalmente representan dos tipos de variable: dependiente [y] e independiente [x]). El valor de este coeficiente puede variar entre -1.0 a +1.0, donde valores cercanos a ±1.0 indican una correlación linear (positiva o negativa). Por el contrario valores cercanos a 0 indican la ausencia de correlación entre los dos conjuntos de datos. Se dice que una correlación es "fuerte" cuando su valor es mayor o igual que ±0.8. Por el contrario, ésta se describe como "débil" cuando su valor es menor que 0.5 (en el caso de la correlación lineal positiva, o mayor a -0.5 cuando ésta es negativa).**

**[5] El coeficiente de determinación $r^2$ indica la fluctuación de los valores de la variable dependiente que se pueden explicar en una relación lineal con los valores de la variable independiente. Los valores de este coeficiente oscilan entre 0 y 1 (o en caso de representarse como porcentaje, 0% y 100%).**

**[6] Un diagrama de caja es un tipo de gráfico donde se representan los cuartiles de un conjunto ordenado de datos. Este diagrama se compone por un rectángulo o "caja" y dos brazos o "bigotes". La caja representa el 50% de los datos, la línea que divide la caja representa la mediana, mientras que cada**

**brazo representa el 25% de los datos. Este diagrama es útil para identificar valores atípicos así como evaluar la simetría de la distribución de los datos.**

**[7] El primer programa se ha omitido ya que de acuerdo al proceso en éste no se realiza una estimación del tamaño.**

**[8] El esfuerzo empleado en compilación es mínimo ya que los estudiantes trabajaron con un entorno de desarrollo integrado (en Inglés, IDE) que les informa sobre errores de sintaxis que cometen.**

# Referencias


**Abrahamsson, P. and Kautz, K. (2002a). Personal Software Process: Classroom Experiences from Finland. In Kontio, J. and Conradi, R., editors, Software Quality -- ECSQ 2002, volume 2349 of Lecture Notes in Computer Science, pp. 175--185. Springer Berlin Heidelberg.**

**Abrahamsson, P. and Kautz, K. (2002b). The Personal Software Process: Experiences from&nbspDenmark. In Euromicro Conference, 2002. Proceedings. 28th, pp. 367--374.**

**Borstler, J., Carrington, D., Hislop, G., Lisack, S., Olson, K., and Williams, L. (2002). Teaching PSP: challenges and lessons learned. Software, IEEE, 19(5):42--48.**

**Cannon, R. L. (1999). Putting the Personal Software Processsm into practice. Conference on Software Engineering Education and Training, 0:34.**

**Carrington, D., McEniery, B., and Johnston, D. (2001). PSPsm in the large class. In Software Engineering Education and Training, 2001. Proceedings.14th Conference on, pp. 81--88.**

**Carver, J., Jaccheri, L., Morasca, S., and Shull, F. (2003). Issues in using students in empirical studies in software engineering education. In METRICS '03: Proceedings of the 9th International Symposium on Software Metrics, page 239, Washington, DC, USA. IEEE Computer Society.**

**Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S., and Matvya, A. (1997). Results of applying the Personal Software Process. Computer, 30(5):24--31.**

**Gómez, O. S., Gómez G. E., Aguileta, A. A. and Aguilar, R. A. (2014). Proceso de Software Personal en la Academia: Experiencias de Aplicación en México. In Zapata, C. M. and González, G., editors, Ingeniería de Software e Ingeniería del Conocimiento: Dos Disciplinas Interrelacionadas, pp. 197—213. Universidad de Medellín.**

**Hayes, W. (1998). Using a Personal Software Processsm to improve performance. In Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International, pp. 61--71.**

**Hayes, W. and Over, J. (1997). Personal Software Process (PSP): An empirical study of the impact of PSP on individual engineers. Technical Report CMU/SEI-97-TR-001, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.**

**Hou, L. and Tomayko, J. (1998). Applying the Personal Software Process in CS1: An Experiment. In Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, SIGCSE '98, pp. 322--325, New York, NY, USA. ACM.**

**Humphrey, W. (2005). PSP(sm): A Self-improvement process for software engineers. Addison-Wesley Professional, First edition.**

**Humphrey, W. S. (1988). Characterizing the software process: A maturity framework. IEEE Software, 5(2):73--79.**

**Humphrey, W. S. (1989). Managing the Software Process. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.**

**Humphrey, W. S. (1995). A Discipline for Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.**

**Höst, M., Regnell, B., and Wohlin, C. (2000). Using students as subjects -- a comparative study of students and professionals in lead-time impact assessment.Empirical Softw. Engg., 5(3):201--214.**

**Jørgensen, M. and Sjøberg, D. I. K. (2001). Software process improvement and human judgement heuristics. Scand. J. Inf. Syst., 13:99--121.**

**Kelly, D. and Culleton, B. (1999). Process improvement for small organizations. Computer, 32(10):41--47.**

**Lisack, S. (2000). The Personal Software Process in the Classroom: student reactions (an experience report). In Software Engineering Education amp; Training, 2000. Proceedings. 13th Conference on, pp. 169--175.**

**Maletic, J. I., Marcus, A., and Howald, A. (2001). Incorporating PSP into a traditional software engineering course: An experience report. In Proceedings of the 14th Conference on Software Engineering Education and Training, CSEET '01, pp. 89--, Washington, DC, USA. IEEE Computer Society.**

**Morisio, M. (2000). Applying the PSP in industry. Software, IEEE, 17(6):90--95.**

**Nichols, W. and Salazar, R. (2009). Deploying TSP on a national scale: An experience report from pilot projects in Mexico. Technical Report CMU/SEI- 2009-TR-011, Software Engineering Institute, Carnegie Mellon, Pittsburgh, PA.**

**Prechelt, L., Unger, B., and Gramberg, O. (1997). Experience report: Teaching and using the personal software process (PSP). Submission to ESEC'1997.**

**Runeson, P. (2001). Experiences from teaching PSP for freshmen. In Software Engineering Education and Training, 2001. Proceedings. 14th Conference on, pp. 98--107.**

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

**Svahnberg, M., Aurum, A., and Wohlin, C. (2008). Using students as subjects -- an empirical evaluation. In ESEM '08: Proceedings of the Second ACM-IEEE international symposium on**

**Empirical software engineering and measurement, pp. 288--290, New York, NY, USA. ACM.**

**Tuma Solutions, L. (1998-2012). Process dashboard. version 1.15.0.1.**

# Notas biográficas:

**Omar S. Gómez** es Ingeniero en Computación por la Universidad de Guadalajara (México, 2001), Maestro en Ingeniería de Software por el Centro de Investigación en Matemáticas A.C. (México, 2005) y Doctor en Software y Sistemas por la Universidad Politécnica de Madrid (España, 2012). Actualmente es profesor en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán (México). Su área de investigación se desarrolla en la experimentación en Ingeniería en Software, diseño y verificación de software así como en temas de procesos para la mejora continua en el desarrollo de software.

**Antonio A. Aguileta** es Licenciado en Ciencias de la Computación, egresado de la Facultad de Matemáticas (FMat) de la Universidad Autónoma de Yucatán (UADY), Maestro en Ciencias Computacionales por el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), campus Monterrey. Actualmente es coordinador del Sistema de Información y Control Escolar Institucional (SICEI) de la UADY y Profesor de Tiempo Completo en la FMat. El área de investigación de su interés gira en torno a la calidad en la Ingeniería de Software.

**Gerzon E. Gómez** es Ingeniero en Sistemas Computacionales por la Universidad Autónoma de Tamaulipas. Actualmente es profesor en el área de Ingeniería de Software en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán (UADY). Es doctor candidato en Informática por la Universidad Politécnica de Madrid. Colaborador en el grupo de investigación SEMEPRO (Seguridad y Mejora de Proceso) y en el proyecto Cátedra de Mejora de Procesos Software en el Espacio Iberoamericano (MPSEI). Autor de Publicaciones en Revistas Internacionales.

**Raúl A. Aguilar** es Licenciado en Ciencias de la Computación por la Universidad Autónoma de Yucatán, Doctor en Informática por la Universidad Politécnica de Madrid, España. Actualmente es profesor de tiempo completo en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán. Su trabajo de investigación incluye las siguientes áreas: Ingeniería de Software e Informática Educativa.