Desarrollo de software
Semestre 5

Programa de la unidad didáctica:
Métricas de desarrollo de software (PSP)

Unidad 1. Métricas de desarrollo de software (PSP)

Clave:
15143529

Ciudad de México, febrero de 2025

Universidad Abierta y a Distancia de México

Índice

Unidad 1. Métricas de desarrollo de software (PSP)

Presentación de la unidad

En esta unidad ahondarás en las características principales del Proceso Personal de Software (PSP, por sus siglas en inglés), una disciplina diseñada por el ingeniero de software, Watts Humphrey, en los años 90.

Conocerás por qué se creó esta disciplina, sus características, la relación tan estrecha que tiene con el Proceso Grupal de Software (TSP) y el Modelo Integrado de Capacidad y Madurez (CMMI). Aprenderás las generalidades y los tipos de registros y métricas que se llevan, tales como: registros de tiempos, defectos, resumen de plan de proyecto.

Conocer PSP te ayudará, como su nombre lo indica, en tu proceso personal de desarrollo, favoreciendo la optimización de tiempos y asegurando la calidad de tu trabajo. Por ello es importante que pongas especial cuidado en el propósito ya que es el enfoque de la unidad y lo que deberás aprender para poder entender las siguientes unidades.

Logros

En esta unidad lograrás:

- Identificar las características e importancia de PSP para el desarrollo de software.
- Analizar el proceso de líneas base, formas, elementos, procesos, mediciones de PSP para poder registrar tiempos, defectos y el resumen del plan del proyecto.
- Analizar en un programa de software la manera de implementar el proceso PSP 0.

Competencia específica

- Identificar los conceptos básicos de PSP para analizar un programa, considerando la relación de PSP con TSP (*Team Software Process)* y CMMI (*Capability maturity model integration*).

1.1. Antecedentes de PSP y su relación con TSP y CMMI

Antes de que comencemos la unidad didáctica, es importante que conozcas los orígenes del PSP y cómo, en conjunto con TSP y CMMI, forman una disciplina organizacional que te permitirá realizar software en un menor tiempo, optimizar costos y sobre todo con calidad.

El CMMI es un producto desarrollado por el Software Engineering Institute (SEI) y se basa en la mejora continua de los procesos. Pretende incrementar la capacidad administrativa de las organizaciones para controlar los costos, tiempos y productividad en general. Por otra parte, el objetivo de TSP es adiestrar a los ingenieros de software en el desarrollo de equipos y llegar en menos tiempo a niveles altos de desempeño por medio de la práctica del proceso personal del software de cada uno de sus integrantes el cual nace como un acercamiento estructurado y disciplinado para el desarrollo de software proporcionando al ingeniero de software un conjunto de formularios, guías y estándares que les ayudan a estimar y planificar su trabajo. Trabajando todo en conjunto y, de manera efectiva, es muy factible poder cumplir con todos los niveles de CMMI.

Es importante entender que PSP es una disciplina personal, sin embargo, en conjunto con TSP y CMMI se pueden lograr varios beneficios para la organización. A continuación, verás sus antecedentes y la manera en que se relaciona con las metodologías mencionadas.

1.1.1. Antecedentes de PSP

¿Cuáles son los pasos que sigues para realizar los programas que te dejan en tus clases? Contesta esta pregunta y ahora analiza si tienes una estructura de trabajo, ¿qué encontraste? Ahora responde a las siguientes:

- - ¿Qué pasa cuando algo te sale mal?
- - ¿Llevas un control de tiempo?
- - ¿Registras actividades?

Los programadores (o ingenieros de software) no siempre siguen una metodología de trabajo, y esto nos lleva a tener resultados diferentes, descontrol en la cantidad de errores, validaciones, duración del proyecto. Esta problemática no es reciente y aplica tanto para estudiantes, programadores junior (principiantes) o senior (expertos). Watts Humphrey, miembro del Instituto de Ingeniería de Software (SEI), de la Universidad de Carnegie Mellon, detectó esta problemática con los alumnos y se dio cuenta de que también en la industria sucedía lo mismo. Humphrey desarrollo una serie de disciplinas para los ingenieros de software, que posteriormente darían pie para estructurar el PSP.

PSP pretende mejorar las prácticas del desarrollo de software y la calidad del mismo. No hay mejor manera de hacer cosas con calidad que midiendo; es por eso que, en PSP se generan las métricas de líneas de código, líneas de código por hora, los tiempos de desarrollo, etc.
Es importante mencionar que PSP no tiene como objetivo demeritar el trabajo de una persona, ni hacerlo sentir menos que sus compañeros, sino simplemente mejorar, como su nombre lo indica: su **proceso personal de desarrollo**. Antes de estudiar en qué consiste este proceso, es importante conocer sus antecedentes:
Después de la Segunda Guerra Mundial, las organizaciones que se dedicaban a la industria del software, normalmente basaban su principal estándar de calidad en pruebas al software, generando departamentos dedicados en buscar y arreglar defectos después de la producción de sus productos. Entre los años 70's y 80's, la industria estadounidense se enfocó a conocer la manera en que los ingenieros de software realizaban sus trabajos y cómo era que desarrollaban sus procesos, lo cual permitió reconocer que el anterior modelo de "probar y corregir" era muy costoso en tiempo y recursos. En 1976, se comenzaron a incluir prácticas de inspecciones al software y posteriormente en 1987, Watts S. Humphrey, aplicó su Modelo de Capacidad de Madurez (CMM). En 1995, los primeros cursos de PSP fueron dados por su creador, en la universidad de Carnegie Mellon. En 1997, lanzó el libro *An introduction to the personal software process* con un enfoque a los ingenieros de software. (García, Y. 2010. Pág. 1).
Ya que hemos conocido algo de la breve historia de PSP es momento para comenzar a conocer sus características, así como, saber: sí este modelo se implementa con éxito en una organización, cuales ventajas se obtendrían, así como lo contrario, que pasaría si no se tiene éxito en su aplicación, cuáles desventajas tendría.

1.1.2. Características de PSP

Para continuar describiendo el modelo PSP es importante mencionar sus características, así como, distinguir sus ventajas y desventajas. Según Humphrey, W. (1995.).

Características:

- Es una metodología de la Ingeniería de Software con fundamentos de CMMI.
- Tiene un enfoque hacia la producción de software de calidad.
- Favorece los procesos de estimación, planeación y desarrollo de software.
- Como todo proceso de calidad, está orientada a mantener la mejora continua.
- Se puede establecer junto con los modelos de calidad TSP y CMMI.
- Es un proceso definido y ayuda a medir la mejora.
- Involucra actividades de revisión e inspección.
- Está diseñado para uso individual.
- Se combinan actividades de: administración de proyectos, Ingeniería de software y calidad.

Cuando se logra un proceso de PSP, estable y maduro tiene las siguientes ventajas:

- Estimación más precisa de tiempos, costos y recursos.
- Cumplir con los compromisos.
- Productividad en aumento.
- Localización de los defectos desde fases iniciales.
- Mejora los tiempos del ciclo de vida.
- Reduce costos.
- Facilita el seguimiento a procesos.
- Los desarrolladores comprenden su condición actual y obtienen un ambiente y disciplina que propicia la mejora de su capacidad.

Cuando la implementación del PSP no tiene una administración adecuada, ocurre n las siguientes desventajas:

- Puede considerarse como un proceso burocrático porque genera documentación.
- La metodología es muy precisa, puede propiciar la exageración en su aplicación.
- Implementar esta metodología puede consumir mucho tiempo extra.
- La renuencia por parte de los desarrolladores de aplicar la documentación y por sentirse expuestos al evidenciar sus tiempos de desarrollo y defectos.
- Si no se considera como una inversión, generará la idea de ser un proceso muy costoso a un corto y mediano plazo.

PSP es un modelo que puede implementarse de manera exclusiva, sin embargo, puede ser perfectamente complementado con TSP y CMMI. En nuestro siguiente subtema veremos cómo se da esta relación.

1.1.3. Relación de PSP con TSP (*Team Software Process*) y CMMI (*Capability maturity model integration*)

El establecimiento y seguimiento de los procesos son como los hábitos, al principio son difíciles de obtener, pero ya que se han logrado son más difíciles de romper. Esto también pasa en la industria del software, lo procesos pueden ser largos, complejos o difíciles de comprender y adoptarlos puede tornarse en una actividad complicada. Es por ello que se ha generado un marco de madurez en el proceso de software, el cual, muestra una manera ordenada para que la organización determine sus procesos y establezca sus mejoras.

El SEI trabaja liderando organizaciones de software en Estados Unidos y esto le ha llevado a definir el CMMI. Como puedes observar en la Tabla: Niveles del CMMI., tiene 5 niveles que son progresivos y demuestran cómo se deben lograr las áreas clave del

proceso (KPAs *Key process areas*), para adquirir progresivamente el nivel de madurez. CMMI es un modelo que contiene metas, métodos y prácticas necesarias para la práctica industrial de la ingeniería de software, por lo tanto, necesitarás comprender cómo aplicar estos principios. La pregunta es ¿cómo lograr esto? y una de las respuestas es: comenzando por tu proceso **personal** de desarrollo de software.

Nivel	Descripción
Nivel 5: Optimizado	Administración del proceso del cambio. Tecnología de gestión del cambio. Prevención del defecto.
Nivel 4: Cuantitativamente administrado	Administración de calidad. Gestión de procesos cuantitativos.
Nivel 3: Definido	Evaluación entre pares. Coordinación intergrupal. Ingeniería del producto de software. Administración de la integración del software. Programa de formación. Definición del proceso del software. Enfoque al proceso del software.
Nivel 2: Repetible	Administración de la configuración del software. Aseguramiento de la calidad del software. Administración de la subcontratación del software. Seguimiento y supervisión a proyectos de software. Planeación de proyectos de software. Administración de requerimientos.
Nivel 1: Inicial	

Tabla. Niveles del CMMI. (Humphrey, W.1995. p. 7)

En el entendido de que: todo esfuerzo hacia la calidad debe considerar el plano personal, debemos estar conscientes de que todos tenemos responsabilidades con otros y con nosotros mismos. Debemos comprender nuestras habilidades para desarrollar tareas, administrar las debilidades y desarrollar más fortalezas. Esto debemos convertirlo en parte de nuestra vida laboral y buscar la excelencia sin dejar de considerar el significado de ser humanos. Partiendo de esto, PSP tiene los siguientes enfoques:

- Identificar en los grandes sistemas de software, todos los métodos y prácticas que pueden ser usados por individuos.
- Definir los conjuntos de métodos y prácticas que pueden ser aplicados mientras se desarrollan programas pequeños.
- Estructurar los métodos y prácticas localizados e ir introduciéndolos gradualmente.

- Proveer en un entorno educativo, ejercicios apropiados para enseñar los métodos y prácticas estructurados.

PSP tiene un marco de trabajo muy parecido al de CMMI, ambos comparten las siguientes áreas de proceso:
Nivel 5 Optimizado
- Administración del proceso del cambio.
- Tecnología de gestión del cambio.
- Prevención del defecto.

Nivel 4 Cuantitativamente administrado
- Administración de calidad.
- Gestión de procesos cuantitativos.

Nivel 3 Definido
- Evaluación entre pares.
- Ingeniería del producto de software.
- Administración de la integración del software.
- Definición del proceso del software.
- Enfoque al proceso del software.

Nivel 2 Repetible
- Seguimiento y supervisión a proyectos de software.
- Planeación de proyectos de software.

Como has podido observar, PSP cubre 12 de las 18 áreas clave del proceso de CMMI. Las áreas que no se cubren son principalmente aquellas que tienen que ser desarrolladas en equipo, ya que de manera individual no podrían existir, por ejemplo: la subcontratación de desarrollos de software, la administración de requerimientos y de la configuración del software, el aseguramiento de calidad y programas de entrenamiento, etc.

TSP está construido sobre el fundamento de PSP y todos los miembros del equipo TSP deben ser entrenados en el uso de PSP. Los principios de PSP y lógica aplican de la misma manera para el TSP. Los principios básicos de PSP consisten en: (1) procesos definidos y estructurados para mejorar el proceso de trabajo, (2) estos procesos deben ajustarse a las características y preferencias de cada miembro del equipo, (3) involucramiento de los desarrolladores en el diseño de los procesos, (4) incluir las buenas prácticas de PSP en la configuración de los estándares, (5) proceso de mejora continua eficiente.

La principal razón del entrenamiento PSP es motivar en los desarrolladores la convicción de trabajar el proceso de TSP. Tu trabajo como líder es convencerlos de que esos mismos beneficios son importantes para el proyecto y cada miembro del equipo debe continuar utilizando esas prácticas.

La lógica para el TSP se construye sobre la lógica de PSP, la cual consiste en que:

- Los equipos trabajan mejor cuando los miembros cooperan y se toleran unos a otros.
- Los miembros del equipo pueden cooperar y tolerarse mejor cuando todos ellos utilizan un adecuado y bien definido proceso.
- Los equipos comprenden y siguen conscientemente un proceso solamente cuando ellos han participado en la definición de ese proceso.
- Los equipos producen productos de calidad para predecir costos y calendarios solamente si ellos utilizan procesos definidos consistentemente.
- Los equipos definen procesos solamente si los miembros saben cómo hacer trabajo con calidad.
- Los equipos siguen procesos solamente si ellos son motivados para hacer esto por sus pares, líderes y gerentes.

Uno de los principales retos del líder es ayudar a su equipo a producir los tipos de proyectos que necesitan para lograr sus objetivos y motivarlos a seguir el proceso. Esos objetivos consisten en hacer trabajos con calidad y cubrir sus calendarios. (Humphrey, W 2006. pp. 116-118).

1.2. Generalidades de PSP

Hasta este punto ya conoces las características de PSP, sus ventajas, desventajas, antecedentes y cómo se relaciona con TSP y CMMI. En el presente tema conocerás como se conforma el proceso de una línea base, la cual consiste en un marco de trabajo necesario para realizar el primer proceso de PSP. Conocerás los formatos, los elementos, el proceso y mediciones que conforman el PSP 0.

1.2.1. El proceso de línea base

Primeramente, comenzaremos con la siguiente pregunta: ¿Qué es un proceso?
Un **proceso** es una serie de pasos para realizar un trabajo. Cuando tenemos los procesos debidamente definidos, estos son una guía para hacer el trabajo. El proceso de software establece la técnica y marco de trabajo para aplicar métodos, herramientas, tareas específicas, mediciones y proveer el éxito y criterios de entrada para mayores logros. Los mejores métodos pueden ser descritos como procesos para ayudar a los integrantes del equipo a aprenderlos. En resumen, un proceso definido identifica los principales pasos de un trabajo.

- Ayuda a separar la rutina de las actividades complejas.
- Se establece el criterio para comenzar y terminar cada paso del proceso.
- Mejora la comprensión del proceso y da una base sólida para la automatización del proceso.

Los procesos definidos también nos ayudan a generar mediciones. Las **mediciones** son útiles para comprender el desempeño, para administrar el trabajo del equipo, para planear y administrar la calidad de los productos que producen.
Cuando trabajas en un equipo TSP tú debes definir tu propio proceso. Al principio todas las tareas te parecerán abstractas y a medida que vas desarrollándolas más y las defines de una mejor manera, con suficiente detalle estarás favoreciendo la mejora.

El proceso de **línea base** es el principal objetivo del PSP0 y consiste en un marco de trabajo para escribir tu primer programa PSP y el acopio de datos de tu trabajo. Estos datos proveen una línea base comparativa para determinar el impacto del método PSP en tu trabajo. Este proceso provee los siguientes beneficios:

- Una estructura conveniente para desarrollar tareas de pequeña escala.
- Un marco de trabajo para medir esas tareas.
- Fundación para la mejora del proceso.

PSP0 es un proceso simple, definido y personal. Consiste en hacer un plan, utilizar tu diseño actual y métodos de desarrollo que utilizas para construir un programa pequeño. Después lleva el registro de los tiempos y de los defectos que generaste. Por último prepara un informe resumido. Observa la Figura: Evolución del proceso PSP0.

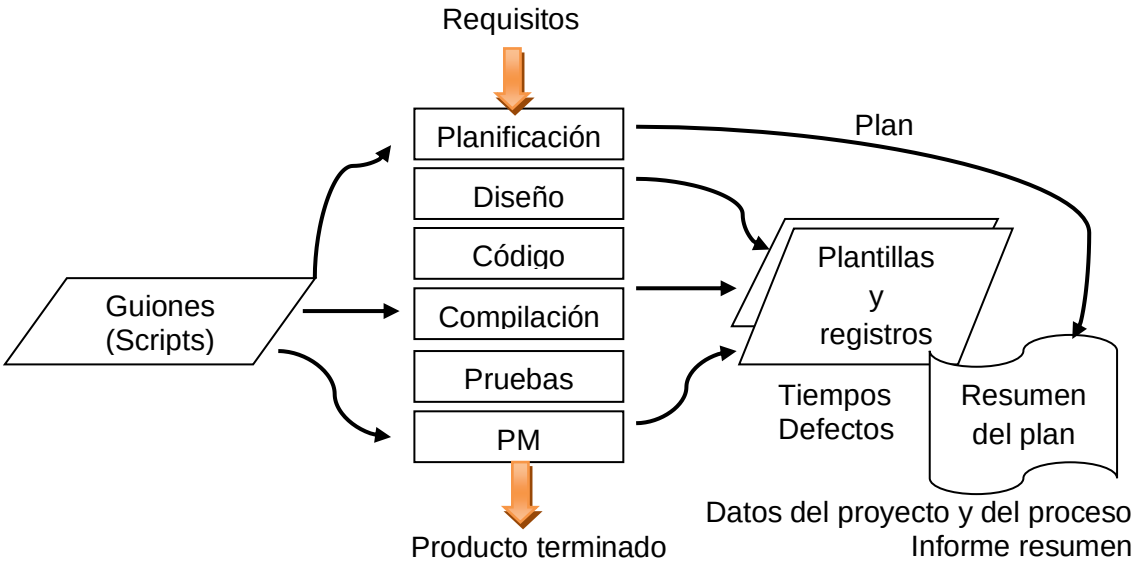


Figura. Evolución del proceso PSP0.

Como puedes observar en la figura, el proceso de PSP0 tiene las siguientes fases:

Planificación: debes producir un plan para desarrollar el programa definido en los requisitos.

Diseño: produce una especificación de diseño para el programa definido en los requisitos.
Código: convierte las especificaciones de diseño en instrucciones del lenguaje de programación.
Compilación: se traducen las instrucciones del lenguaje de programación en código ejecutable.
Pruebas: verificar que el código ejecutable cumple con los requisitos correcta y completamente.
PM: Postmortem, resume y analiza los datos del proyecto.

Deberás tomar en cuenta los siguientes factores para saber cómo aplicar estas etapas:

1. Cuando los programas son pequeños o bien comprendidos, puede ejecutar las fases en orden.
2. Cuando los programas son grandes o que no se comprenden bien puede requerir un enfoque iterativo. Por ejemplo, en un proyecto el diseño se puede realizar en un sólo paso y descubrir que hay dos módulos que ejecutarán separadamente las etapas de código, compilación y pruebas. Y por último habrá una sola fase de PM que integrará datos de ambos módulos.
3. Puede haber más de dos ciclos, cada módulo podrá incluir su propia fase de diseño. Cada uno enfocado a producir parte del programa. El tamaño de cada módulo determinará cada ciclo. Este tamaño podrá medirse con la cantidad de clases, métodos o funciones. Esto quedará determinado por ti.

Como has podido observar, el proceso involucra el uso de formas y otros elementos, mismos que a continuación se te presentan explicando la manera en que deberás utilizarlos.

1.2.2. Las formas y elementos de PSP

Las formas de PSP

En PSP se utilizan muchas formas. Aunque en un principio se podría ver esto como una desventaja, posteriormente descubrirás que las formas te serán de gran ayuda.
Generalmente los proyectos desarrollan los siguientes pasos:

1. Determinar que lo que se va a hacer.
2. Decidir cómo hacerlo.
3. Hacerlo.
4. Revisar y asegurarte que es correcto.
5. Corregir problemas.
6. Entregar el resultado final.

Inicialmente todo proceso lleva un aprendizaje y una inversión de tiempo considerable. Aparte del proceso el registro de formas, es una actividad pesada y deberá ser revisada periódicamente para asegurar que ella aún presenta tus necesidades, por lo que es necesario asegurarse que todas las formas del proceso están diseñadas en su conjunto de manera coherente. Cuando no son cuidadosamente diseñadas, se podría duplicar información, la terminología puede ser inconsistente y, en general, pueden presentar varias confusiones.

Elementos de PSP

Como viste en la Figura: Evolución del proceso PSP0, el **proceso de PSP0** consta de diferentes pasos: en el paso de planeación, generas el plan para hacer el trabajo; los siguientes 4 pasos abarcan: diseño, código, compilación y pruebas. Al final en el paso de postmortem, compararás tu actual desempeño con el plan y producirás un reporte de resumen. Aunque los pasos de planeación y postmortem parecen no ser necesarios cuando escribes programas pequeños, sí lo son, para reunir los datos para administrar y mejorar tu proceso personal. Si el programa es tan simple que pareciera que no tiene sentido, entonces hacer el plan debiera ser una tarea trivial. Sin embargo, es programación “trivial” pudiera albergar sorpresas que en el plan deberían anticiparse. En tanto vayas ganando experiencia te darás cuenta que el postmortem es un tiempo ideal para pensar sobre tus datos y ver la manera de mejorar.

Un proceso puede ser simple o complejo. La definición de un proceso largo tendrá muchos elementos o fases. Cada una de ellas puede ser vista como un proceso definido. En un bajo nivel las fases están compuestas por pasos que no tienen una estructura. En procesos largos cada paso puede ser una fase, en el siguiente nivel cada fase puede tener una definición de proceso con los pasos que comienzan con tareas de bajo nivel. Cuando un elemento del proceso tiene una definición y una estructura puede ser llamada fase. Si no cuenta con una estructura definida, será llamada paso o tarea.

Como todo proceso de calidad, deberá ser medido para asegurarnos qué funciona y qué puede ser mejorado, es por ello que PSP incorpora elementos y puntos de medición para comparar cómo es el desempeño de cada ingeniero de software y cómo podrá ser mejorado. A continuación, sabremos cómo funciona este proceso.

1.2.3. Proceso y mediciones del PSP 0

Los scripts de PSP te guiarán a través de los pasos del proceso. Los principales elementos del guion del proceso son su propósito, los criterios de entrada, las fases o pasos para ser desarrollado. El guion del proceso PSP0 es una serie de directrices de cómo usar el proceso. Se describe el propósito, los criterios de ingreso, aspectos generales, pasos y criterios de salida.

- La **planificación**. Deberás contar con los requisitos del proyecto y completar las partes no sombreadas del plan de la Tabla: *Resumen del plan del proyecto*. Por último, escribe el tiempo dedicado a hacer el plan en el cuaderno de *Registro de tiempos*.
- El **diseño.** Se diseña la solución del programa antes de desarrollarlo. Se puede utilizar cualquier técnica de diseño o modelado, como: diagramas de flujo, seudocódigo, diagramas UML, etc. Cuando se termine el diseño se deberá registrar el tiempo que se tardó en hacerlo.
- **Codificación:** Se implementa el diseño, codificándolo en un lenguaje de programación. Deberá aplicar los estándares de codificación. Al término deberá registrar sus tiempos en el cuaderno de registro de tiempos.
- **Compilación:** Compilar el programa y corregir todos los defectos que se localicen. Continuar compilando y corrigiendo hasta depurar el código. Aunque corrijas el código todo este tiempo se registra en el cuaderno de registro de tiempos.
- **Pruebas:** Realiza las pruebas necesarias para asegurar que los programas están cumpliendo con el requisito y no presentan errores. Todo el tiempo que se dedique en esta fase se contabiliza aún que hayas tenido que codificar para corregir errores. El tiempo se registra en el cuaderno de registro de tiempos.
- **Postmortem:** Completa los datos de la tabla resumen del plan del proyecto, dedica tiempo para hacer cálculos finales y escríbelos en el tiempo postmortem estimado. Utiliza el tiempo estimado para calcular el tiempo de desarrollo total y el resto de los cálculos.

Observa las siguientes tablas, éstas te indicarán cómo se debe ejecutar el proceso PSP0 y en ellas puedes revisar: Muestra el guion completo; Muestra la guía para el plan; Muestra la guía del desarrollo y Muestra la guía postmortem.

Guion del proceso PSP0	
Propósito	Guía para el desarrollo de programas pequeños.
Criterios de entrada	La descripción del problema. Tabla Resumen del plan del proyecto PSP. Datos de tamaños y tiempos reales de programas anteriores. Cuaderno de Registro de Tiempos.

Paso	**Actividades**	**Descripción**
1	Planificación	Obtén una descripción de las funciones del programa. Escribe los datos del plan en la tabla Resumen del Plan del proyecto. Anotar el tiempo de planificación en el Cuaderno de registro de tiempos.
2	Diseño	Diseñar el programa.

		Anotar el diseño en el formato indicado. Anotar el tiempo del diseño en el Cuaderno de registro de tiempos.
3	Codificación	Implementa el diseño. Utiliza un formato estándar para introducir el código. Anota el tiempo de codificación para el Cuaderno de registro de tiempos.
4	Compilación	Compilar el programa. Corregir defectos encontrados. Anotar el tiempo de compilación en el Cuaderno de registro de tiempos.
5	Pruebas	Probar el programa. Corregir los defectos encontrados. Recuerda que cada error encontrado debes registrarlo en la bitácora de registro de defectos. Anotar el tiempo de pruebas en el Cuaderno de registro de tiempos.
6	Postmortem	Completa la tabla de Resumen del plan del proyecto con los datos de tiempo y tamaño reales. Anotar el tiempo postmortem en el Cuaderno de registro de tiempos.
Criterios de salida		Programa probado completamente. Diseño documentado. Programa completo. Resumen del plan del proyecto terminado. Cuaderno de registro de tiempos terminado.

Guion de la planeación PSP0

Propósito	Guía para el proceso de planeación de PSP0
Criterios de entrada	La descripción del problema. Tabla Resumen del plan del proyecto PSP. Cuaderno de Registro de Tiempos.

Paso	**Actividades**	**Descripción**
1	Requisitos del programa	Producir u obtener los requisitos del programa. Asegurarte que esos requisitos son claros y sin ambigüedades. Resolver preguntas.
2	Estimación de recursos	Realizar tu mejor estimación del tiempo requerido para desarrollar el programa.
Criterios de salida		Requisitos documentados. Resumen del plan del proyecto terminado con tiempos estimados de desarrollo.

	Cuaderno de registro de tiempos terminado.

Guion del desarrollo PSP0	
Propósito	Guía para el desarrollo de programas pequeños.
Criterios de entrada	Declaración de requisitos. Tabla Resumen del plan del proyecto con la estimación del tiempo de desarrollo del programa. Cuaderno de Registro de Tiempos. Estándar del tipo de defectos.

Paso	**Actividades**	**Descripción**
1	Diseño	Revisar los requisitos y diseñar el programa. Registrar el tiempo del diseño y defectos encontrados en requerimientos. Registrar tiempos en el Cuaderno de registro de tiempos.
2	Codificación	Implementa el diseño. Utiliza un formato estándar para introducir el código. Anota el tiempo de codificación para el Cuaderno de registro de tiempos.
3	Compilación	Compilar el programa. Corregir los errores encontrados y registrarlos. Anotar el tiempo de compilación en el Cuaderno de registro de tiempos.
4	Pruebas	Probar el programa. Corregir los errores encontrados y registrarlos. Anotar el tiempo de pruebas en el Cuaderno de registro de tiempos.
Criterios de salida		Programa probado completamente. Diseño documentado. Programa completo. Resumen del plan del proyecto terminado. Cuaderno de registro de tiempos terminado.

Guion del postmortem PSP0	
Propósito	Guía para el proceso postmortem.
Criterios de entrada	Descripción del problema y declaración de requisitos. Tabla Resumen del plan del proyecto con tiempos de desarrollo. Cuaderno de Registro de tiempos y defectos. Una prueba y ejecución del programa.

Paso	Actividades	Descripción
1	Registro de defectos	Revisar los requisitos y diseñar el programa. Registrar el tiempo del diseño y defectos encontrados en requerimientos. Registrar tiempos en el cuaderno de registro de tiempos.
2	Consistencia en la definición del defecto	Revisar que los datos en todos los defectos en el registro de defectos sean exactos y completos. Verificar que el número de defectos inyectados y removidos por fases son razonables y correctos. Utilizando su mejor recuerdo, corregir cualquier defecto de datos faltantes o incorrectos
3	Time	Revisar el tiempo de registro terminado de errores u omisiones. Revisar el registro de algún dato faltante o incompleto.
Criterios de salida		Programa probado completamente. Resumen del plan del proyecto terminado. Cuaderno de registro de tiempos terminado.

Mediciones PSP0

PSP0 tiene dos mediciones:

- El tiempo gastado por fase.
- Los defectos encontrados por fase.

PSP0 tiene cuatro formas:

> **Resumen del plan del proyecto:** resumen planificado, tiempo actual y defectos por fase.
> **Registro de tiempos:** se utiliza para registrar el tiempo.
> **Registro de defectos:** se utiliza para registrar defectos.
> **Estándar de tipo de defecto:** se utiliza para definir el estándar de los tipos de defectos.

El tiempo por fase es un simple registro del tiempo gastado en cada parte del proceso PSP. Registrar tiempos y defectos puede tomar algo de tiempo, nos servirán para aplicar muchas herramientas del PSP. Deberás contabilizar el tiempo que tardas en corregir todos los defectos durante la fase de compilación y pruebas. El defecto puede ser muy simple o abarcar varios segmentos.

La razón por la que se registran tiempos y defectos es para ayudarte a planear y administrar proyectos. Estos datos te servirán para identificar dónde gastas tiempo, dónde inyectas y corriges más defectos. Además, te serán de utilidad, para realizar cambios en tu desempeño actual y buscar la calidad de tus productos de trabajo.

1.2.4. Registro de tiempos y defectos

Registro de tiempos

Para comenzar con todo este proceso, deberás identificar la mejor manera de contabilizar tus tiempos para cada paso que realices de los procesos de PSP. Al realizar esto, te ayudará a conocer las tareas que realizas y cuánto tiempo demoras en cada una. Por lo tanto, tendrás que registrar tus tiempos y para ello podrás utilizar una plantilla destinada al registro. (Ver Tabla: Registro de tiempos PSP). Esta tabla o cualquier otro formato que utilices deberán tener los siguientes datos:

- Tu nombre, el proyecto, lenguaje, fecha, y otros datos en el encabezado.
- Comienzo del proyecto o programa.
- La fase del proceso para la tarea.
- La fecha y hora en que la que comenzaste y terminaste trabajando en la tarea.
- Tiempo de alguna interrupción.
- Tiempo delta trabajando en la tarea.
- Comentarios.

Además, la herramienta que utilices deberá ayudarte a capturar el tiempo de inicio de la tarea, el tiempo de término y el tiempo de interrupción y de manera automática debiera calcular el tiempo delta, el cual es el tiempo real dedicado a la tarea. Cuando las actividades duran varias horas, seguramente tendrás interrupciones, como llamadas por teléfono, conversaciones, comidas, etc. Si no estás consciente del tiempo que interrumpes tu trabajo, no sabrás exactamente cuál es el tiempo que tardas en una actividad. Tal vez pienses que los tiempos de interrupción son realmente pequeños que puedes ignorarlos. Podrías colocar en la columna comentarios cuando comienzas el tiempo de interrupción, para así calcular el tiempo perdido cuando hayas terminado la interrupción, de tal manera que si este cálculo te da, por ejemplo, 18 minutos, deberás registrarlos en la columna de Tiempo Int. (Tiempo de interrupción).

Estudiante				Fecha		
Programa				# Programa		
Instructor				Lenguaje		
Proyecto	Fase	Fecha y hora de inicio	Tiempo Int.	Fecha y hora de término	Tiempo Delta	Comentarios

Tabla. Registro de tiempos PSP
(Echeverría, C.M., Echeverría, C.D. Asencio, J.L. 2006. Pág. 23)

Propósito	Utilizar la plantilla de registro de tiempos para anotar el tiempo gastado en cada actividad del proyecto. En PSP normalmente cada fase tiene una actividad, pero los proyectos grandes tienen múltiples actividades en una simple fase.
General	Registra siempre el tiempo que demoras haciendo actividades. Registra el tiempo en minutos. Se lo más exacto posible. Si olvidaste registrar el comienzo, el término o interrupción para alguna actividad, inmediatamente registra tu mejor estimación.
Encabezado	Introducir tú nombre, la fecha, nombre y número del programa, nombre del instructor y lenguaje de programación.
Proyecto	Introduce el nombre, número o clave del proyecto.
Fase	Introduce el nombre de la fase en la que estás trabajando, por ejemplo planeación, diseño, pruebas, etc.
Fecha y hora de inicio	Introduce la fecha y hora de inicio de alguna actividad del proceso.
Tiempo Int.	Registra el tiempo de interrupción, en el que no estuviste trabajando en alguna actividad del proceso. Si tuviste muchas interrupciones, registra el tiempo total, también puedes escribir la razón por la cual interrumpiste la actividad en la columna de comentarios.
Fecha y hora de término	Registra la fecha y hora en la que terminaste la actividad.
Tiempo delta	Registra el cálculo del tiempo que realmente trabajaste en la actividad menos el tiempo de interrupción.
Comentarios	Registra algún comentario que debas recordar posteriormente.

Tabla. PSP Instrucciones del registro de tiempos.

Registro de defectos

Posiblemente te harás la pregunta ¿qué es un defecto en PSP?, una respuesta es la que sigue, un defecto es algo que no es correcto o error en un programa, desde un simple carácter en el lugar que no le corresponde (error de sintaxis), hasta un defecto de funcionalidad donde haya errores en la lógica del código.

Cuando encuentras un defecto y decides arreglarlo, cuenta el tiempo de corrección, cuando terminaste de corregirlo, introduce todos los datos del defecto: tipo de defecto, la fase en la que se detectó, tiempo de corrección y una breve descripción del defecto. Si olvidaste registrar alguno de estos datos, realiza tu mejor estimación. Normalmente en la fase de compilación y pruebas encontrarás la mayoría de los defectos, sin embargo, los puedes encontrar en cualquier fase y corregirlos. Si este es el caso puedes registrar la fase en la que los encontraste y corregiste.

Si estos errores o defectos no son detectados por los ingenieros de software, muy probablemente quién los encontrará será el cliente, es por ello que es una buena práctica que de manera personal cada ingeniero de software busque y detecte la mayor parte de sus defectos.

Tipos de Defectos

10 Documentación	60 Chequeo
20 Sintáxis	70 Datos
30 Construcción, Empacar	80 Función
40 Assignación	90 Sistema
50 Interfaz	100 Ambiente

Formato del Registro de Defectos

Estudiante ______________ Fecha ______________
Instructor ______________ Programa # ______________

Fecha	Número	Tipo	Encontrado	Removido	Tiempo de compostura	Defecto arreglado

Descripción:

Fecha	Número	Tipo	Encontrado	Removido	Tiempo de compostura	Defecto arreglado

Descripción:

Fecha	Número	Tipo	Encontrado	Removido	Tiempo de compostura	Defecto arreglado

Descripción:

Fecha	Número	Tipo	Encontrado	Removido	Tiempo de compostura	Defecto arreglado

Descripción:

Tabla. Registro de defectos.
(Echeverría, C.M., Echeverría, C.D. Asencio, J.L. 2006. Pág. 25)

El tiempo de corrección es a veces mal comprendido. Este es el tiempo para encontrar y corregir el defecto. Por ejemplo:

> 8:00 ejecutas el compilador y localizas error de tipo en "*line 23 – type mismatch*".
> 8:01 ejecutas el editor.
> 8:10 cambias la declaración de la línea 5, tecleas el tipo *"real"* en lugar del *"integer".*
> 8:12 ejecutas el compilador y te indica que no hay errores "no errors".

¿Cuánto tiempo tiene este defecto?

> 12 minutos.

A continuación, revisa las instrucciones para el llenado de la tabla de registro de defectos.

Propósito	Se utiliza esta plantilla para registrar los defectos detectados y corregidos.
General	Registrar cada defecto separadamente y completamente.
Encabezado	Introduce tu nombre, fecha, nombre y número de programa, nombre del instructor y lenguaje de programación que estás utilizando.
Proyecto	Dar a cada programa un nombre o número. Por ejemplo, registro de prueba de programa.
Fecha	Registra la fecha en la que localizaste los defectos.
Número	Introduce el número del defecto. Utiliza un número secuencial para cada defecto por programa o módulo. Comienza desde el 1 (1...100... etc.)
Tipo	Registra el tipo de defecto utilizando la lista de tipos de defectos seleccionando el que consideres que se ajusta al tipo de defecto encontrado.
Inyección	Registra la fase en la que se inyectó el defecto.
Removido	Introduce la fase en la cual corregiste el defecto.
Tiempo de corrección	Introduce el tiempo necesario para encontrar y corregir el defecto.
Referencia de corrección	Si al corregir estos defectos tú o alguien más ha introducido nuevos defectos registra el número del defecto origen, si no lo tienes ubicado, coloca una X.
Descripción	Escribe una breve descripción del defecto, el error que lo causó y por qué sucedió esto.

1.2.5. Resumen del plan del proyecto

La plantilla del resumen del plan del proyecto nos servirá para definir qué trabajo se hará y el tiempo que necesitará. Hay que definir la tarea principal, estimar el tiempo y recursos y la forma de trabajo para la revisión y seguimiento. El plan del proyecto es una parte importante de éste. Para proyectos grandes, es muy importante contar con un plan. Poco a poco irás adquiriendo mayor habilidad para elaborar planes, al principio comenzarás

estimando el tamaño del producto y el tiempo que te llevará realizarlo. Este debe realizarse antes de comenzar el proceso, cuando hayas terminado deberás registrar los tiempos reales del tamaño y tiempo. Así mismo lo harás con la fecha estimada de culminación y la fecha real. Esta práctica te servirá para realizar mejores estimaciones cada vez que realices un programa, ya que podrás ir ajustándolos en base a la experiencia.

A continuación, se te presenta la tabla del resumen del plan del proyecto y posteriormente las instrucciones del llenado del resumen del plan del proyecto.

Proyecto
1

PSP0 Resumen del plan del proyecto - Programa 1A

Estudiante			Fecha	?????
Programa			# programa	
Instructor			Lenguaje	?????

	Plan	Real	Hasta la fecha	% Hasta la fecha
Tiempo en la fase (min.)				
Planificación				
Diseño				
Codificación				
Compilación				
Pruebas				
Postmortem				
Total				
Defectos inyectados				
Planificación				
Diseño				
Codificación				
Compilación				
Pruebas				
Total en desarrollo				
Defectos eliminados				
Planificación				

Diseño			
Codificación			
Compilación			
Pruebas			
Total en desarrollo			
Después del desarrollo			

Tabla. Resumen del plan del proyecto.
(Zapata, J., García, J., Cerrada, J. 2001. Pág. 142)

Revisa las siguientes instrucciones para el llenado de la forma de defectos:

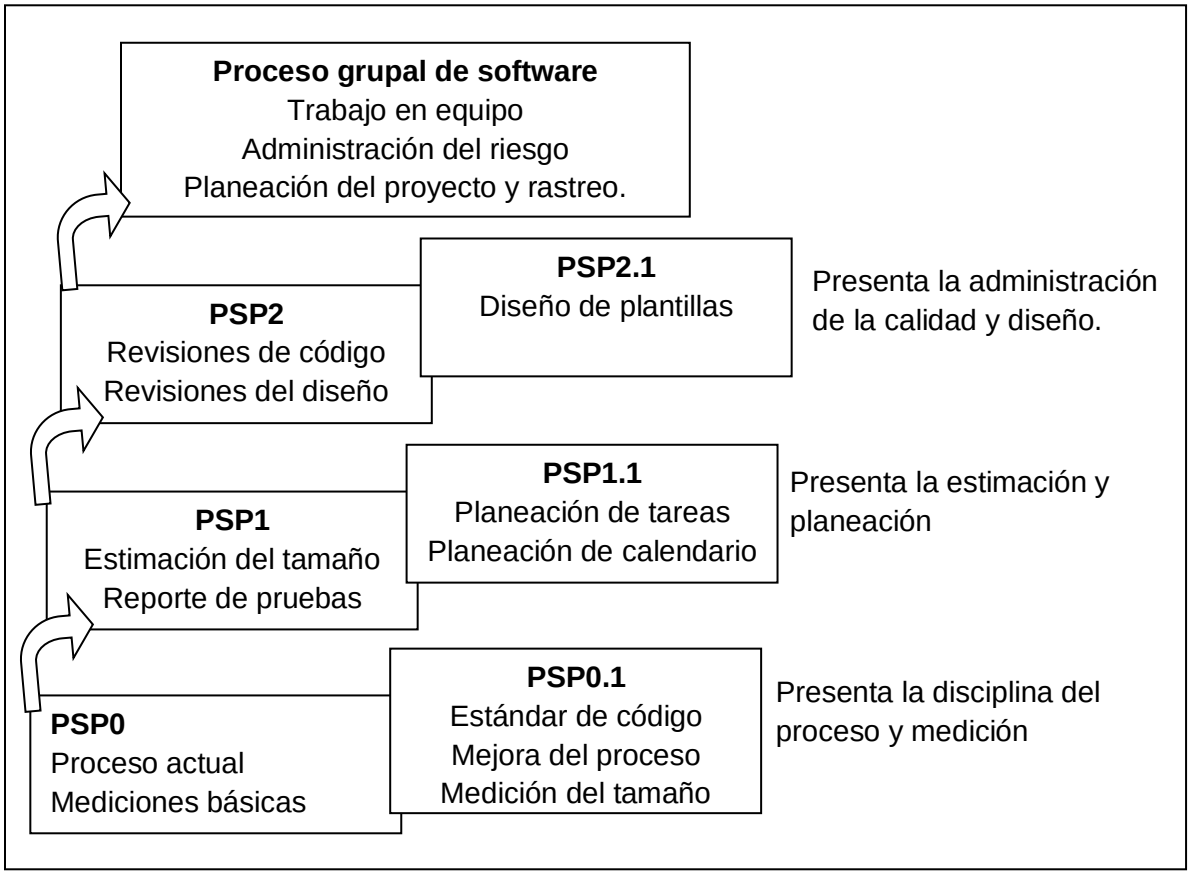
Propósito	Para contener los datos reales del plan de los programas.
General	"Hasta la fecha" significa el total actual de los valores de todos los productos desarrollados.
Encabezado	Introducir tú nombre, la fecha, nombre y número del programa, nombre del instructor y lenguaje de programación.
Tiempo en la fase	Introduce el tiempo total estimado. Introduce el tiempo actual por fase y el tiempo total. Hasta la fecha: introduce la suma de los tiempos actuales para este programa más los tiempos a la fecha del programa más recientemente desarrollado. % hasta la fecha: Introduce el porcentaje del tiempo hasta la fecha en cada fase.
Defectos inyectados	Introduce los defectos actuales por fase y el total actual de defectos. Hasta la fecha: Introduce la suma de los defectos actuales inyectados por fase y los valores hasta la fecha del programa más recientemente desarrollado.
Defectos removidos	Hasta la fecha: Introduce los defectos actuales removidos por fase más el total de defectos del último programa desarrollado. % hasta la fecha: Introduce el porcentaje de defectos removidos hasta la fecha por fase. Después del desarrollo, registrar todos los defectos encontrados durante las pruebas uso, re uso o modificación del programa.

Algunos desarrolladores no consideran importante registrar los defectos, sin embargo todos los programadores que realizan esta actividad logran elevar la calidad de sus productos de trabajo y aumentar su productividad.

1.2.6. Personalización del proceso inicial

En esta unidad didáctica analizarás y realizarás varios programas utilizando la evolución del proceso que se muestra en la Figura: Evolución del PSP. El primer proceso es un proceso genérico simple, que comienza con la escritura de tus programas – como normalmente lo haces – lo que agregarás es el conteo del tiempo y medición de defectos. En el PSP0 aplicarás esto en un programa, posteriormente la medición del tamaño la verás en el programa PSP0.1. La medición del tamaño es una actividad importante ya que te servirá para comprender cómo trabajas y que con qué velocidad vas mejorando tus tiempos.

En el resto de esta unidad didáctica verás cómo se utilizan la estimación de datos históricos del tamaño y tiempo del programa. Hasta que logres la estimación y planeación de tareas y calendarios. El contenido de un curso PSP es llevar a cabo todo el proceso en la organización y demostrar con proyectos reales como cada desarrollador implementa su proceso personal de software, hasta complementar objetivos del Proceso grupal de software, como se demuestra en la Figura: Evolución del PSP.



Cierre de la unidad

Fue una unidad llena de nuevos conocimientos, para que adquieras habilidades que te ayudarán a tener una mejor formación como ingeniero de software. Revisaste desde los antecedentes del PSP y metodologías con las que se relaciona como TSP y CMMI, hasta el primer proceso PSP0 y todos los formatos en los que se apoya. Hemos visto que datos debemos registrar y cómo hacerlo. También revisamos cómo se debe evolucionar en el conocimiento de los procesos PSP.

En las siguientes unidades podrás poner en práctica los conocimientos de esta unidad más los nuevos que tienen que ver con los siguientes procesos del PSP.

Para saber más…

Si deseas saber acerca de PSP, TSP o CMMI puedes consultar la siguiente dirección electrónica:

- http://www.sei.cmu.edu/

Es el sitio oficial de la empresa que comercializa estos modelos. Encontrarás: novedades, productos y servicios, libros, videos, artículos, etc.

Fuentes de consulta

Blokdyk, G. (2020). *Personal software process: A complete guide*. 5STARCooks.

Blokdyk, G. (2017). *Personal software process: Best practices guide*. CreateSpace Independent Publishing Platform.

Hayes, W., & Over, J. W. (1997). *The Personal Software ProcessSM (PSPSM): An empirical study of the impact of PSP on individual engineers*. Carnegie Mellon University.

Humphrey, W. (1995). *A discipline for software engineering* (The complete PSP book). Addison-Wesley.

Humphrey, W. (2005). *PSP: A self-improvement process for software engineers*. Addison-Wesley.

Humphrey, W. (2006). *TSP(SM): Leading a development team*. Addison-Wesley.

Flores Figueroa, J. (2016). *Calidad en el software: PSP & TSP*. Académica Española.

Pomeroy-Huff, M., Cannon, R., Chick, T. A., Mullaney, J., & Nichols, W. (2014). *The Personal Software Process (PSP) body of knowledge, version 2.0*.

Zapata, J., García, J., & Cerrada, J. (2001). *Introducción al proceso software personalSM*. Addison-Wesley.

Bibliografía complementaria

Alvarado, A. (2008). *Desarrollo de sistemas con PSP y TSP*. México, D.F. https://docplayer.es/25202328-Desarrollo-de-sistemas-con-psp-y-tsp.html

Barbieri, S., Bianchi, A., & Rossi, G. (2007). *Framework de mejora de procesos de desarrollo de software*. La Plata, Argentina. http://sedici.unlp.edu.ar/bitstream/handle/10915/4075/Documento_completo.pdf?sequence=1

Chrissis, M. B., Konrad, M., & Shrum, S. (2009). *CMMI: Guía para la integración de procesos y la mejora de productos* (2ª ed.). Madrid, España. http://www.sei.cmu.edu/library/assets/cmmi-dev-v12-spanish.pdf

Echeverría, C. M., Echeverría, C. D., & Asencio, J. L. (2006). *Implementación de un sistema integrado de control de costos de producción, órdenes de trabajo, presupuesto de obras, bodega y control de inventario utilizando PSP (Personal Software Process) y TSP (Team Software Process)*. Guayaquil, Ecuador. http://www.dspace.espol.edu.ec/handle/123456789/5006

García, Y. (2010). *Proceso de software personal*. EcuRed (Enciclopedia colaborativa en la red cubana). Cuba. http://www.ecured.cu/index.php/Proceso_de_Software_Personal