Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

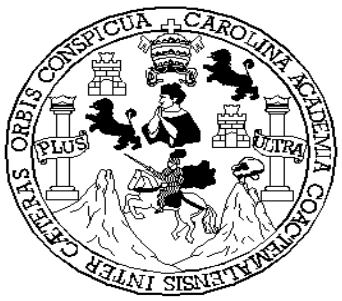
UN ANÁLISIS DEL PROCESO DE SOFTWARE PERSONAL EN LAS EMPRESAS GUATEMALTECAS

MARCO VINICIO IBARRA GUZMAN

Asesorado por Ing. Jorge Gómez Méndez

Guatemala, octubre de 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

UN ANÁLISIS DEL PROCESO DE SOFTWARE PERSONAL EN LAS EMPRESAS GUATEMALTECAS

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA

POR

MARCO VINICIO IBARRA GUZMAN

ASESORADO POR EL ING. JORGE GÓMEZ MÉNDEZ

AL CONFERÍRSE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	
VOCAL II	Lic. Amahán Sánchez Álvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Bra. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Veliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Inga. Virginia Victoria Tala Ayerdi de Alemana
EXAMINADOR	Ing.. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Luís Alberto Vettorazzi España
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

UN ANÁLISIS DEL PROCESO DE SOFTWARE PERSONAL EN LAS EMPRESAS GUATEMALTECAS,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha enero de 2004.

Marco Vinicio Ibarra Guzman

AGRADECIMIENTO

A:

DIOS:	por darme una vida llena de bendiciones y de oportunidades de crecer junto a mi familia tan especial.
MIS PADRES Y ABUELITA:	por su amor, cariño, confianza, esfuerzo, apoyo y comprensión.
MIS TÍOS Y PRIMOS:	por su apoyo incondicional.
MIS COMPAÑEROS Y AMIGOS:	por su ayuda, sincera amistad, apoyo, motivación y enseñanza.
UNIVERSIDAD DE SAN CARLOS:	por la enseñanza recibida.

DEDICATORIA

A mi abuelita y familia, ejemplo de esfuerzo, trabajo, comprensión, quienes con mucho amor mé han guiado hasta obtener este triunfo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

Figuras

Tablas

GLOSARIO

Administración de la calidad total:	visión integral para que una empresa mejore todos los aspectos de calidad y satisfacción a clientes, incluyendo velocidad de respuesta y servicios.
Administración de información del producto:	programa basado en sistemas que vincula, administra y organiza la información relativa a producción de varias fuentes internas y externas de proveedores a lo largo de una plataforma computacional en diferentes áreas funcionales y localizaciones físicas.
Bug:	puede ser una equivocación en la interpretación de un requerimiento, un error de sintaxis en una pieza de código o la causa (hasta ahora desconocida) del colapso de un sistema.
Debugging:	depuración, proceso de eliminar errores de un programa.
Defecto:	es un error desde una vista interna del sistema.
Diccionario de Datos:	tabla con las propiedades y estructura de los ficheros que forman una base de datos.

Falla: es un desvío respecto del comportamiento requerido del sistema.

Ingeniería de *Software:* nueva parte de la informática dedicada a crear entornos de desarrollo de *software* que adecuen la complejidad creciente de los programas a presupuestos aceptables. Se ocupa de todas las fases desde el diseño hasta el mantenimiento y usa nuevas técnicas y herramientas más acordes con las nuevas generaciones de *hardware* y *software*.

Precursor: es un evento o un conjunto de eventos que deben ocurrir antes de que la actividad pueda comenzar.

Proceso: es un conjunto de procedimientos organizado de tal modo, que los productos se construyen para satisfacer un conjunto de metas y estándares.

Riesgo: es un evento no deseado que tiene consecuencias negativas.

RESUMEN

El PSP es un conjunto ordenado de procesos definidos que orientan a los ingenieros de *software* a medir, evaluar y monitorear las tareas. La disciplina del PSP provee un marco estructurado para desarrollar habilidades personales y métodos que se necesitarán más adelante para ir forjando al ingeniero de *software*. Es importante que la calidad del *software* desarrollado abarque hasta el más mínimo detalle, por muy pequeño que éste sea, ya que si no se hace así, puede dañar el sistema entero. El uso de PSP depende del ámbito en el que se quiera utilizar y del propósito que le dé el programador.

Estos propósitos representan la marca personal de cada programador, ya sea que utilice PSP al máximo o que se limite solamente a algunos campos. He aquí el encanto de PSP, su multifuncionalidad y acoplamiento a cualquier exigencia.

Cabe recalcar que lo anteriormente mencionado, depende de la habilidad individual de los ingenieros. La clave para aprender la metodología del proceso personal de *software* (PSP) está en leer y analizar los datos del trabajo que se desarrollará y lo que en realidad estos datos “dicen” acerca del desempeño personal. Este proceso se detalla adelante.

PSP también muestra cómo aplicar métodos avanzados de ingeniería a sus proyectos y/o deberes diarios. Asimismo, provee métodos de estimación y de planeación muy bien detallados, necesarios para dar seguimiento a su trabajo. A medida que el trabajo mejora, también tiende a mejorar la calidad de los proyectos.

El modelo PSP está dividido en niveles, implantados de manera incremental. Los niveles superiores suman características a los niveles ya implantados, lo que minimiza el impacto de los cambios en los hábitos del desarrollador. Este deberá tan sólo adaptar nuevas técnicas ya existentes y conocidas. Lo más importante en el proceso de aprendizaje es la información recogida después de cada fase, cuya información obtenida en la fase actual, servirá para mejorar el desempeño personal para la siguiente fase.

Los principios de PSP son: mejorar las estimaciones, mejorar el planeamiento y cronogramas, crear un compromiso personal con la calidad, aumento de la calidad a través de la reducción de la incidencia de errores, mayor precisión en las estimaciones de tamaño del *software* y tiempo de desarrollo.

OBJETIVOS

- **General**

  Realizar una investigación y un análisis que provea un panorama general sobre lo que es el proceso personal de *software* (PSP), desde sus principios hasta la actualidad

- **Específicos**

1. Ayudar al ingeniero de *software* desarrollar habilidades y hábitos necesarios para planificar y analizar todo tipo proyecto desde una perspectiva personal (mejorar habilidades y disciplina individual)

2. Introducir los métodos y técnicas de PSP para el desarrollo de *software* de calidad.

3. Que el ingeniero logre una mejora en su planeación de trabajo, conocer con precisión el desempeño, medir la calidad de productos y mejorar las técnicas para su desarrollo usando datos personales.

4. Evaluar a las empresas guatemaltecas para observar los estándares que usan.

INTRODUCCIÓN

El P*SP* es una metodología que tiene como disciplina las actividades individuales, para desarrollar *software* de calidad, mediante métodos avanzados de ingeniería, planeación, estimación y seguimiento de su trabajo, que se necesitarán para medir, evaluar y monitorear la manera de hacer sus tareas.

El modelo PSP está divido en niveles, implantados de manera incremental. Los niveles superiores agregan características a los niveles ya desarrollados. Lo más importante es el aprendizaje de la información recogida después de cada fase, la cual nos sirve para mejorar el desempeño personal para la siguiente fase.

En otras palabras:

- es un proceso de mejora personal diseñado para controlar, gestionar y mejorar la forma de trabajo;
- el PSP es una aplicación de las ideas del CMM para individuos;
- se persigue un conjunto de las actividades personales de forma parcial.

El PSP se considera la solución para pasar entre los niveles de CMM al lograr un mejor entendimiento de nuestras capacidades, habilidades y un mejor control sobre nuestro trabajo. Cabe destacar que el PSP puede ser el mejor complemento para cumplir con los requerimientos exige CMM. Esto serviría para hacer la certificarse de la empresa que quiera alcanzar dicha certificación. PSP se puede ajustarse a la medida de otras metodologías ya que al fin es el mismo el ciclo de vida del *software* par0a todos niveles, pues, se ha visto que PSP puede trabajar tanto con a CMM como independiente. El uso de PSP depende del entorno que el desarrollador quiera utilizarlo.

1. PROCESO DE *SOFTWARE* PERSONAL

PSP se concentra en las prácticas de trabajo de los ingenieros en una forma individual. El principio detrás de PSP es ése, sirve para producir *software* de calidad, cada ingeniero debe trabajar en la necesidad de realizar trabajo de calidad.

1.1. Los valores de las personas en la administración del conocimiento

Calidad es estar en conforme con los requerimientos del cliente; anticipar y satisfacer a los clientes. Una de las evoluciones más importantes en el estudio de la calidad está en notar que la calidad del producto es buena, mas la calidad del proceso de producción es todavía más importante. Lo que se debe considerar en la mejora de la calidad del *software* es: administración de la calidad, administración de proyectos, administración de los recursos humanos, Técnicas de Ingeniería de *software* como también administración del conocimiento.

La aplicación de la mejoría de calidad del *software* se caracteriza por la productividad de la empresa. Debe tenerse en cuenta que el problema de calidad no está en el *software* en sí, sino en la forma que se desarrolla el *software*. Actualmente, hay varias instituciones que se preocupan por crear normas para permitir la correcta evaluación de calidad, tanto de productos de *software* como de los procesos de desarrollo de *software*. Al garantizar la calidad del proceso, se da un enorme paso en la garantía de calidad del producto.

La calidad total comienza por predicar la necesidad de un cambio del modelo mental. La calidad total inicia con la administración del conocimiento más allá de información y tecnología.

Las empresas actuales están integradas y orientadas a los procesos. Se busca las actividades ejercidas por cada persona, con autonomía, responsabilidad, transparencia de acciones y orientación para el crecimiento personal a través del conocimiento y de las nuevas habilidades. El objetivo es la persona, con lo que se buscan resultados globales a través de resultados personales. El conocimiento humano, da resultados globales, es el mayor factor competitivo. La empresa que tiene el conocimiento domina el mercado.

La habilidad es la utilización del conocimiento para agregar valor para las personas; trabajar es practicar y cuanto más diversificadas sean las habilidades de una persona, mayor será su índice de contratación.

Los desafíos llevan a las personas a buscar recursos de conocimiento y habilidad para superarlos, por lo tanto, establecer metas cuyo desafío es superior a la capacidad actual conduce a nuevos conocimientos. Invertir en tiempo y esfuerzo, así como tener la visión de que todo trabajo es un medio de generación de nuevos conocimientos y por tanto, de crecimiento personal, es la manera de pensar del profesional moderno.

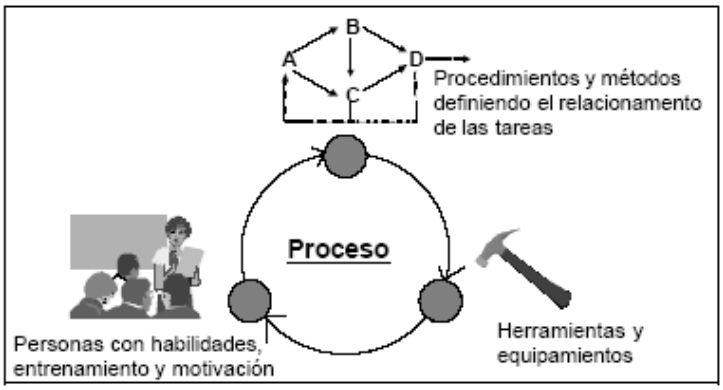
Todo trabajo exige habilidades, que provienen del conocimiento, principal factor de sobre vivencia de los individuos; tener disciplina proviene de la autonomía y de la responsabilidad; tener disciplina es tener un proceso personal y seguirlo, sin dejar de lado los conceptos de mejoría continua.

El valor de la persona en la era del conocimiento implica, para cada persona, la búsqueda de aprendizaje constante, compartimiento de conocimiento, innovación, creatividad, emprendimiento y creación de un proceso personal.

1.2. Conociendo el proceso de las actividades del trabajo

Proceso: es un conjunto de actividades del trabajo (tiempo, espacio) y de medidas destinadas a un proyecto específico para un cliente, como se muestra en la figura 1.

Figura 1 Proceso de trabajo



Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

En las organizaciones desarrolladoras de *software* por lo general hay varios proyectos al mismo tiempo, con equipos de trabajo que utilizan herramientas y métodos de trabajo que varían proyecto a proyecto con la experiencia del equipo, del líder y las características de cada proyecto. Con lo que lleva a que cada proyecto se trabaje de diferente forma, la cual la empresa a reorganizar los equipos, dándose problemas de de adaptación y re aprendizaje.

La definición de un proceso de patrón de desarrollo busca eliminar esos problemas, porque todos trabajan bajo las mismas reglas y conceptos, que se define quién hace, lo que hace, cuándo y cómo para alcanzar el objetivo.

1.2.1. Creando la base para el proceso personal de las actividades del trabajo

Producir productos de calidad, no sobrepasar el costo esperado y realizar el trabajo según el plan acordado son algunos de los aspectos de un trabajo de calidad. Para realizar un servicio eficaz es necesario:

1. planificar el producto;
2. hacer el producto conforme con el plan;
3. producir productos de calidad.

Cuando no existe un proceso personal en el trabajo, hay un desperdicio de tiempo y aumenta el riesgo de efectividad del trabajo.

Para mejorar el trabaja tenemos los siguientes pasos:

1. definir la calidad del producto
2. conocer el proceso de trabajo a ser utilizado;
3. ajustar del proceso;
4. hacer uso del proceso ya ajustado;
5. analizar los resultados;
6. evaluar los resultados;
7. reinicie y continúe optimizando el proceso.

Cuanto más complejo sea el proyecto, el producto, el proceso, más fases serán necesarias. Con un proceso sofisticado el ciclo de mejora continua, nunca debería tener fin, pues siempre es posible de mejorar.

1.2.1.1. Usando mejor el tiempo

Para generar el tiempo se puede, ser tratada como:

- las actividades tienden a repetirse entre tiempo en tiempo, el espacio que hay entre tiempos hay actividades no previstas,

- para hacer un plan real, se debe registrar como se ocupa el tiempo, porque las personas por lo general recuerdan algunas cosas y olvidan otras, para saber adónde se está utilizando el tiempo, se debe tener un registro del tiempo,

- para hacer las validaciones y correcciones de los tiempos y planes, se debe documentar, para ver que ocurre realmente,

- realizar evaluaciones del plan, para hacerlo mejor;

- Para generar el tiempo, planifique y siga un plan para perder menos tiempo.

El formato y el procedimiento usados para guardar los datos del tiempo se verán mas adelante. El registro de su tiempo es útil para analizar desvíos en su planificación.

1.2.1.2. Planeamiento de tiempo y producto

Hay dos tipos de planeamiento:

- el primer planeamiento nos dice cómo utilizar el tiempo, durante un rango.
- el segundo se basa en las actividades o productos. como programas.

Un plan de producto incluye:

- tamaño y características del producto,
- tiempo estimado para realizar el trabajo,
- una proyección del planeamiento.

1.2.1.3. Tamaño de los productos

Las actividades varían en tamaño y complejidad es útil tener un medio de comparar sus tamaños. Para cada actividad a ser realizada se debe evaluar, cuál es la medida a definir el tamaño como el tipo de trabajo. Es importante es que exista la medida para la actividad a ser realizada y definir cuál es la mejor y debe ser parte del planeamiento.

1.2.1.4. Administrando el tiempo

Para la administración de tiempo tenemos que tomar en cuenta:

- decidir como se quiere utilizar el tiempo,
- hacer un cronograma,

- utilizar el tiempo de acuerdo con el cronograma.

El cronograma es su plan de actividades por tiempo utilizado. El objetivo para la administración de tiempo, es balancear gradualmente la manera de utilizar el tiempo estableciendo prioridades.

1.2.1.5. Administración de compromisos

Un compromiso es personal y contractual, y requiere de un voluntario acuerdo entre sus dos o más partes, en:

- ¿Qué va a ser?
- ¿Cuál es el criterio para determinar lo que es hecho?
- ¿Quién lo hará?
- ¿Cuándo será hecho?
- ¿Qué se dará en el retorno?
- ¿Quién proporcionará ese retorno?

Para tener la seguridad de que los compromisos están bien administrados se tiene que tomar en cuenta:

- analizar el servicio antes de acordar el compromiso,
- tener apoyo del compromiso con un plan,
- documentar el compromiso (contratos).

Las principales razones para administrar los compromisos es no descuidar y no olvidar ninguno de ellos. Los pasos para administración de compromisos son:

1. hacer una lista de sus compromisos actuales;
2. incluir lo que debe hacer y cuándo;
3. incluya una estimación de cuánto trabajo tiene cada compromiso.

1.2.1.6. Administrar cronogramas

Un cronograma es un listado de actividades planificadas en el tiempo. En el plan de un proyecto es importante dividir el trabajo en varias partes para cada una de ellas pueda ser estimada y planeada. Cada actividad, debe ser tratada como un ítem del cronograma. Un punto de verificación es un punto bien identificado en el proyecto.

1.2.1.7. El plan del proyecto

El Plan del proyecto define el trabajo y cómo será hecho. Contiene una definición para cada una de las tareas, una estimación de tiempo y recursos requeridos, una estructura de trabajo para revisiones y controles. Es un poderoso vehículo de aprendizaje. Cuando está documentado, puede ser utilizado para compararlo con el rendimiento actual. La comparación permite a quien planifica evaluar sus errores y mejorar planeamientos futuros.

1.3. Principios de PSP

El diseño de PSP se basa en los siguientes principios de planeación y de calidad.

- Un proceso definido y estructurado puede mejorar la eficiencia en el trabajo

- El proceso personal definido debe encajar en las habilidades y preferencias del individuo
- Los individuos se deben involucrar en la definición de su proceso
- El proceso debe evolucionar según evolucionan sus habilidades y capacidad
- La mejora continua del proceso se mejora gracias a una retroalimentación rápida y explícita

Para hacer un trabajo de ingeniería de *software* de la manera correcta, los ingenieros deben planear el trabajo antes de comenzarlo. Para que los ingenieros produzcan productos de calidad, deben planear, medir y deben centrarse en la calidad desde el principio.

1.4. Conceptos asociados al proceso de *software* personal

Entre los conceptos que se asocian al proceso de *software* personal tenemos.

1.4.1. ¿Qué es ingeniería de *software*?

Es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del *software*; es decir, la aplicación de ingeniería al *software*.

1.4.2. ¿Qué es proceso?

El proceso define un marco de trabajo para un conjunto de áreas clave de proceso (ACP's) que se deben establecer para la entrega efectiva de la tecnología de la ingeniería del *software*

1.4.2.1. ¿Qué es un proceso de *software*?

Un proceso de *software* proporciona la estructura desde la que se puede establecer un detallado plan para el desarrollo del *software*. Un pequeño número de actividades estructurales se puede aplicar a todos los proyectos de *software*, sin tener en cuenta su tamaño o complejidad.

1.4.2.2. ¿Qué es calidad?

La calidad de *software* significa diferentes cosas para distintos grupos. Para la IEEE la calidad de *software* es el grado en que un sistema, componente o proceso cumple con los requerimientos especificados y con las necesidades o expectativas del cliente o usuario. En la definición de la norma ISO 9000, la calidad de *software* es el grado (pobre, bueno o excelente) en que un conjunto de características inherentes del *software* cumplen con los requisitos.

En conclusión: la calidad de *software* se define como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo *software* desarrollado profesionalmente.

1.4.2.2.1. ¿Qué es concordancia?

Es el grado de cumplimiento de las especificaciones de diseño durante su realización.

1.4.2.3. ¿Qué es proceso de calidad?

Es un marco de trabajo que se requiere para construir *software* de alta calidad.

1.4.3. ¿Qué es Estimación?

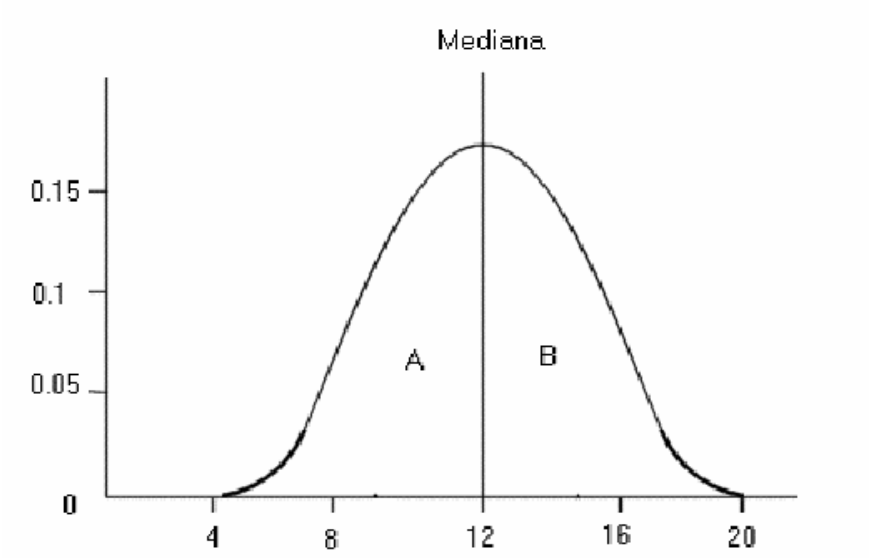
Es importe pensar en una predicción como en un rango más que como un simple número. Una estimación o predicción no es un objetivo, sino una valoración probabilística. El valor que se obtiene de una estimación es el centro del rango.

DeMarco fue uno de los primeros en explicar que una estimación es una predicción que es igualmente probable que esté por encima o por debajo del resultado real. Para comprender lo que esto significa, consideremos una situación en la cual queremos estimar (desde un conjunto de requerimientos) el tiempo que nos llevará finalizar un proyecto. Imaginemos que pudiéramos registrar los valores reales de finalización de un gran número de proyectos, que hayan implementado el mismo conjunto de requerimientos. Entonces podríamos ser capaces de dibujar la función de densidad del tiempo t necesario para finalizar el proyecto con los requerimientos establecidos; tenemos un ejemplo en la figura 2. La función de densidad en la figura 2 es una distribución normal, pero no necesariamente ha de ser siempre así. Desde este gráfico podemos calcular la probabilidad de que cualquier proyecto basado en los requerimientos

dados será finalizado en un intervalo de tiempo [t1, t2]; la probabilidad es simplemente el área bajo la curva, entre t1 y t2. Por ejemplo, la probabilidad de que el proyecto se complete entre los meses 8 y 16 es aproximadamente 0,9, y la probabilidad de que el proyecto sea realizado en menos de 12 meses es de 0,5. Normalmente, estaremos interesados en que la probabilidad de que un proyecto pueda ser finalizado en un tiempo t, es decir el intervalo será [0, t].

Formalmente una estimación se define como la mediana de una distribución. De este modo, la estimación debe dividir el área bajo la curva en dos partes iguales: parte A y parte B (figura 2). En otras palabras, el valor real es la probabilidad de ser menor o mayor que la estimación.

Figura 2. **El área bajo la curva**



Fuente: David Cerrillo
Planificación y Gestión de Sistemas de Información.

Los ingenieros de software frecuentemente entienden mal el concepto de estimación, considerándole, en el ejemplo, como el número más pequeño de meses en el cual el proyecto puede ser completado. Esta interpretación apunta al mínimo valor t para el cual un intervalo [t, t + e] tiene una probabilidad distinta de cero (esto es, el valor mas a la izquierda de la curva, distinto de cero). Este valor mínimo es el tiempo más corto en el cual se puede completar el proyecto. En la figura este valor es aproximadamente cuatro.

Incluso cuando el significado de la estimación se entiende correctamente, muchos ingenieros de software crean estimaciones y las establecen como objetivos. Mediante la designación de la mediana de la distribución de probabilidad como el objetivo, sabemos que hay un 50% de probabilidad de que el proyecto dure más tiempo. De este modo hay el 50% de probabilidad de que no se alcance el objetivo. Un estudio realizado por Jeffery y Lawrence encontró que la productividad era peor en proyectos donde fueron establecidos objetivos. Los gestores de proyectos entienden que la estimación es el centro del rango en el cual el proyecto puede ser completado, y estudian la probabilidad de los casos en los que el proyecto puede tomar menos o mas tiempo que el estimado.

De este modo, para indicar el rango, la estimación se ha de presentar como una tripleta: el valor mas probable (que es, la mediana de la distribución), mas los límites superior e inferior del valor. En términos estadísticos a estos límites inferior y superior se les denomina intervalos de confianza.

1.4.3.1. ¿Qué es estimación de PROBE?

PROBE (PROxy Based Estimating) por sus siglas en inglés, traducido al español se entiende como estimación basada en la evaluación. Es para realizar las estimaciones, tanto del tamaño del programa como de los recursos del mismo.

El método PROBE defiende el uso de medidas seleccionadas personalmente y modelos de regresión basadas en datos personales para estimar el tamaño de un producto *software*. Un *Proxy* es una medida de tamaño que puede ser utilizada para estimar la longitud de un proyecto medida en líneas de código.

Por ejemplo PROBE utilizar una cuenta de objetos como un Proxy de medida de tamaño. La estimación de líneas de código se utiliza para predecir el esfuerzo individual para desarrollar el esfuerzo. Los límites superior e inferior del intervalo de predicción deben ser también estimados.

Se utilizan modelos de regresión simples para estimar las líneas de código desde las medidas de tamaño basadas en Proxy's y las horas de trabajo desde las líneas de código. Se realiza una vuelta atrás desde cada estimación, comparando el valor estimado con el real. La vuelta atrás se introduce para mejorar el rendimiento de la estimación individual.

1.4.4. ¿Qué es planificación?

En realidad, la planificación de un proyecto de *software* no difiere de la planificación de cualquier proyecto de ingeniería. Se identifica una serie de tareas del proyecto. Se establecen interdependencias entre las tareas. Se estima el esfuerzo asociado con cada tarea.

Se hace la asignación del personal y de otros recursos. Se crea una red de tareas. Se desarrolla una agenda de fechas. La planificación temporal para proyectos de desarrollo de *software* puede verse desde dos perspectivas bastante diferentes. En la primera, la fecha final de lanzamiento del sistema basado en computadora ya ha sido (irrevocablemente) establecida. La organización del *software* se ve forzada a distribuir el esfuerzo dentro del marco prescrito. El segundo enfoque de la planificación temporal del *software* asume que se han estudiado unos límites cronológicos aproximados pero que la fecha final es fijada por la organización del *software*.

El esfuerzo se distribuye para hacer un mejor uso de los recursos y la fecha final se define después de un cuidadoso análisis del elemento del *software*. Desafortunadamente a la primera perspectiva se encuentra bastante más a menudo que la segunda.

En conclusión: es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto, asignando el esfuerzo a las tareas específicas de la ingeniería del *software*.

1.4.5. ¿Qué son las KPA's?

Las KPA´s son las áreas de procesos clave o *Key Process Áreas* por su significado en inglés, estas áreas ayudan a guiar a los programadores a que exista un mejoramiento notable en el proceso de *software*

1.4.5.1. ¿Qué son características comunes?

Son atributos que indican si la implementación e institucionalización de una KPA son eficaces, repetidas y duraderas.

1.4.5.2. ¿Qué son prácticas clave?

Éstas describen la infraestructura y actividades que contribuyen para la implementación e institucionalización efectiva de la KPA.

1.4.6. ¿Qué es CMM?

El modelo "clásico" en el tratamiento de la capacidad de los procesos de desarrollo y su madurez es CMM (*Capability Maturity Model*) del SEI (*Software Engineering Institute*). CMM tiene como objetivo evaluar los procesos en sus distintos niveles de madurez, identificar los niveles a través de los cuales una organización debe formarse para establecer una cultura de excelencia en la ingeniería de *software*. El modelo de madurez de procesos fue generado a través de la experiencia colectiva de los proyectos más exitosos de *software*, generando así un conjunto de prácticas importantes que deben ser implantadas por cualquier entidad que desarrolla o mantiene *software*.

En particular, CMM es un marco de trabajo especificando guías para organizaciones de *software* que quieren incrementar su capacidad de procesos, considerando los siguientes puntos:

- Identificar fortalezas y debilidades en la organización.
- Identificar los riesgos de seleccionar entre diferentes contratos y monitorear los mismos.
- Entender las actividades necesarias para planear e implementar los procesos de *software*.
- Ayudar a definir e implementar procesos de *software* en la organización a través de una guía.

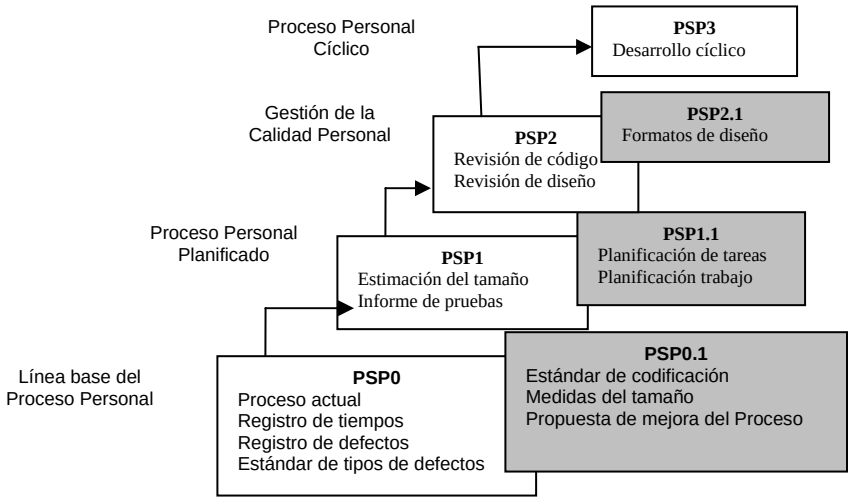
El modelo de madurez de la capacidad del proceso de *software* (CMM) nos permite determinar la capacidad de las organizaciones de desarrollo de *software* para producir de manera consistente y predecible productos de calidad superior.

Y el modelo nos brinda guías para seleccionar estrategias de mejoramiento del proceso mediante la determinación de las capacidades actuales del proceso y la identificación de los puntos críticos para mejorar el proceso y la calidad del *software*

1.5. Niveles de PSP

La figura 3 muestra los niveles de PSP con sus propios requerimientos, para poder implementar PSP deben de cumplir con los requerimientos de cada nivel, para poder subir de nivel. PSP se puede personalizar según las necesidades de cada desarrollador u organización.

Figura 3. Evolución del proceso de *software* personal



Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

La información de la figura anterior es de ayuda al programador a tener una idea de los puntos que se deben cubrir en cada uno de los niveles de PSP en resumen se tendría:

- Línea Base del Proceso Personal (PSP0):
  - se incluyen medidas e informes básicos
  - base para la mejora
- Proceso Personal Planificado (PSP1):
  - se documentan los planes para:
    - comprender la relación entre tamaño y tiempo
    - comprometerse con lo que se puede cumplir
    - establecer un plan ordenado para trabajar
    - determinar el estado del trabajo
- Gestión de la Calidad Personal (PSP2):
  - tratar objetivamente los defectos
  - 5% - 20% defectos sin revisión
  - gestionar defectos para encontrarlos antes
    - PSP2.1: verificación y consistencia

- Proceso Personal Cíclico (PSP3):

  - subdividir un programa grande en piezas del tamaño PSP2
  - se establece un núcleo base que se mejora en ciclos iterativos
  - integración de incrementos de alta calidad
  - regresiones: Los test de prueba anteriores son fundamentales en la integración

2. NIVELES DEL PROCESO DE *SOFTWARE* PERSONAL

PSP hace uso de un gran número de formatos y *scripts*, los cuales son útiles para hacer el análisis del programa ha desarrollará entre los pasos tenemos:

- definir el programa.
- decidir que método se utilizara para resolverlo
- aplicar la solución
- ver si es la solución correcta.
- evaluar y corregir los errores encontrados
- entregar la solución

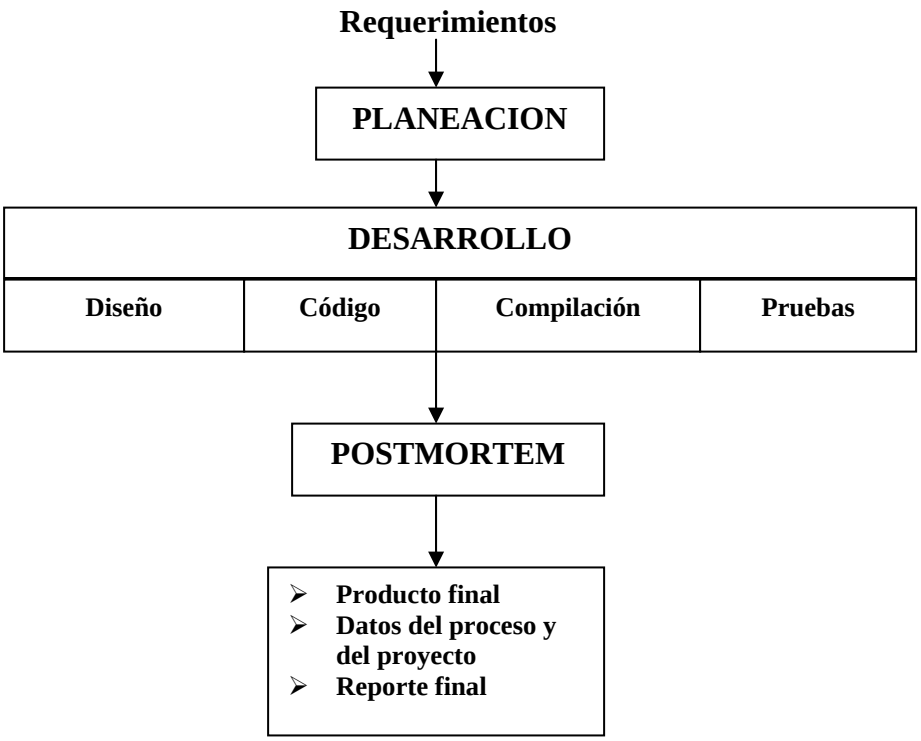
Cuando se tiene bien estructurados los pasos anteriores, sólo se llenan los formatos del nivel inicial de PSP y los cambios de las fases serán más notables.

2.1. Nivel PSP 0 y PSP 0.1

Comenzando con los requerimientos, el primer paso en el proceso PSP es la planeación. Hay un guión de planeación que guía este trabajo y un sumario del plan para registrar los datos de la planeación. Mientras los ingenieros están siguiendo el guión para hacer el trabajo, registran sus datos de tiempo y defectos en los formularios para tiempo y defectos. Al final del trabajo, durante la fase postmortem, ellos hacen un resumen de los datos de tiempo y defectos de los formularios, y de las medidas del tamaño del programa e incluyen esos datos en el formulario del resumen del plan.

Cuando el trabajo está terminado ellos entregan el producto terminado junto con el formulario del plan completado, todo esto se muestra más específico en la figura 4.

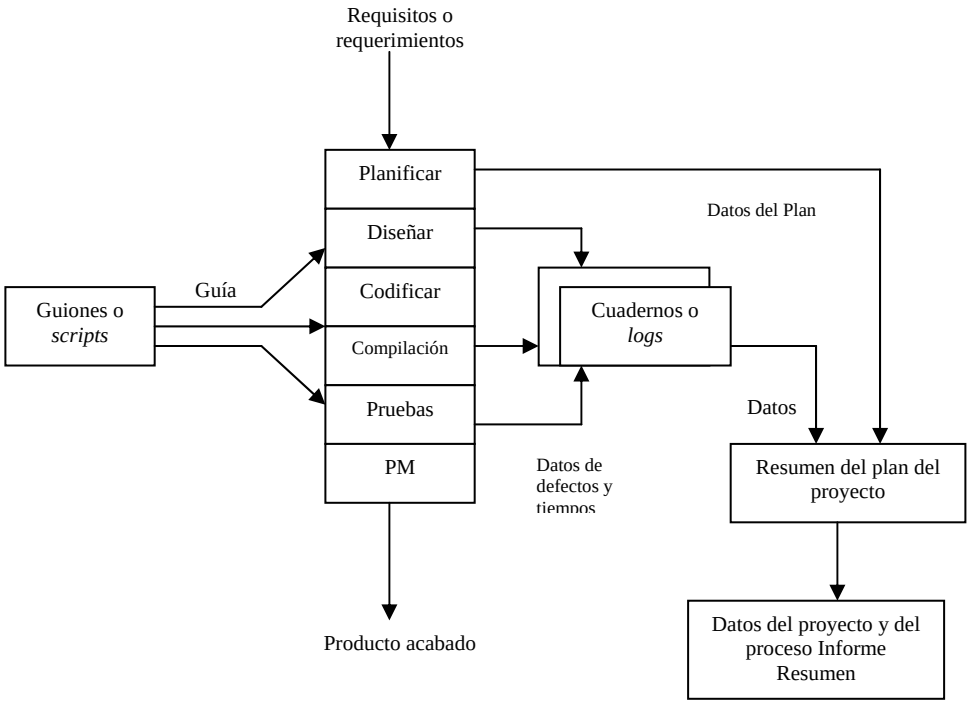
Figura 4. El proceso PSP 0



Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Para realizar la captura de la información del proyecto es importante que se tomen todos los requerimientos. El flujo del proceso inicial está en la figura 5.

Figura 5. Flujo de PSP 0



Fuente: Dr. Leopoldo Altamirano Robles.
Metodologías de diseño

PSP 0 posee unas plantillas o formatos para almacenar la información recaudada, que ayudan a guiar al desarrollador durante la plantación del proyecto en todos sus niveles y es necesario llevar un resumen del plan, para registrar toda la información.

2.1.1. Línea base del Proceso (PSP0)

En las fases que PSP requieren de un *script,* para el ciclo de vida.

- El *script* de planeación tiene como entrada los requerimientos del programa y la estimación de los recursos a utilizarse.
- El *script* de desarrollo le pide al desarrollador los requerimientos de cada fase diseño, desarrollo, compilación y pruebas, para obtener al final un programa a prueba de errores.
- El *script* de p*ost-mortem* le pide al desarrollador un listado de todos los defectos que se encontraron y el tiempo de corrección, todo esto es para tener un historial de los posibles defectos para no caer en los mismos.

PSP estima el tamaño y recursos de los productos que se desarrollan con un método llamado PROBE, que consiste en que los desarrolladores deben determinar, los objetos que se requieran para realizar el producto.

Haciendo uso de la información histórica podemos determinar el tamaño de los módulos, para poder hacer el cálculo del tamaño del producto por medio de la regresión lineal.

Esto se realizar por medio de una comparación de la información históricos contra lo estimado del programa actual, se debe realizar la comparación con al menos tres programas desarrollados anteriores. Los resultados de la correlación que PSP requiere del tamaño y el tiempo debe ser al menos 0.5.

Haciendo uso de los porcentajes que se encuentra en el formato de registro de tiempo, el desarrollador puede estimar el tiempo total de las fases de planeamiento, diseño, revisión de diseño, código, revisión de código, compilación, pruebas y *post-mortem* para tener tiempo mas real de desarrollo y el tiempo que se le dedicara a cada tarea como la de pruebas y corrección.

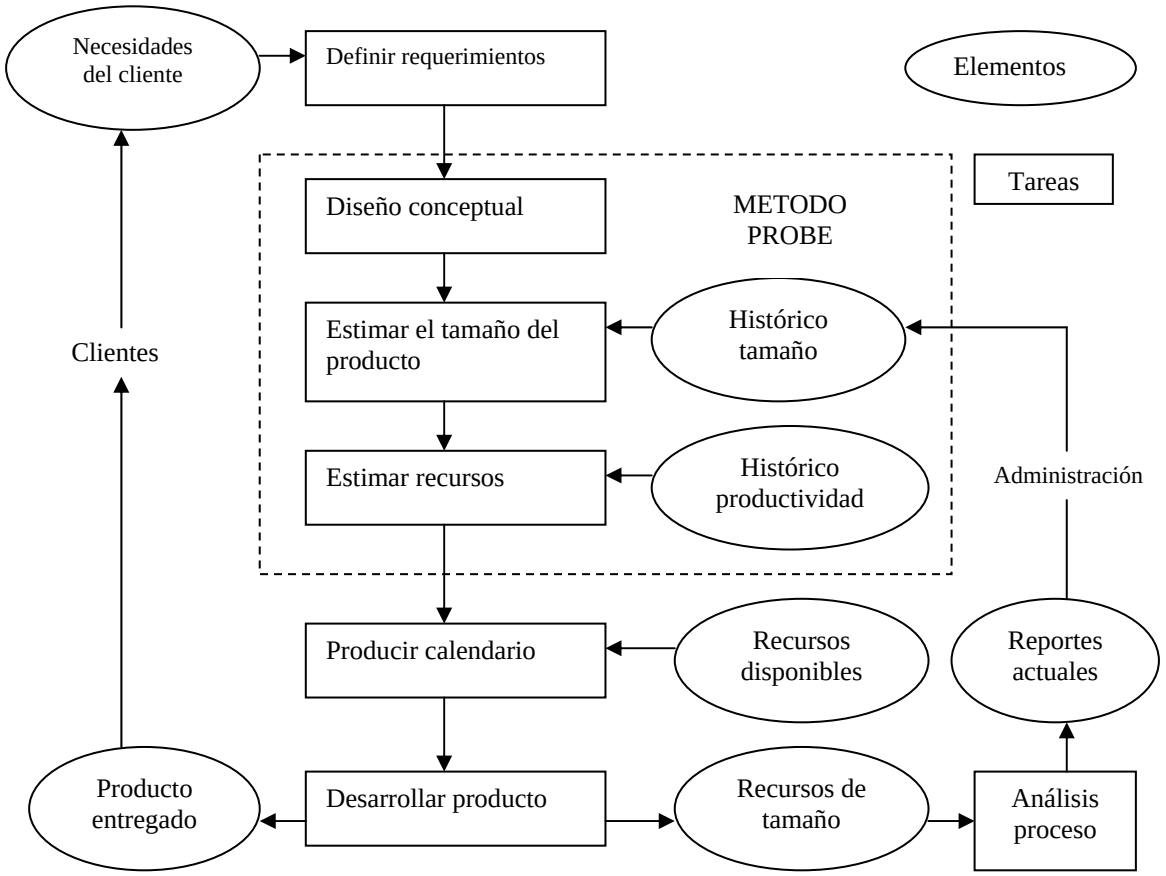
Para aprender el proceso de *software* personal, está en analizar la información recauda en proyectos ya desarrollados, para ver como es el desempeño personal de cada desarrollador, en la tabla I lleva a cabo la medición personal en los niveles PSP, conjuntamente en la figura 6 se muestra la relación que existe con el cliente.

Tabla I. Medición personal de PSP

Fase	Objetivo	Guía al desarrollo de programas, a nivel modular.
	Entrada requerida	➢ Descripción del problema ➢ Formulario resumen del plan de proyecto PSP0 ➢ Cuadernos de registro de tiempos y defectos ➢ Estándar de tipos de defectos ➢ Reloj de control (opcional)
1	**Planificación**	➢ Obtener o producir requisitos para el programa ➢ Estimar el tiempo de desarrollo requerido ➢ Introducir los datos del plan en el formulario Resumen del plan de proyecto ➢ Rellenar el cuaderno de registro de tiempos
2	**Desarrollo**	➢ Diseñar el programa ➢ Implementar el diseño ➢ Compilar el programa y fijar y registrar los defectos ➢ Probar el programa y fijar y registrar los defectos ➢ Rellenar el cuaderno de registro de tiempos.
3	**Post-mortem**	➢ Completar el formulario resumen del plan de proyecto con datos de tiempo, defectos y tamaño reales.
	Criterio de salida	➢ Un programa probado ➢ El resumen del plan de proyecto completo con datos estimados y reales ➢ Cuadernos de registro de tiempos y defectos completados.

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Figura 6. Marco de planeación de proyectos



Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

2.1.2. Línea base del proceso de *software* personal

Los formatos que utilizan PSP 0 y PSP 0.1 se basan en dos medidas importantes que son:

- el tiempo empleado en cada nivel y
- los defectos encontrados en cada nivel.

El formato del registro de tiempo se muestra en la figura 7 y según se valla avanzando de nivel se pueden agregaran más campos.

Figura 7. Formato de registro de tiempo

Programador _____ Fecha _____
Jefe Proyecto_____ Programa # _____

Fecha	Inicio	Termino	Tiempo de interrupción	Tiempo delta	Fase	Comentario	C	U

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

El objetivo de este formato es el de registrar el tiempo empleado en cada nivel del proyecto, que debe ser en minutos.

Los campos que forman éste formato son:

- Encabezado: la información que se debe ser introducida debe de ser el nombre del programador, la fecha, jefe de proyecto y un correlativo que le corresponde al programa a desarrollar.
- Fecha: la fecha que se introdujo la información.
- Inicio: la hora de inicio a trabajar en el producto.
- Término: la hora que se deja de trabajar en el producto.

- Tiempo de interrupción: el tiempo que no se emplea en trabajar en el producto.

- Tiempo delta*:* es el tiempo de la resta del tiempo empleado en el producto menos el tiempo de interrupción.

- Fase: el nombre de la fase en la que se trabaja.

- Comentarios: se hace todo tipo de comentario útil que puedan recordar ciertas circunstancias.

- C (completado): se marca esta columna cuando se termina una tarea,..

- U (unidades): el número de unidades de una tarea acabada.

El formato de registro de defectos se muestra en la figura 8, que es una base importante de información para la estimación del tiempo del proyecto para no caer en los mismos errores.

Figura 8. Formato de registro de defectos

Tipos de Defectos

Código	Nombre
10	Documentación
20	Sintaxis
30	Construcción
40	Asignación
50	Interfaz
60	Chequeo
70	Datos
80	Función
90	Sistema
100	Ambiente

Programador _____ Fecha _____
Jefe proyecto _____ 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:

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

El objetivo del formato de registro de defectos es llevar un registro los defectos.

Los campos que intervienen en este formato son:

- Fecha: la fecha cuando se encuentro el defecto.
- Número: correlativo de defectos encontrados en cada programa.
- Tipo*:* se refiere al tipo de defecto en la tabla adjunta al formato.
- Encontrado*:* el nombre de la fase, cuando se encontró el defecto
- Removido: al nombre de la fase cuando se removió el defecto encontrado.
- Tiempo de compostura: el tiempo para reparar el defecto encontrado.
- Defecto arreglado: este campo quiere decir, si se encontró algún defecto extra mientras se reparaba algún defecto, si no hubiera se coloca una X.

En la figura 9 se muestra el formato del resumen del plan del proyecto, en el cual contiene la información de las estimaciones y los datos reales del producto, para hace las comparaciones necesarias y exista un historial de todos los proyectos.

Figura 9. Resumen del plan del proyecto

Programador _____ Fecha _____
Programa _____ Programa # _____
Jefe proyecto_____ Lenguaje _____

Tiempo en la fase (min.)	**Plan**	**Actual**	**A la fecha**	**A la fecha %**
Planeación				
Diseño				
Codificación				
Compilación				
Pruebas				
Post-mortem				
Total				

Defectos encontrados	**Actual**	**A la fecha**	**A la fecha %**
Planeación			
Diseño			
Codificación			
Compilación			
Pruebas			
Total desarrollo			

Defectos removidos	**Actual**	**A la fecha**	**A la fecha %**
Planeación			
Diseño			
Codificación			
Compilación			
Pruebas			
Total desarrollo			
Después de desarrollo			

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Este formato nos sirve de respaldo para cada proyecto que se desarrolla, ya que se encuentra la información que es útil para los siguientes proyectos semejantes a este.

Los campos del formato resumen del plan del proyecto son:

Área de tiempo en la fase.

- Plan*:* es el tiempo que se empleará para desarrollar el proyecto
- Actual: es el tiempo real, que se emplea en cada una de las fases de desarrollo.
- A la fecha: la suma del tiempo actual, con el tiempo del último programa que se desarrollado.
- A la fecha %: es el porcentaje del tiempo que se emplea en cada fase de desarrollo.

Área de defectos encontrados.

- Actual: número de defectos encontrados en cada fase de desarrollo.
- A la fecha: suma de los defectos encontrados, con el campo de A La Fecha del último proyecto.
- A la fecha %: es el porcentaje de defectos encontrados del campo A La Fecha.

Área de defectos removidos.

- Actual: el número de defectos removidos en cada fase.
- A la fecha: es la suma de los defectos removidos, con el valor del campo A La Fecha del último proyecto.
- A la fecha %: porcentaje de defectos removidos del campo A La Fecha.

2.2. Nivel PSP 1 y PSP1.1

A partir del nivel 1, el programador se encuentra en la fase de planeación que abarca hasta el nivel 1.1.

2.2.1. Nivel PSP 1

Con los niveles de PSP se busca cual se puede adaptar a las necesidades del proyecto, ya reconocido este nivel, se puede hacer uso de los formularios y *scripts* de cada nivel.

En el nivel de PSP 1, incluye los pasos de la revisión del diseño y del código, los cuales se realizan personalmente en cada producto, antes de las fases compilación y de pruebas. El objetivo de la revisión del diseño es que el programador no tenga los mismos errores ya cometidos. Para no caer en esto se debe realizar un análisis de la información de los defectos y una lista de las acciones para encontrarlos.

La mejor manera de manejar los defectos es la prevención, entre las maneras que PSP previene los defectos, tenemos:

- consiste en tener personal que se dediquen a registrar la información de cada defecto y después a arreglarlos.
- otra forma es con un método de diseño y una notación eficaz del diseño final. Para registrar totalmente un diseño, los desarrolladores deben entenderlo bien.

El objetivo de PSP 1 es tener un registro de las pruebas hechas y los resultados obtenidos. Entre los formularios que utilizan en PSP 1, se encuentra

el formato de reporte de pruebas, que sirve para mantener un registro de las pruebas que se realizaron a cada programa y los resultados de estas, esto nos sirve para tener una mejor estimación del tamaño del programa y llevar un registro de pruebas para futuras pruebas. La figura 10 se muestra el formato de reporte de pruebas.

Figura 10. Formato de reporte de pruebas

Programador _____ Fecha _____
Jefe proyecto_____ Programa #_____

Prueba nombre/número _____
Prueba objetivo _____
Prueba descripción _____
Condiciones prueba _____
Resultados esperados _____
Resultados actuales _____
Prueba nombre/número _____
Prueba objetivo _____
Prueba descripción _____
Condiciones prueba _____
Resultados esperados _____
Resultados actuales _____

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Los campos que conforman el formato de reporte de pruebas son:

- Prueba nombre/número: indica el nombre de la prueba de cada programa.
- Prueba objetivo: una descripción del objetivo de la prueba.
- Prueba descripción: describe los datos de cada prueba y procedimientos que se aplicaron, para reutilizarla.

- Condiciones prueba*:* se describe alguna configuración especial y arreglo de la prueba.

- Resultados esperados: se hace una lista de los resultados que se esperan de la prueba.

- Resultados actuales: son los resultados que produjo la prueba.

La figura 11 trata de la estimación del tamaño y para mantener un registro de los datos estimados. Asiendo uso de fórmulas que sirven para calcular parámetros, para hacer comparación entre los datos del programa actual, contra datos de programas históricos.

Figura 11. Formato de estimación de tamaño

Programador_____ Fecha: _____
Jefe proyecto_____ Programa # _____

Programa base LOC

Tamaño base (B) => => => => => => => => => => => => _____
LOC Suprimido (D) => => => => => => => => => => => => _____
LOC Modificado (M) => => => => => => => => => => => _____

LOC Proyectado (P)

Adiciones:	Tipo	Métodos	Tamaño Rel.	LOC
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____

Total adiciones base (BA) => => => => => => => => => _____

Objetos Nuevos:	Tipo 1	Métodos	Tamaño Rel.	LOC (nuevo / rehusado*)
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____

Total objetos nuevos (NO) => => => => => => => _____

Objetos rehusados LOC

_____ _____

Total rehusado (R) => => => => => => => => _____
LOC Proyectado: P = BA + NO _____
Parámetro de regresión: ß0 _____
Parámetro de regresión: ß1 _____
LOC Estimado nuevo y cambiado: N = ß0 + ß1 *(P + M) _____
LOC Estimado total: T = N + B - D - M + R _____
Total Estimado nuevo/rehusado (suma de * LOC): _____
Rango de predicción: Rango _____
Intervalo superior de predicción: UPI = N + Rango _____
Intervalo inferior de predicción: LPI = N - Rango _____

Porcentaje de intervalo de predicción:
1 L-Lógico, I-I/O, C-Calculo, T-Texto, D-Dato

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

El formato se divide en cuatro áreas distintas su descripción es:

- área de programa base: esta área se llena cuando el programa es una modificación de un programa existente: se realiza estimación del tamaño del programa colocando el dato en (B), luego las LOC a ser borradas en (D) y luego las LOC que se modificarán en el campo (M).

- área de adiciones base (BA): si se agregan líneas de código en el programa, entonces se debe de identificar las funciones que se agregarán y estimar si esta función debe de ser tomada como un objeto nuevo.

- área de adiciones base (NO)*:* se asigna el nombre de cada objeto nuevo planeado y estimar el tamaño del objeto, como: (VS) Muy pequeño, (S) Pequeño, (M) Mediano, (L) Grande, (VL) Muy grande.

- área de objetos re-usados: se asigna el nombre de cada objeto que no tuvo modificaciones y que es re-usado. En el campo de "TAMAÑO" se escribe las líneas de código de cada objeto, y se suman los LOC totales en el campo de (R).

El nuevo formato de PSP 1, es planeación de tareas que se muestra en la figura 12 y sirve para guiar al desarrollador una calendarización de las horas empleadas para cada tarea.

Figura 12. Formato de asignación de tareas

Tareas		Plan					Actual		
Numero	Nombre	Horas	Valor planeado	Horas acumuladas	Valor Planeado acumulado	Fecha lunes	Fecha	Valor adquirido	Valor adquirido acumulado
Totales									

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

- Número y nombre*:* se debe asignar un nombre y un número a cada tarea para hacer referencia de ellas, en el orden que se desea que se cumplan las tareas.
- Horas*:* es el tiempo en hora para cada tarea.
- Valor planeado: es la suma de las horas de todas las tareas.
- Horas acumuladas: suma acumulada de las horas de cada tarea.
- Valor planeado acumulado: es la suma de lo planeado de cada tarea.
- Fecha lunes: primero se detectan las tareas que exceden las horas planeadas acumuladas, introduciendo la fecha más cercana que comienza en lunes.
- Fecha: es la fecha real de terminación tarea.

- Valor adquirido*:* para tareas completadas, se escribe el valor planeado.

- Valor adquirido acumulado*:* cada finalización de tarea, se suman los valores adquiridos y se escribe el valor de la tarea que se completó.

2.2.2. Nivel PSP 1.1

El objetivo de PSP 1.1 es mejorar los detalles de la planeación de recursos y el calendario. PSP 1.1 provee las herramientas necesarias para que la planeación sea lo mas relista, por medio de los formatos que tiene.

Existen dos tipos de planeación.

- El basado en rango de tiempo.
- El otro se basa en la actividad a desarrollar.

Ambos dependen uno del otro para ser realizados.

La planeación ayuda a medir cuanto tiempo tomará realizar el proyecto. PSP 1.1 dice que para hacer una planeación se debe tener una definición clara del producto y que hay que tomar en cuenta lo siguiente:

- El tamaño y las características del producto.
- Una estimación del tiempo para realizar el proyecto.
- Una calendarización.

Es necesario realizar un plan a la magnitud y complejidad del proyecto. Esto se puede hacer con la comparación de proyectos pasados, se podrá estimar el tiempo para realizar el proyecto. En la figura 13 se agregan los

campos al formato de PSP, son muy pocos, pero los nuevos campos cumplen un papel esencial en los métodos de análisis de tiempo.

Figura 13. Formato plan del proyecto

Nombre _____ Fecha _____
Programa descripción _____ Programa _____
Lenguaje _____

Resumen	**Plan**	**Actual**	**A la fecha**
LOC/hora			
Tiempo planeado			
Tiempo actual			
CPI (Costo / índice desempeño)			
% Rehusado			
% Nuevo rehusado			

Tamaño programa (LOC)	**Plan**	**Actual**	**A la fecha**
Total LOC (T)			
Base (B)			
Suprimido (D)			
Modificado (M)			
Rehusado(R)			
Añadido (A)			
Total nuevo y cambiado (N)			

Tiempo fase	**Plan**	**Actual**	**A la fecha**	**A la fecha %**
Planeación				
Diseño				
Codificación				
Compilación				
Pruebas				
Post-mortem				
Total desarrollo				
Después desarrollo				

Defectos encontrados	**Actual**	**A la fecha**	**A la fecha %**
Planeación			
Diseño			
Codificación			
Compilación			
Pruebas			
Total desarrollo			

Defectos removidos	**Actual**	**A la fecha**	**A la fecha %**
Planeación			
Diseño			
Codificación			
Compilación			
Pruebas			
Total desarrollo			
Después desarrollo			

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

2.3. Nivel PSP 2 y PSP 2.1

El nivel después de planeación, es el nivel que tiene que ver con la administración de calidad personal que abarca las fases PSP 2 y 2.1. Este nivel induce al desarrollador a realizar revisiones detalladas del código y del diseño.

2.3.1. Nivel PSP 2

Los objetivos de este nivel de PSP son, que el programador debe de realizar revisiones de diseño y de codificación. Cuando ya se hayan hecho las revisiones, el desarrollador debe aplicar los métodos de evaluación para asegurar la calidad.

El método de evaluación que se utiliza en PSP 2 es el siguiente:

- Los datos del proceso deben estar completos.
- Los datos deben ser precisos y consistentes.
- El reporte del proceso se debe de llenar en el orden correcto.
- La información histórica debe de ser utilizada para la fase de planeación.
- La información histórica debe de ser utilizada constantemente para mejorar el proceso.

Se necesita hacer una revisión de la información de los proyectos anteriores antes de hacer la planeación del proyecto. Luego de realizar los cambios debidos, se utilizan los *checklists* para los proyectos futuros.

El orden del reporte es el siguiente:

- Resumen del plan del proyecto (PSP 2).
- Plantilla del reporte de pruebas.
- *Checklist* de revisión de diseño.
- *Checklist* de revisión de código.

- Formato *PIP*.
- Templete de la estimación del tamaño.
- Templete de la planeación de tareas.
- Templete de la planeación del tiempo.
- Formato del registro de tiempo.
- Formato del registro de defectos.
- Listado del código fuente.

Es necesario que cada reporte cuente con una copia de seguridad y de referencia. Los formatos que se utilizan ayudan a listar los procedimientos.

Lo nuevo que se agrega en los formatos de este nivel, es que se utiliza el concepto de *checklists* que son listas, que sirven para asegurar que cada elemento de programación esté en su lugar. La figura 14 muestra el proceso completo de la estimación del tamaño del programa.

Figura 14. El proceso PSP 2

Nombre _____ Fecha _____
Programa descripción _____ Programa #_____
Lenguaje _____

Resumen	**Plan**	**Actual**	**A la Fecha**
LOC/Hora			
Tiempo planeado			
Tiempo actual			
CPI (Costo / índice de desempeño)			
Pruebas defectos/KLOC			
Total defectos/KLOC			
Porcentaje %			

Tamaño Programa	**Plan**	**Actual**	**A la Fecha**
Total LOC (T)			
Base (B)			
Suprimido (D)			
Modificado (M)			
Rehusado (R)			
Añadido (A)			
Total nuevo y cambiado (N)			
Predicción intervalo %			
UPI (Predicción Intervalo Superior)			
LPI (Predicción Intervalo Inferior)			

Tiempo Fase	**Plan**	**Actual**	**A la Fecha**	**A la Fecha %**
Planeación				
Diseño				
Diseño revisión				
Codificación				
Codificación revisión				
Compilación				
Pruebas				
Post-mortem				
Total desarrollo				
Predicción Intervalo %				
UPI (Predicción Intervalo Superior)				
LPI (Predicción Intervalo Inferior)				
Después desarrollo				

Defectos *Injected*	**Plan**	**Actual**	**A la Fecha**	**A la Fecha %**
Planeación				
Diseño				
Diseño revisión				
Codificación				
Codificación revisión				
Compilación				
Pruebas				
Total desarrollo				

Defectos *removed*	**Plan**	**Actual**	**A la Fecha**	**A la Fecha %**
Planeación				
Diseño				
Diseño revisión				
Codificación				
Codificación revisión				
Compilación				
Pruebas				
Total desarrollo				
Después desarrollo				

Eficiencia removiendo defectos	**Plan**	**Actual**	**A la Fecha**
Defectos/hora (Diseño revisión)			
Defectos/hora (Codificación revisión)			
Defectos/hora (Compilación)			
Defectos/hora (Pruebas)			
Defectos/hora (Después desarrollo)			
DRL (DLDR/pruebas)			
DRL (Codificación revisión/pruebas)			
DRL (Compilación / pruebas)			

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

2.3.2. Nivel PSP 2.1

PSP 2.1 este nivel ayuda a asegurar la calidad de desarrollo, por medio de la administración y registro de la información.

Los objetivos PSP 2.1 son los siguientes:

- PSP 2.1 ayuda al programador a reducir el número de defectos que se encuentran en la etapa de diseño.

- PSP 2.1 ayuda a verificar hasta el último detalle de la calidad de los diseños.

Cuando se hayan cumplido los objetivos, el desarrollador se puede estar seguro de la calidad del programa y pasar al siguiente nivel.

Los formatos utilizados son los mismos de PSP 2, pero hay un nuevo templete en *checklist de* revisión de diseño, que es de referencia para los otros templetes, y se sigue el mismo orden que los niveles anteriores. Sugiriendo algunos cambios de cómo agregando algunos nuevos campos, que servirán para que las revisiones, y en la figura 15 se muestra el paso final del proceso que asegura la calidad.

Figura 15. El proceso PSP 2.1

Nombre _____ Fecha _____
Programa descripción_____ Programa # _____ Lenguaje _____

Resumen	**Plan**	**Actual**	**A la Fecha**
LOC/hora			
Tiempo planeado			
Actual tiempo			
CPI (costo / índice de desempeño)			
Pruebas defectos /KLOC			
Total defectos/KLOC			
Porcentaje (%)			
Evaluación COQ (%)			
Margen error COQ (%)			
COQ A/F rango			

***Program Size* (LOC)**	**Plan**	**Actual**	**A la Fecha**
Total LOC (T)			
Base (B)			
Suprimido (D)			
Modificado (M)			
Rehusado (R)			
Añadido (A)			
Total nuevo y cambiado (N)			
Predicción intervalo %			
UPI (Predicción Intervalo Superior)			
LPI (Predicción Intervalo Inferior)			

Tiempo Fase	**Plan**	**Actual**	**A la Fecha**	**A la Fecha %**
Planeación				
Diseño				
Diseño revisión				
Codificación				
Codificación revisión				
Compilación				
Pruebas				
Total desarrollo				
Predicción intervalo %				
UPI (Predicción Intervalo Superior)				
LPI (Predicción Intervalo Inferior)				
Después desarrollo				

Defectos encontrados	**Plan**	**Actual**	**A la Fecha**	**A la Fecha %**
Planeación				
Diseño				
Diseño revisión				
Codificación				
Codificación revisión				
Compilación				
Pruebas				
Total desarrollo				

Defectos *removed*	**Plan**	**Actual**	**A la Fecha**	**A la Fecha %**
Planeación				
Diseño				
Diseño revisión				
Codificación				
Codificación revisión				
Compilación				
Pruebas				
Post-mortem				
Total desarrollo				
Después desarrollo				

Eficiencia removiendo defectos	**Plan**	**Actual**	**A la Fecha**
Defectos/hora (Diseño revisión)			
Defectos/hora (Codificación revisión)			
Defectos/hora (Compilación)			
Defectos/hora (Pruebas)			
Defectos/hora (Después desarrollo)			
DRL (DLDR/pruebas)			
DRL (Codificación revisión/pruebas)			
DRL (Compilación/pruebas)			

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

2.4. Nivel PSP 3

Con el nuevo nivel se introduce una nueva fase, para realizar el proceso personal en una manera cíclica. En la que el desarrollador programa de una manera única.

PSP 3 ayuda al desarrollador a crear programas más grandes en poco tiempo y con menos fallas. Entre los objetivos de PSP 3 son los mismos de PSP 2.1, con solo una diferencia que el desarrollador es capaz de hacer programas mucho mas grandes.

También hay un elemento nuevo que es el resumen cíclico, que puede acumular diversos módulos de implementación de hasta 100 Líneas de código para poder hacer comparaciones con los datos estimados con los reales.

Los formatos de PSP 3 se representados en las figuras 16 y 17. Como también tenemos el de registro de tareas que se muestra en la figura 18 y el de registro del tiempo ce dada tarea y resultados finales se muestra en la figura 19.

Figura 16. El proceso PSP 3

Nombre _____ Fecha _____
Programa descripción_____ Programa #_____ Lenguaje _____

Resumen	**Plan**	**Actual**	**A la fecha**
LOC/hora	_____	_____	_____
Tiempo planeado	_____		_____
Tiempo actual		_____	_____
CPI (costo/índice de desempeño)			_____
Pruebas defectos/KLOC	_____	_____	_____
Total defectos/KLOC	_____	_____	_____
Porcentaje (%)	_____	_____	_____
Evaluación COQ (%)	_____	_____	_____
Margen de error COQ (%)	_____	_____	_____
COQ A/F rango	_____	_____	_____

Tamaño programa (LOC)	**Plan**	**Actual**	**A la fecha**
Total LOC (T)		_____	
Base (B)		_____	
Suprimido (D)		_____	
Modificado (M)		_____	
Rehusado (R)		_____	

Añadido (A)		_____		
Total nuevo y cambiado (N)	_____	_____	_____	
Predicción intervalo %	_____			
UPI (Predicción Intervalo Superior)	_____			
LPI (Predicción Intervalo Inferior)	_____			

Tiempo fase	**Plan**	**Actual**	**A la fecha**	**A la fecha %**
Planeación	_____	_____	_____	_____
Alto-nivel diseño	_____	_____	_____	_____
Revisión alto-nivel diseño	_____	_____	_____	_____
Diseño detallado	_____	_____	_____	_____
Revisión diseño detallado	_____	_____	_____	_____
Codificación	_____	_____	_____	_____
Codificación revisión	_____	_____	_____	_____
Compilación	_____	_____	_____	_____
Pruebas	_____	_____	_____	_____
Post-mortem	_____	_____	_____	_____

Tiempo fase	**Plan**	**Actual**	**A la fecha**	**A la fecha %**
Total desarrollo	_____	_____	_____	_____
Predicción intervalo %	_____			
UPI (Predicción Intervalo Superior)	_____			
LPI (Predicción Intervalo Inferior)	_____			
Después desarrollo		_____	_____	

Defectos encontrados	**Plan**	**Actual**	**A la fecha**	**A la fecha %**
Planeación	_____	_____	_____	_____
Diseño alto-nivel	_____	_____	_____	_____
Revisión diseño alto-nivel	_____	_____	_____	_____
Diseño detallado	_____	_____	_____	_____
Revisión diseño detallado	_____	_____	_____	_____
Codificación	_____	_____	_____	_____
Codificación revisión	_____	_____	_____	_____
Compilación	_____	_____	_____	_____
Pruebas	_____	_____	_____	_____
Total desarrollo	_____	_____	_____	_____

Defectos removidos	**Plan**	**Actual**	**A la fecha**	**A la fecha %**
Planeación	_____	_____	_____	_____
Diseño alto-nivel	_____	_____	_____	_____
Revisión diseño alto-nivel	_____	_____	_____	_____
Diseño detallado	_____	_____	_____	_____
Revisión diseño detallado	_____	_____	_____	_____
Codificación	_____	_____	_____	_____
Codificación revisión	_____	_____	_____	_____
Compilación	_____	_____	_____	_____
Pruebas	_____	_____	_____	_____
Total desarrollo	_____	_____	_____	_____
Después desarrollo		_____	_____	

Eficiencia removiendo defectos	**Plan**	**Actual**	**A la fecha**
Defectos/hora (diseño revisión)	_____	_____	_____
Defectos/hora (codificación revisión)	_____	_____	_____
Defectos/hora (compilación)	_____	_____	_____
Defectos/hora (pruebas)	_____	_____	_____
Defectos/hora (a*fte development*)	_____	_____	_____
DRL (DLDR/pruebas)	_____	_____	_____
DRL (Codificación Revisión/Pruebas)	_____	_____	_____
DRL (Compilación/Pruebas)	_____	_____	_____

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Figura 17. Resumen cíclico de PSP 3

Nombre _____ Fecha: _____
Programa descripción_____ Programa #_____ Lenguaje _____

Tamaño programa (LOC)	**A la Fecha**	**1**	**2**	**3**	**4**	**5**	**Total**
Total LOC (T)							
Base (B)							
Suprimido (D)							
Modificado (M)							
Rehusado (R)							
Añadido (A)							
Total nuevo y cambiado (N)							

Tiempo fase	**A la Fecha**	**1**	**2**	**3**	**4**	**5**	**Total**
DL Diseño							
DL Diseño revisión							
Codificación							
Codificación revisión							
Compilación							
Pruebas							
Total							

Defectos encontrados	**A la Fecha**	**1**	**2**	**3**	**4**	**5**	**Total**
DL Diseño							
DL Diseño revisión							
Codificación							
Codificación revisión							
Compilación							
Pruebas							
Total							

Defectos removidos	**A la Fecha**	**1**	**2**	**3**	**4**	**5**	**Total**
DL Diseño							
DL Diseño revisión							
Codificación							
Codificación revisión							
Compilación							
Pruebas							
Total							

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Figura 18. Formato de planeación de tareas

Nombre _____ Fecha _____ Programa # _____

Tarea		**Valor Planeado**			**Valor Ganado**	
No.	Nombre	Horas	Horas Acumuladas	Fecha	Fecha	Horas Acumuladas

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

Figura 19. Formato de calendarización

Nombre _____ Fecha _____ Programa # _____

Día/semana		**Plan**		**Actual**			
No.	Fecha	Horas	Horas Acumuladas	Horas	Horas Acumuladas	EV. Acumuladas	EV Ajustado

Fuente: Watts S. Humphrey
Introducción del proceso de *software* personal

2.5. Pasos involucrados en cada nivel

Los pasos que se necesitan para llevar a cabo un trabajo utilizando PSP son:

- Medición Personal (PSP0)
- Planificación Personal (PSP1)
- Calidad Personal (PSP2)
- Proceso Personal Cíclico (PSP3)

2.5.1. Medición personal: administración y seguimiento del tiempo

Se debe aprender como llenar los formatos del PSP y anotar los datos del trabajo personal.

2.5.1.1. Administración del tiempo

La mejor forma de gestionar el tiempo es comprender cómo se utiliza, lo que exige que se sepan catalogar las actividades, anotar el tiempo empleado en cada actividad y almacenar esos datos en un lugar apropiado.

2.5.1.2. Cuaderno del ingeniero

Para poder llevar acabo la tarea propuesta se necesita una agenda o libro para seguir el rastro al tiempo. También Se utilizará para otras cosas como almacenar asignaciones, anotar los compromisos, apuntar notas y como un libro de trabajo para diseñar ideas y cálculos

2.5.1.3. Seguimiento del tiempo

Se necesita almacenar en un formulario especial la fecha y la duración de las actividades que se realizan en cada jornada y de las interrupciones que se presentan. El manejo de las interrupciones es muy importante, mediante su identificación es posible eliminarlas para realizar un trabajo más eficaz y de más calidad.

2.5.2. Planificación del personal

Se debe realizar una planificación personal del trabajo. Este paso proporciona una estimación del tamaño y de los recursos utilizados.

2.5.2.1. Planificación del tiempo

Conviene hacer un resumen de actividad semanal. Es un nuevo reporte que se hace con las actividades de una semana.

2.5.2.2. Planificación del proyecto

Es otro reporte para almacenar los datos referentes al tiempo estimado en realizar un proyecto. Es un registro de planificación que contiene datos referentes al producto.

2.5.2.3. Relaciones entre el tamaño del programa y el tiempo de real

Como las tareas varían en tamaño y complejidad es útil tener una manera de comprobar sus tamaños. Basándose en la experiencia se puede estimar el tiempo de realización, empleando otro formulario para ello.

Se puede estimar el tiempo de realización de un programa, examinando los requisitos del programa que va a desarrollarse (numero de ciclos, listas de datos, cálculos, etc.) y después hacer una clasificación por tamaño del nuevo programa.

2.5.2.4. Administración de compromisos

Debe realizarse una lista de compromisos donde se anotará la fecha de cada compromiso y la cantidad de tiempo que probablemente se utilizará. Luego se establecerá una calendarización empleando Mapas de *Gant*.

2.5.2.5. Bases para realizar un seguimiento del trabajo

Proporciona una definición de cada tarea, una estimación del tiempo, de los recursos requeridos y un marco de gestión de revisión y control. Se hace un resumen de planificación del proyecto.

2.5.3. Calidad personal

Para realizar un buen trabajo se deben tener en cuenta los defectos que los programadores introducen en el código del programa. Estos defectos, en la mayoría de los casos, se deben a errores de tipos, por descuido o de nombres, pero en cualquier caso se deben prevenir y corregir.

2.5.3.1. Clasificación de los defectos

Los defectos se clasifican para ver que categorías causan la mayoría de los problemas y poder así prevenirlos y almacenarlos mejor. Se tienen errores de documentación, sintaxis, construcción, asignaciones, interfaz, chequeo, datos, función, sistema y entornos.

2.5.3.1.1. Registro de defectos

Los datos referentes a los defectos deben almacenarse, deben almacenarse en un apartado especial del Resumen de Planificación del Proyecto.

2.5.3.1.2. Métodos para hallar y arreglar defectos

Se puede utilizar el propio compilador. Se deben realizar pruebas con los datos apropiados.

Y desde luego repasando el código fuente. Se puede eliminar los defectos empleando pequeños prototipos de funciones poco familiares o procedimientos antes que se usen en un programa.

2.5.3.1.3. Estimación de los defectos

Se deben estimar los defectos tomando en consideración el número de líneas de código del programa

2.5.3.2. Costo de la calidad

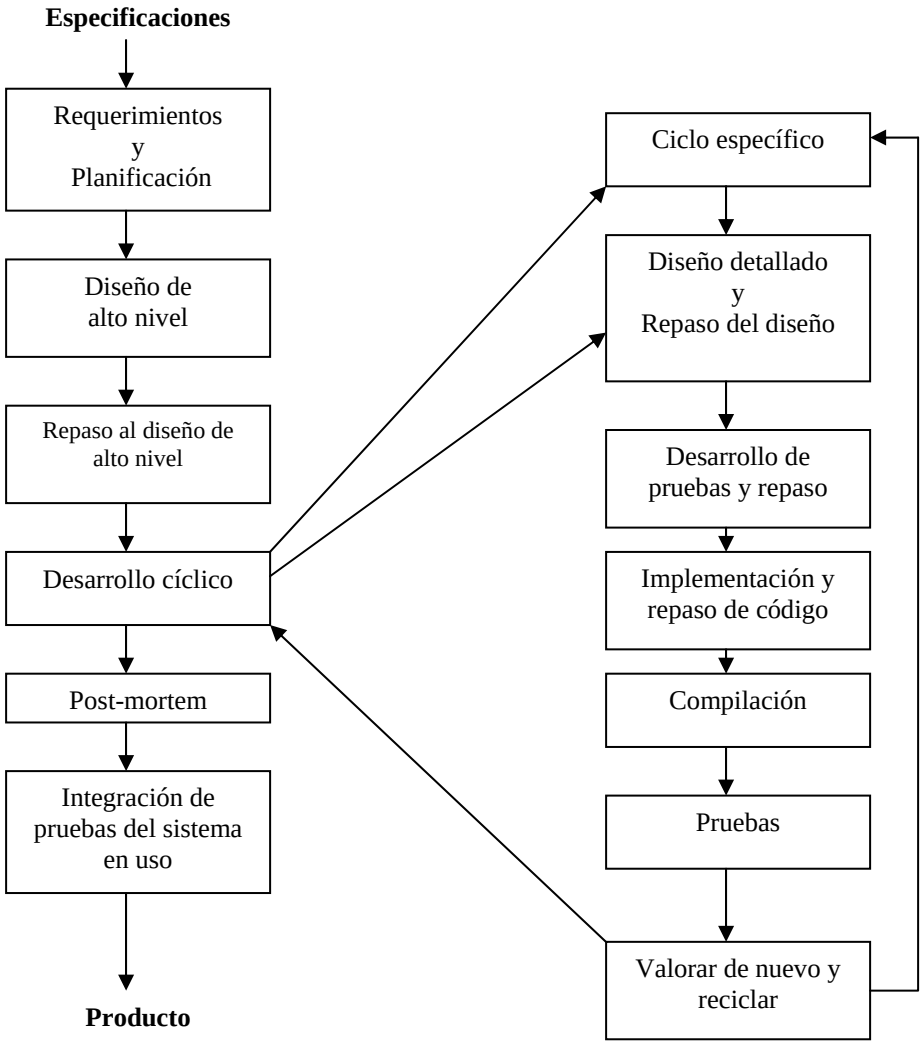
Para obtener *software* libre de defectos, se necesitará tomar en cuenta el tiempo empleado en encontrarlos, por lo que para evaluar el Costo de la Calidad, se deben considerar los costos del fracaso, de apreciación y de prevención.

2.5.4. Proceso cíclico

PSP3 es el último paso del PSP. Hasta ahora se ha visto un proceso lineal para construir programas pequeños.

Este último paso incluye métodos para uso individual que se utilizan para desarrollar programas a gran escala ver figura 20.

Figura 20. Proceso cíclico de desarrollo



Fuente: Dr. Leopoldo Altamirano Robles.
Metodologías de diseño

3. EJEMPLO DE LA UTILIZACIÓN DEL PROCESO *SOFTWARE* PERSONAL

Las siguientes instrucciones son todos los pasos que se deben seguir para la aplicación de PSP.

3.1. Tabla de registro del tiempo

El objetivo de registrar el tiempo es obtener información de cómo se trabaja realmente.

La cantidad típica de tiempo no interrumpido que los ingenieros dedican a sus tareas es generalmente inferior a una hora. Medir el trabajo en unidades de horas no proporcionará el detalle necesario para posteriormente planificar y gestionar el trabajo. Es mucho más fácil controlar el tiempo en minutos que en fracciones de una hora.

En la tabla de registros de tiempo se introduce en cada línea un período de tiempo:

- Fecha: la fecha de realización de alguna actividad (asistir a reuniones, analizar o escribir un programa).
- Inicio
- Fin
- Interrupción: algún tiempo perdido
- Tiempo delta*:* tiempo dedicado a alguna actividad, entre los tiempos de comienzo y fin, menos el tiempo de interrupción.
- Actividad*:* nombre la actividad a realizar
- Comentarios: descripción más completa de lo que se está haciendo.

- C (completado): se marca esta columna cuando se termina una tarea correspondiente a alguna actividad.
- U (unidades): el número de unidades de una tarea acabada.

Programa: Análisis y diseño de plan de prestaciones **Fecha:** 22/03/04
Nombre miembro grupo: Programador X
Código grupo: X

Fecha	Inicio	Fin	Tiempo Interrupción	Tiempo Delta	Actividad	Comentarios	C	U
22/03	9:00	12:00	10 + 15 + 5	150	Entrevista	Entrevistar al personal que usara el programa		
23/03	13:00	17:00	10 + 15 + 5 + 3 + 6 + 2	199	Entrevista	Sacar de las entrevistas los requerimientos		
24/03	9:00	17:00	30 + 45 + 10	395	Realización del documentos de requerimientos	Redacción de los documentos de requerimientos	X	3
25/03	7:00	10:00	20	160	Gestión Proyecto	Estimación de tiempo y personal(Teléfono)		
	10:00	12:30	15	135	Diagramas de uso	Elaboración de los Diagrama de uso del plan de prestaciones (Consulta de un tramite)		
	14:00	16:45	30	135	Diagrama de actividades	Reparto de casos de uso entre el Equipo para dibujar los diagramas de actividades (llamada de cliente)		
26/03	8:00	10:30		150	Revisión de Diagrama de Actividades	Explicación de los diagramas de actividades en la reunión		
	11:30	13:00	3 + 15	72	Diagramas de casos de uso	Reunión del equipo para versión final del diagrama general (descanso, juego)	X	1
	14:00	17:00	25	155	Gestión de proyecto	Reunión del equipo para reparto de tareas	X	1

Programa: Análisis y diseño de plan de prestaciones **Fecha:** 29/03/04
Nombre miembro grupo: Programador X **Código grupo:** X

Fecha	Inicio	Fin	Tiempo Interrupción	Tiempo Delta	Actividad ó Fase	Comentarios	C	U
29/03	7:30	11:30	5 + 8 + 10	217	Diagrama de clases	Estudio del proyecto y lista de posibles clases (descanso, charla)		
30/03	7:45	12:30		285	Gestión proyecto	Calculo del costo del proyecto	X	1
	14:00	16:00		120	Gestión Proyecto	Definir que infraestructura tendrá	X	1
31/03	7:30	17:00	5 + 10 + 150 + 5	400	Gestión proyecto	Recopilar y dar formato a documentos definitivos; (Descanso, Almuerzo,..)		
1/04	8:00	15:00	30	390	Gestión proyecto	Redacción de los documentos (Teléfono)		
2/04	7:30	11:30		240	Gestión Proyecto	Reunión con el gerente para verificación de documentación		
	14:00	17:00		180	Gestión Proyecto	Corrección de documentación	X	3

Programa: Análisis y diseño de plan de prestaciones **Fecha:** 05/04/04
Nombre miembro grupo: Programador X **Código grupo:** X

Fecha	Inicio	Fin	Tiempo Interrupción	Tiempo Delta	Actividad ó Fase	Comentarios	C	U
05/04	7:30	11:00		210	Distribución de tareas	Reunión con el equipo para asignación de tareas	X	1
06/04	7:45	12:35		290	Gestión Proyecto	Realización de la Base de datos		
	14:00	16:00		120	Gestión Proyecto	Reunión con equipo para revisión de la base de datos		
	16:10	17:30		80	Gestión Proyecto	Corrección de la base de datos	X	1
07/04	7:30	12:30	5 + 10	285	Gestión Proyecto	Generar el *scripts* e instalación	X	1

3.2. Administración de interrupciones

Un problema común con el control del tiempo son las interrupciones. La forma de administrar las interrupciones en el cuaderno de registro de tiempos consiste en anotarlas en la columna tiempo de interrupción. Puesto que el tiempo de las interrupciones no es tiempo de trabajo productivo, se deben controlar las interrupciones. Si la cantidad de este tiempo fuese constante no habría que hacer mucho para gestionarlo. Sin embargo, el tiempo de las interrupciones es muy variable. Si no se mide, habría que añadir un número aleatorio en todos los datos de tiempos, lo que haría más difícil utilizar estos datos para planificar o gestionar el tiempo. Esos datos registrados pueden utilizarse para comprender con qué frecuencia se interrumpe el trabajo. Las interrupciones no son solamente un despilfarro de tiempo, sino que rompen el ritmo de pensamiento, llevando a la ineficiencia y al error. Comprender cómo se es interrumpido ayuda a mejorar la calidad y eficiencia del trabajo.

3.3. Control de tareas

Para controlar cómo se gasta el tiempo se necesita controlar los resultados producidos. Cuando se desarrollan proyectos, se documenta las tareas realizadas y se necesita saber el tiempo que llevo al ser realizadas, con esto se podrá calcular la productividad de cada tarea. Con esta información se puede mejorar la planificación de futuros proyectos.

- Las columnas C y U del cuaderno de registro de tiempos significan completado (C) y unidades (U). Estas columnas ayudan a identificar rápidamente el tiempo dedicado a las distintas tareas.

- Por unidad se entiende unidad de trabajo completado.

- Para colocar en la columna C hay que comprobar cuando se ha terminado una tarea. También podría determinar cuántas tareas se ha realizado y cuánto tiempo.

Para tener unos registros de tiempo exactos, es importante completar las columnas C y U cada vez que se finalice una tarea que tenga resultados que se puedan medir.

3.4. Ideas para registrar tiempo

El control del tiempo es sencillo y pueden ayudar a hacerlo de forma más consistente y precisa:

- Lleva siempre el cuaderno de notas.

- Cuando se olvide registrar la hora de comienzo, la hora de fin o la duración de la interrupción, se hace una estimación tan pronto se recuerde.

- Se puede utilizar un cronómetro para controlar las interrupciones.

- Resume el tiempo utilizando el resumen semanal de actividades.

3.5. Resumen de actividades

Para hacer una planificación correcta es importante conocer como se gasta el tiempo. Lo primero es registrar el tiempo utilizando el cuaderno de registro de tiempos.

Después de reunir la información del tiempo de unas dos semanas se mira cómo se emplea el tiempo. Puesto que los registros de tiempos son muy detallado para la planificación, se necesita resumir los datos de una forma más útil. En la siguiente tabla se muestra el formato del resumen semanal de algún desarrollador:

Nombre: Programador X **Fecha:** 29/04

Actividad	Fecha L 22/03	M	M	J	V	S	D	Totales
Entrevistas	150	199						**349**
Realización del documentos de requerimientos			395					**395**
Gestión proyecto				160	155			**315**
Diagramas de uso				135	72			**207**
Diagramas de actividades				135				**135**
Revisión de Diagrama de Actividades					150			**150**
Diagrama de clases								**0**
Distribución de tareas								**0**
Resumen semanas anteriores								
Entrevistas								**0**
Realización del documentos de requerimientos								**0**
Gestión proyecto								**0**
Diagramas de uso								**0**
Diagramas de actividades								**0**
Revisión de Diagrama de Actividades								**0**
Diagrama de clases								**0**
Distribución de tareas								**0**
Resumen incluyendo la última semana								
Entrevistas								**349**
Realización del documentos de requerimientos								**395**
Gestión proyecto								**315**
Diagramas de uso								**207**
Diagramas de actividades								**135**
Revisión de Diagrama de Actividades								**150**
Diagrama de clases								**0**

Nombre: Programador X **Fecha:** 05/04

Actividad	Fecha L 29/03	M	M	J	V	S	D	Totales
Entrevistas								
Realización del documentos de requerimientos								
Gestión proyecto		405	400	390	420			**1615**
Diagramas de uso								
Diagramas de actividades								
Revisión de Diagrama de Actividades								
Diagrama de clases	217							**217**
Distribución de tareas								**0**
Resumen semanas anteriores								
Entrevistas								**349**
Realización del documentos de requerimientos								**395**
Gestión proyecto								**315**
Diagramas de uso								**207**
Diagramas de actividades								**135**
Revisión de Diagrama de Actividades								**150**
Diagrama de clases								**0**
Distribución de tareas								**0**
Resumen incluyendo la última semana								
Entrevistas								**349**
Realización del documentos de requerimientos								**395**
Gestión proyecto								**1930**
Diagramas de uso								**207**
Diagramas de actividades								**135**
Revisión de Diagrama de Actividades								**150**
Diagrama de clases								**217**
Distribución de tareas								**0**

Nombre: Programador X **Fecha:** 12/04

Actividad	Fecha L 05/04	M	M	J	V	S	D	Totales
Entrevistas								
Realización del documentos de requerimientos								
Gestión proyecto		490	285					**775**
Diagramas de uso								
Diagramas de actividades								
Revisión de Diagrama de Actividades								
Diagrama de clases								
Distribución de tareas	210							**210**
Resumen semanas anteriores								
Entrevistas								**349**
Realización del documentos de requerimientos								**395**
Gestión proyecto								**1930**
Diagramas de uso								**207**
Diagramas de actividades								**135**
Revisión de Diagrama de Actividades								**150**
Diagrama de clases								**217**
Distribución de tareas								**0**
Resumen incluyendo la última semana								
Entrevistas								**349**
Realización del documentos de requerimientos								**395**
Gestión proyecto								**2705**
Diagramas de uso								**207**
Diagramas de actividades								**135**
Revisión de Diagrama de Actividades								**150**
Diagrama de clases								**217**
Distribución de tareas								**210**

3.6. Actividades generales

De forma general, se utilizarán las siguientes actividades para controlar el proceso:

- Entrevistas platicar con la gente que hará uso de la aplicación.

- Gestión del proyecto: planificación, diseño, reparto de tareas
- Actividades por artefactos: la elaboración de cada uno de las distintas actividades.
- Documentación: recopilación de diagramas y del trabajo,

3.7. Soporte de UML

El PSP consiste de un conjunto de métodos, formas, y guiones que muestran a los ingenieros de *software* como planear, medir y administrar su trabajo. El PSP está diseñado para usarse con cualquier lenguaje de programación o metodología de diseño y puede ser usado para la mayoría de los aspectos del trabajo de *software*, incluyendo requerimientos de escritura, corridas de prueba, definición de procesos y reparación de defectos. Por lo anterior UML tiene cabida para la realización de las acciones indicadas antes, ya que su empleo queda implícito en las mismas.

3.8. Herramientas de apoyo

PROBE. *PROxy Based Estimating*. Emplea objetos como la base para estimar el tamaño de los productos. Los ingenieros hacen referencia a datos históricos de los tamaños de objetos similares que han desarrollado previamente y usan regresión linear para determinar el tamaño estimado del producto final. También con PROBE se puede estimar el desarrollo de recursos empleados.

4. ESTUDIO DE PSP EN LAS EMPRESAS DE GUATEMALA

El siguiente estudio se realizo en las empresas Guatemaltecas, que desarrollan *software*.

4.1. Evaluación del PSP

La evaluación tiene como objetivo determinar la capacidad y el proceso de *software* personal. Una evaluación es la que realiza apreciación del proceso de *software* (SPA): realizada por profesionales del SEI, en conjunto con personal de la organización.

Las encuestas tienen respuestas, Si, No, No Aplica, No sé.

4.2. Investigación de campo

Para la investigación se hizo uso de una encuesta para recopilar la información la cual me permite conocer las diferentes empresas guatemaltecas.

4.2.1. Objetivos del cuestionario

- Conocer e identificar los puntos de PSP, que aplican en las empresas guatemaltecas.
- Identificar si las empresas guatemaltecas conocen el concepto de Proceso de *Software* Personal.
- Identificar si las empresas utilizan PSP para la realización de los proyectos.

4.2.2. Metodología de la encuesta

Para la elección de las preguntas del cuestionario, se hizo uso de una metodología cerrada para que las personas que respondan lo hagan de una forma más directa.

La encuesta es para personas que se encuentran en el área de la informática o desarrollo del *software*. Que trabajan ya sea en el sector público o privado de las empresas guatemaltecas.

4.2.3. Contenido de la encuesta

La encuesta tiene 41 preguntas. De los cuales se realizará una gráfica de la pregunta más importante. Y se paso la encuesta a nueve empresas guatemaltecas.

4.2.3.1. Encuesta

Datos generales

Nombre: _____
Fecha: _____
Puesto: _____

¿Cuál es su puesto actual?

- Líder de proyecto
- Desarrollador
- Mando medio

¿En cuales de estas actividades trabaja?

- Análisis y/o especificación de requerimientos
- Diseño
- Codificación
- Pruebas
- Aseguramiento de la calidad
- Mejora de procesos de *software*
- Otro

¿Ha recibido entrenamiento en PSP?

- Si
- No

INSTRUCCIONES: marque con una X tu respuesta. Responda Si, si la práctica esta establecida y se hace consistentemente. No, si la práctica se lleva a cabo inconsistentemente. NA, si no aplica a su proyecto. No se no lo conoce.

Planeación del proyecto *software*

¿Se documentan los estimados de tamaño, costo y tiempo para planear el seguimiento de los proyectos?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se documentan las actividades, entregables y compromisos hechos para el proyecto?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se sigue una política escrita para planear un proyecto?

Si ___ **No** ___ **Na** ___ **No se** ___

¿El líder del proyecto revisa las actividades para la planeación del proyecto, tanto periódicamente como cuando se dan circunstancias especiales?

Si ___ **No** ___ **Na** ___ **No se** ___

Seguimiento y supervisión del proyecto

¿Se comparan los resultados reales (tiempos, tamaño y costo) con los estimados del plan?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se toman acciones correctivas cuando los resultados difieren de los planes?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Sigue el proyecto una política escrita para dar seguimiento y controlar las actividades?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Alguien del proyecto tiene asignada la responsabilidad de dar seguimiento a las actividades (esfuerzo, tiempos y presupuesto) y a los entregables?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se usan métricas para determinar el estado de las actividades de seguimiento y revisión, como esfuerzo usado en las actividades de seguimiento y control?

Si ___ **No** ___ **Na** ___ **No se** ___

Enfoque en procesos organizacionales

¿Están las actividades para desarrollar y mantener los procesos coordinadas en la organización?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Los mandos medios apoyan las actividades de desarrollo y mejora de procesos?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se usan métricas para determinar el estado de las actividades de desarrollo y mantenimiento de procesos?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Revisan periódicamente los mandos medias las actividades de desarrollo y mantenimiento de procesos?

Si ___ **No** ___ **Na** ___ **No se** ___

Definición del proceso de la organización

¿Las personas quienes desarrollan y mantienen los procesos reciben capacitación para realizar actividades de PSP?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se usan mediciones para determinar el estado de las actividades hechas para definir y mantener los estándares de procesos personales?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Están las actividades y productos para desarrollar y mantener procesos sujetas a revisiones de aseguramiento de la calidad?

Si ___ **No** ___ **Na** ___ **No se** ___

Administración de integración de software

¿Se desarrolló el plan del proyecto de acuerdo a los procesos definidos para la organización?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se planea y maneja el proyecto de acuerdo al plan escrito?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se requiere capacitación para las personas encargadas de adecuar los procesos para definir el plan para nuevos proyectos?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se usan métricas para determinar la efectividad de las actividades del manejo integrado del software?

Si ___ **No** ___ **Na** ___ **No se** ___

Ingeniería de productos de *software*

¿Se producen los resultados de acuerdo al plan del proyecto?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Sigue el proyecto una política escrita para hacer las actividades?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se proveen recursos adecuados para hacer las tareas, tales como equipo, herramientas, capacitación, etc.?

Si ___ **No** ___ **Na** ___ **No se** ___

Revisiones periódicas

¿Se planean revisiones entre analista/desarrolladores para requerimientos, diseño, codificación, pruebas, etc.?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se da seguimiento a los defectos encontrados en las revisiones hasta que se remueven?

Si ___ No ___ Na ___ No se ___

¿Los analista/desarrolladores reciben la capacitación requerida para hacer su parte?

Si ___ No ___ Na ___ No se ___

Administración de la calidad del *software*

¿Están planeadas las actividades de PSP en el proyecto para el manejo de la calidad del *software*?

Si ___ No ___ Na ___ No se ___

¿Se comparan las medidas de calidad con los objetivos de calidad del producto para determinar si se cumplen?

Si ___ No ___ Na ___ No se ___

¿Sigue el proceso una política organizacional escrita para manejar la calidad del *software*?

Si ___ No ___ Na ___ No se ___

¿Se usan métricas para determinar el estado de las actividades para el manejo de la calidad del *software*?

Si ___ No ___ Na ___ No se ___

Prevención de defectos

¿Se planean las actividades de prevención de defectos, por ejemplo, el análisis de defectos encontrados y la toma de acciones para prevenir su ocurrencia futura?

Si ___ No ___ Na ___ No se ___

¿Se prioritizan y eliminan las causas comunes una vez que se identifican?

Si ___ No ___ Na ___ No se ___

¿Sigue el proyecto una política organizacional escrita para las actividades de prevención de defectos?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se usan métricas para determinar el estado de las actividades de prevención de defectos, por ejemplo, el tiempo y costo invertido en identificarlos?

Si ___ **No** ___ **Na** ___ **No se** ___

Administración de cambios de tecnología

¿Sigue la organización un plan para el manejo de cambios de tecnología: identificación, selección y evaluación de nuevas tecnologías incorporándolas posteriormente en la organización?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se evalúan nuevas tecnologías para determinar su efecto en la calidad y productividad?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Existen datos que apoyen la selección de nuevas tecnologías?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Se usan métricas para determinar el estado de las actividades de manejo de cambio de tecnología?

Si ___ **No** ___ **Na** ___ **No se** ___

Administración de cambios del proceso

¿La gente de la organización participa en actividades de mejora de procesos?

Si ___ **No** ___ **Na** ___ **No se** ___

¿Sigue la organización una política escrita para implantar mejora de procesos de software?

Si ___ **No** ___ **Na** ___ **No se** ___

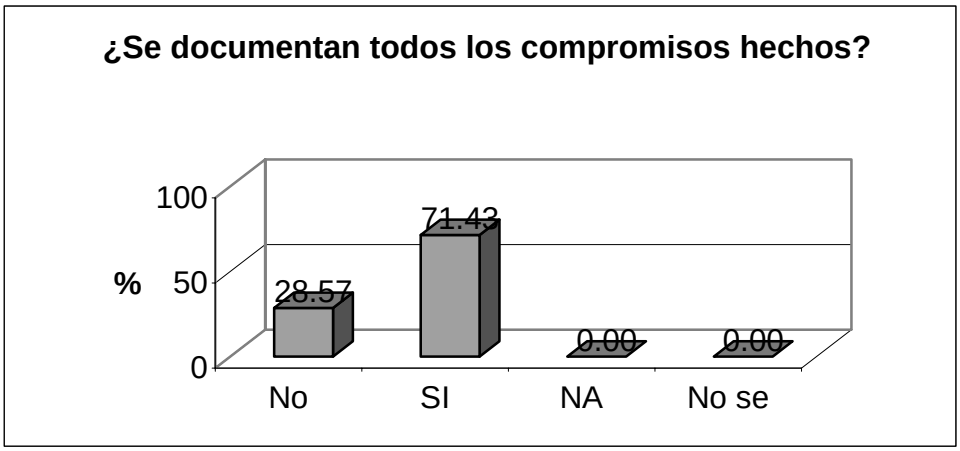
¿Se requiere capacitación en mejora de procesos para líderes y miembros técnicos?

Si ___ **No** ___ **Na** ___ **No se** ___

4.3. Análisis de resultados

Planeación del proyecto *software:* es establecer planes de desarrollo y manejo del proyecto, tales como: definición, restricciones, estimación de tamaño, esfuerzo, recursos y manejo de riesgos.

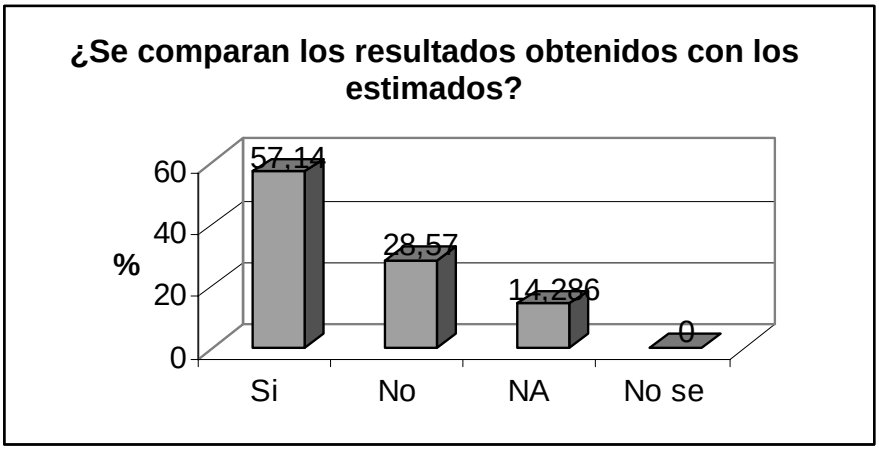
Figura 21. Planeación del proyecto *software*



El 71.42% de las empresas guatemaltecas si documentan sus compromisos hechos para el desarrollo del proyecto, lo que da una buena planificación del proyecto.

Seguimiento y supervisión del proyecto: la clave de esta sección es conocer el progreso del proyecto en cualquier momento. Haciendo comparaciones de lo alcanzado contra lo estimado y planeado, para poder hacer acciones correctivas fuera necesario.

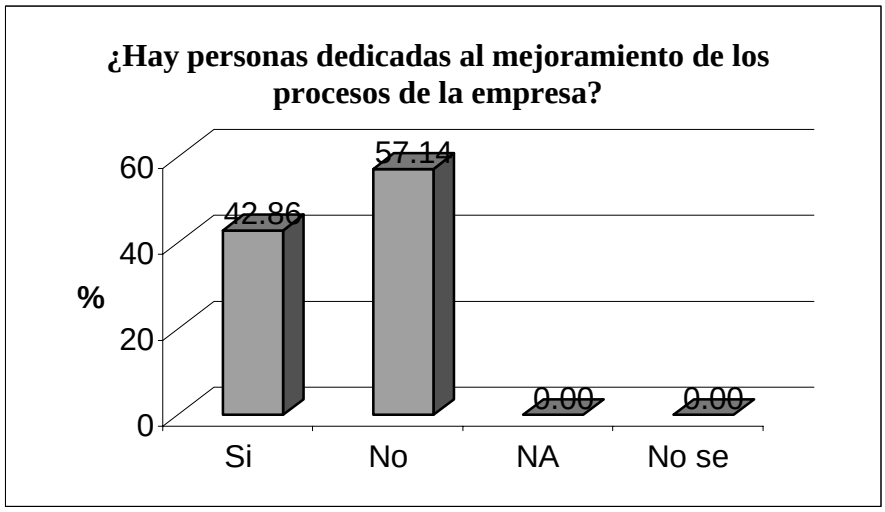
Figura 22. Seguimiento y supervisión del proyecto



El 57.14% de las empresas guatemaltecas si comparan los resultados con los planeados, y el otro 28.57% no compara sus resultados lo que a veces cometan los mismos errores en otros proyecto.

Enfoque del proceso de la organización: es responsabilidad de la organización de las actividades de desarrollo y mejoras de los procesos, usados por los proyectos a través de un grupo de ingeniería de *software*.

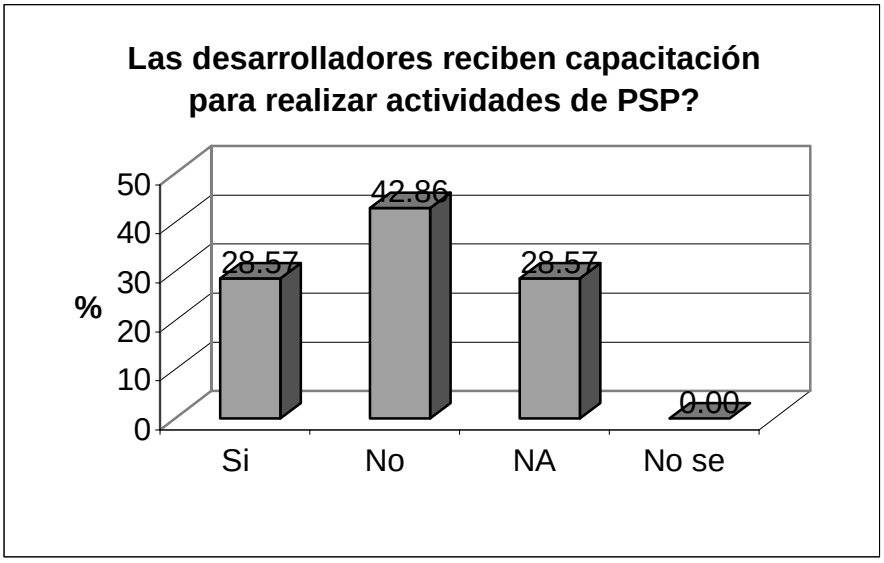
Figura 23. Enfoque del proceso de la organización



En el 57.14 % de las empresas guatemaltecas no existen personas que se dediquen a las mejoras de procesos, lo da que se siga desarrollando *software* de mala calidad.

Definición del proceso de la organización: el objetivo es desarrollar y mantener herramientas como la capacitación, para mejoran la aplicación de los procesos en las actividades de los proyectos.

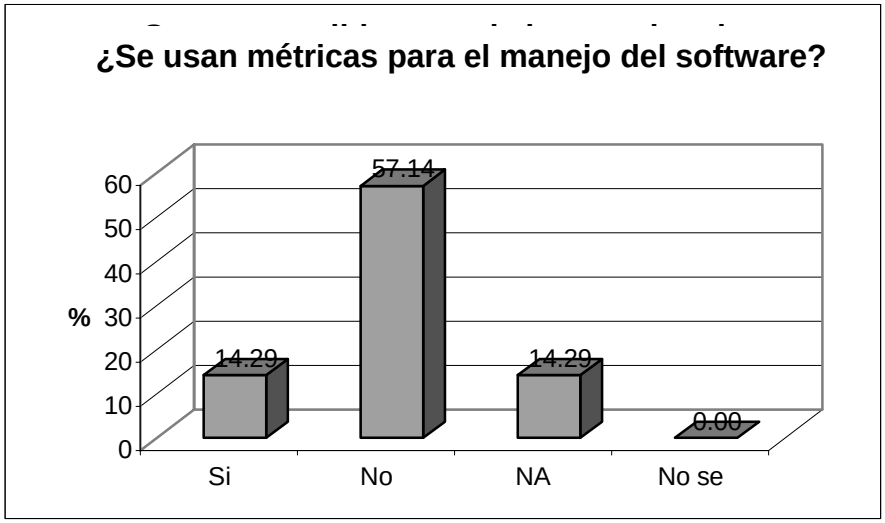
Figura 24. Definición de procesos organizacionales



En el 42.86% de las empresas guatemaltecas no dan capacitación al empleado en las actividades de la empresa, lo que origina que los desarrolladores organicen el tiempo, los recursos de una forma no adecuada.

Administración de integración de *software*: la clave es integrar las actividades de ingeniería de *software* y de administración de procesos de la organización. Tanto desarrollo de procesos y del proyecto usando dichos procesos.

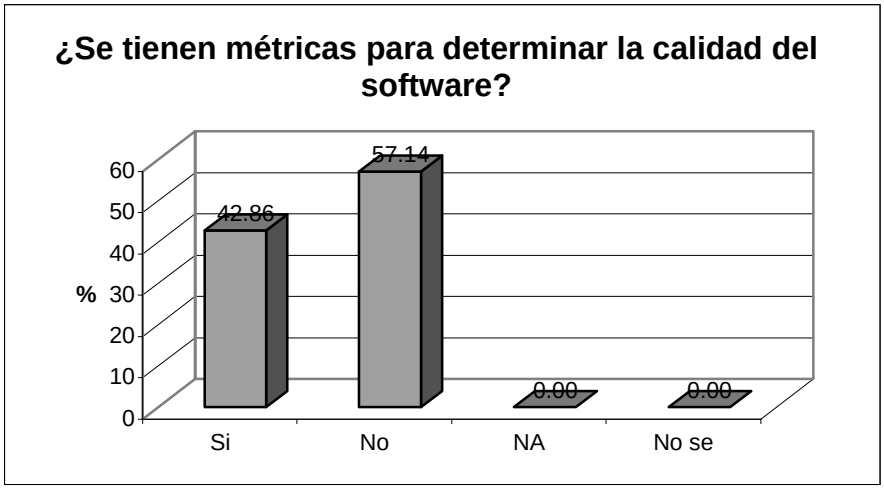
Figura 25. Administración de integración de *software*



El 57.14% de las empresas guatemaltecas no usan medidas para la integración del *software*.

Ingeniería de productos de *software*: El objetivo es que la ingeniería de *software* logre que los proyectos tengan resultados de acuerdo con lo planeado haciendo uso correcto de las herramientas y métodos apropiados al desarrollo.

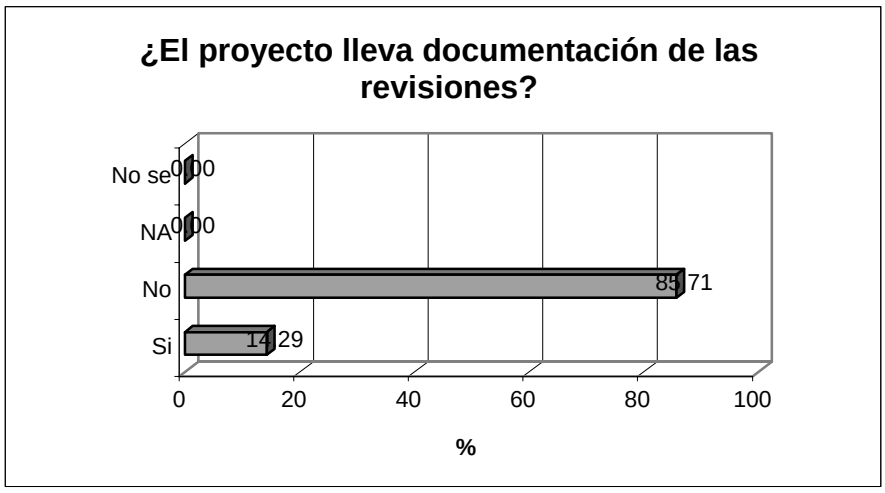
Figura 26. Ingeniería de productos de *software*



En la mayoría de las empresas no usan métricas para ver si los proyectos cumplen con lo planeado, porque no hacen uso correcto de las herramientas, lo que ocasiona que el proyecto tanga mala calidad y el cliente quede insatisfecho.

Revisiones periódicas: el objetivo es encontrar y arreglar los defectos lo más rápido posible, para poder prevenirlos hay que constar con una base de información que contenga un listado de los defectos para no caer en los mismos y el lugar donde se encuentran.

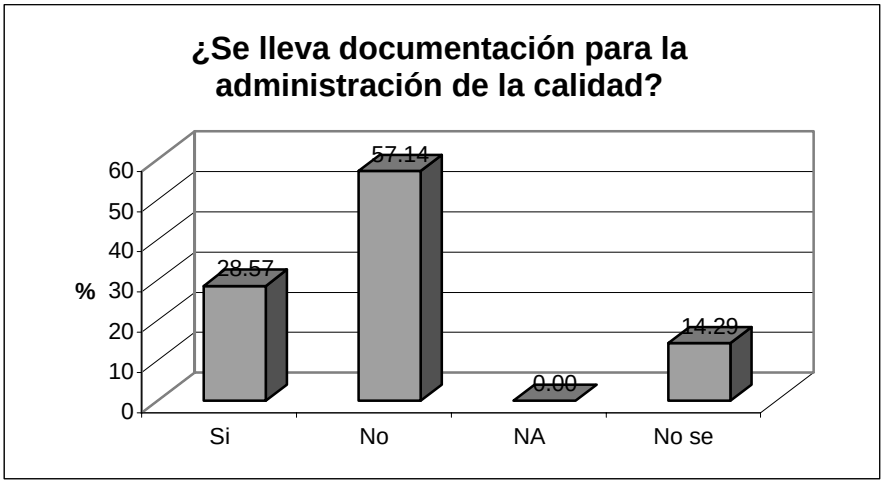
Figura 27. Revisiones periódicas



El 85.71 % de las empresas no documentan las revisiones de los proyectos, lo cual ocasiona que se puede caer en los mismos defectos en próximo proyecto, lo cual nos perderemos tiempo en arreglarlo.

Administración de la calidad del *software**:*** El objetivo es desarrollar productos de calidad, por medio de actividades que satisfacen las necesidades del cliente final y obteniendo producto de calidad.

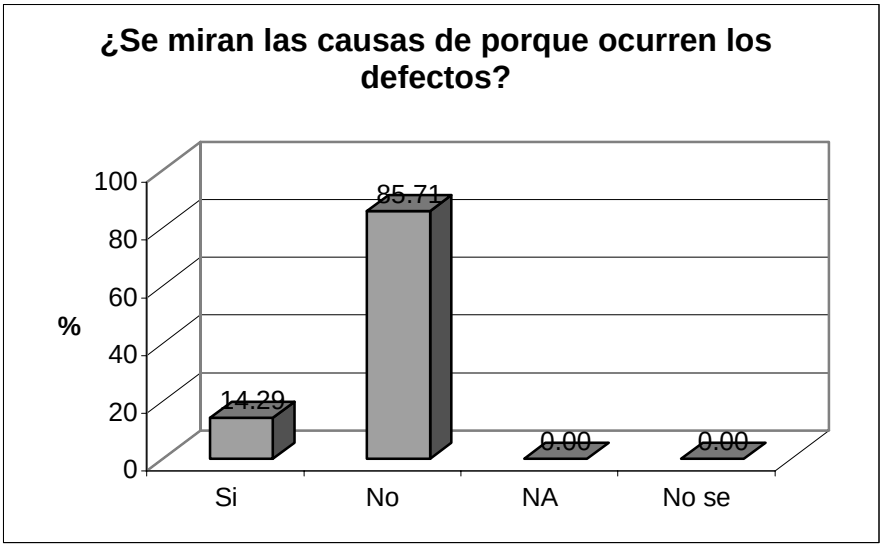
Figura 28. Administración de la calidad del *software*



Un 57.14% de las empresas no llevan documentación de la calidad del *software*, lo cual nos lleva a caer en los mismos errores.

Prevención de defectos: el objetivo es identificar los motivos de las apariciones de los defectos y que puede ocasionar en actividades que se hagan en el futuro. Se deben de tomar actividades tanto a nivel de la organización con del proyecto para prevenir los defectos.

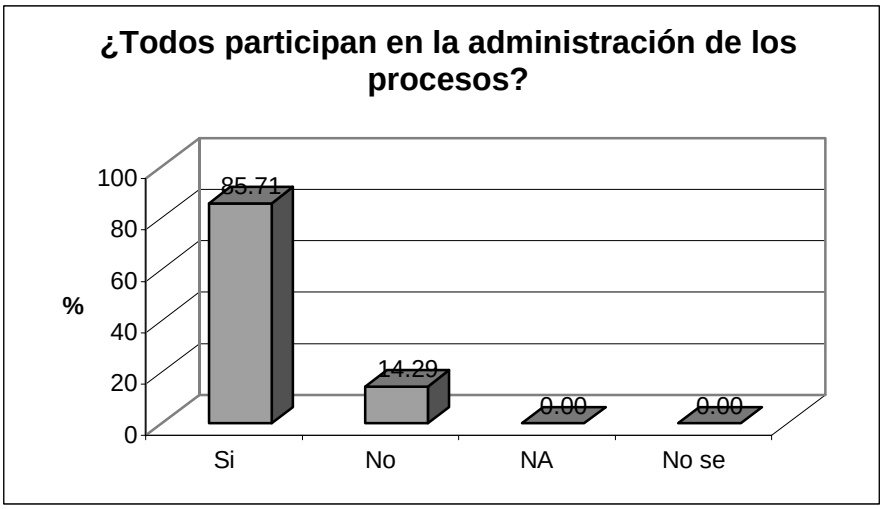
Figura 29. **Prevención de defectos**



Se puede observar que la mayoría de las empresas guatemaltecas no realizan reuniones para ver la causa de los defectos, lo que da como resultado que se caiga en los mismos errores o defectos.

Administración de cambios del proceso: esta sección consiste en que las personas de la organización participe en las actividades de mejorar la calidad del desarrollo, de procesos, por medio de programas de capacitación, y luego se hacen pruebas para poder implementar los cambios dejando una documentación escrita, para tener un historial.

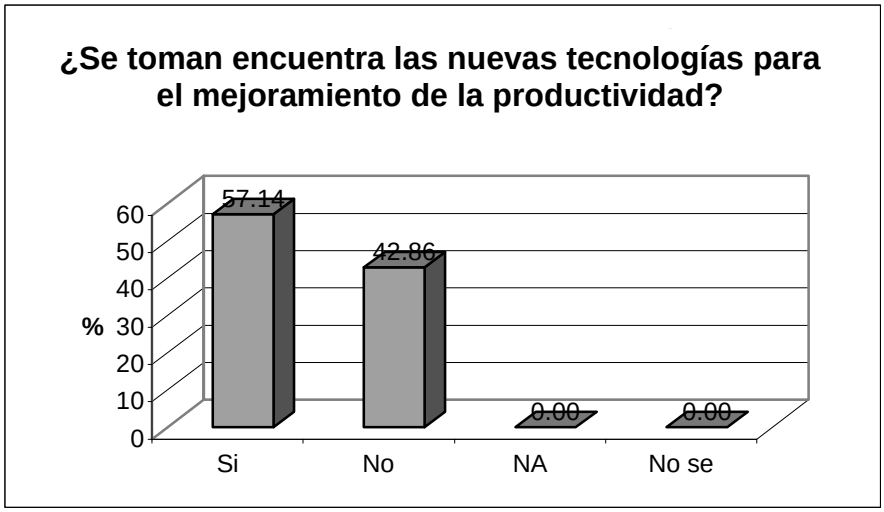
Figura 30. **Administración de cambios del proceso**



El 85.71% de los desarrolladores participan en actividades de mejorar los procesos, pero no lo documentan y no tienen control de la mejora.

Administración de cambios de tecnología: la clave de esta sección es tener un plan de manejo de los cambios tecnológicos, herramientas y procesos. Que con lleva a la evaluación de las mismas para ver su efecto en la calidad y productividad en el desarrollo del *software.*

Figura 31. **Administración de cambios de tecnología**



El 57.14 % de las empresas si evalúan las nuevas tecnologías para ver su efecto en la productividad del sistema y la calidad del mismo.

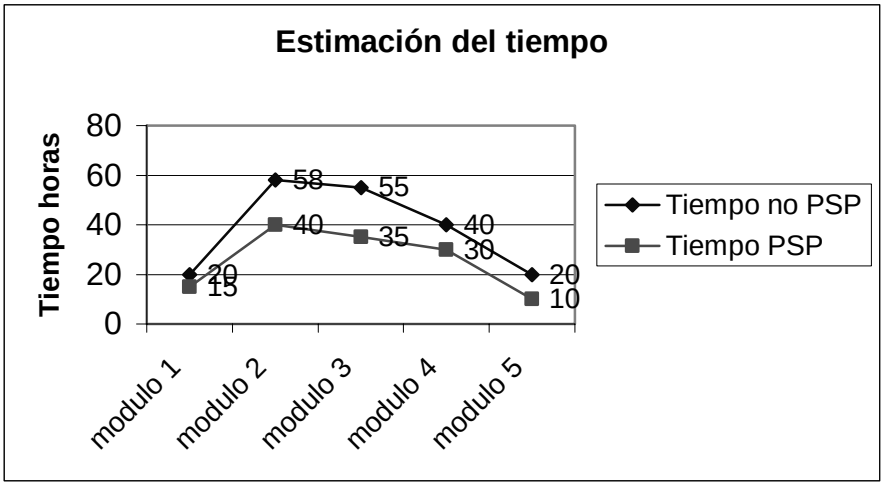
Los ingenieros que sí cuentan con experiencia en PSP mejoran la estimación de tiempo en los proyectos. Cuando se cumplen las necesidades del cliente, el analista/programador se da a conocer como:

- una persona confiable.
- esto es señal de garantía de un buen trabajo
- un buen servicio
- seguridad y cumplimiento de contrato.

A continuación se presentan un de ejemplo que se emplea PSP.

En la figura 32 se muestra la estimación del tiempo usando PSP y no usándolo para cada modulo.

Figura 32. Estimación del tiempo



En la figura 33 y 34 se muestra una mejora en compilación y pruebas para encontrar defectos de un 60% error a un 30% error. Mejorando la productividad del desarrollo ya que no se tienen los mismos errores, ya que se contaba con un historial de los defectos que podían ocurrir, para no caer en los mismo y ayudo a dedicarle mas tiempo a otras actividades.

Figura 33. Los resultados de calidad

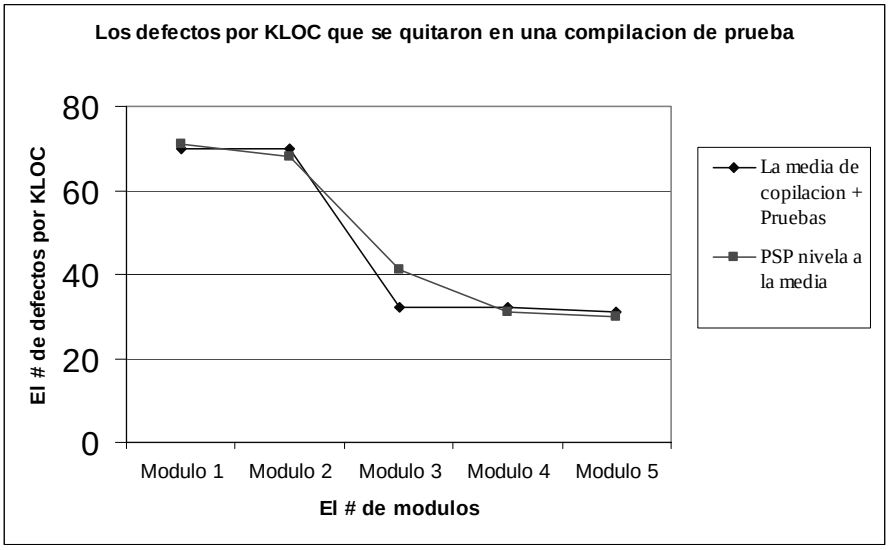
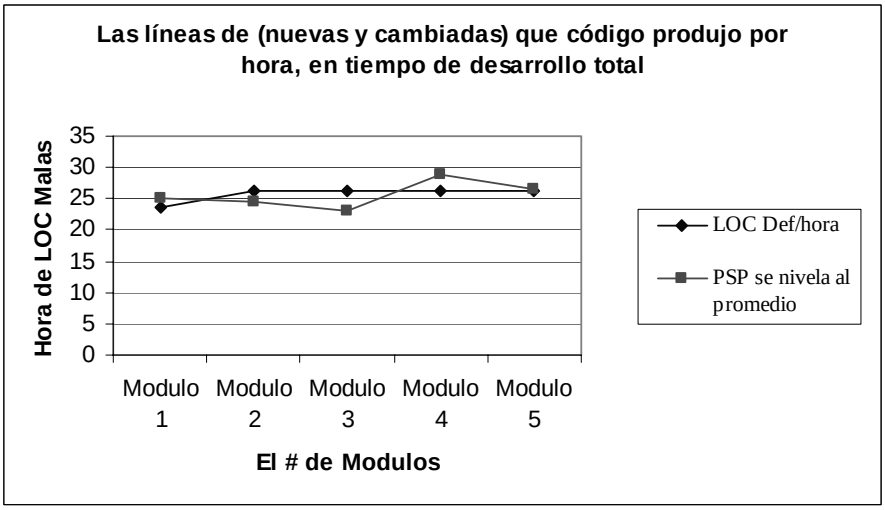
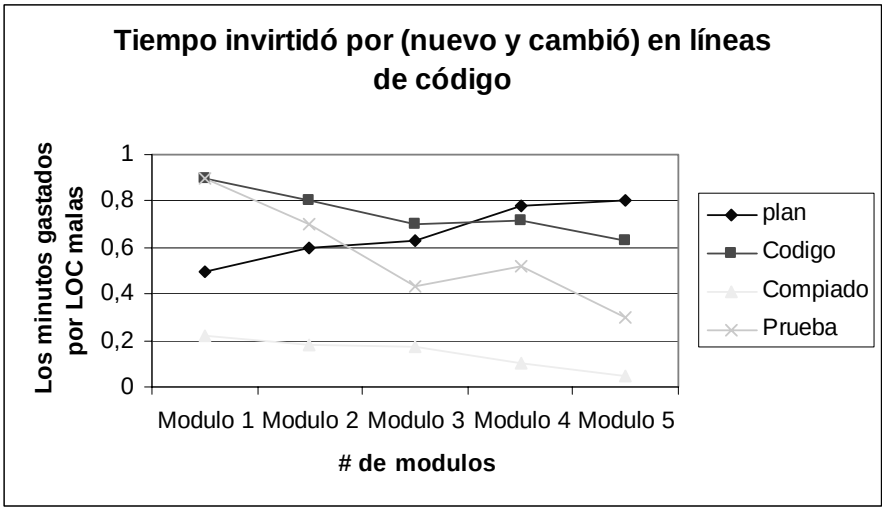


Figura 34. Los resultados de productividad



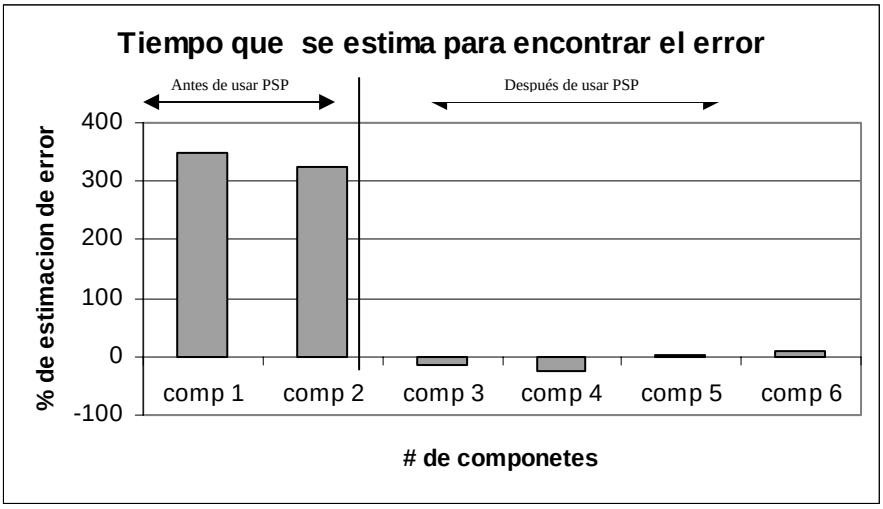
Como se muestra en figura 35, antes de usar PSP los analistas/desarrolladores gastaban más tiempo en el desarrollo, que en el diseño. Después de usar PSP los analistas/desarrolladores le dedican más tiempo al diseño que a cualquier otra fase del proyecto, mejorando su planificación, calidad y productividad del proyecto.

Figura 35. Los resultados de distribución de esfuerzo



PSP ha sido usado por el área de informática al aplicar este método es evidente los beneficios, en la figura 36 se muestra en la parte izquierda de la figura cuando la planificación del tiempo del primer componentes era de dos semanas y en realidad fueron 4 semanas su error entre los primeros componentes fue de 330%. Después al usar PSP en los siguientes componentes su porcentaje de error era -10.9. La estimación original para los 6 componentes, por ejemplo, era 6 semanas y el trabajo se completó en 4 semanas.

Figura 36. Tiempo en que se estima el error



4.3.1. Ventajas

- Se aumenta la productividad de los desarrolladores personalmente.
- Se mejora la forma de administrar el tiempo.
- Se logra realizar un buen plan de trabajo.
- Se consigue una manera más eficiente de programar.
- Se puede evaluar la productividad de cada desarrollador.
- Utilizar PSP existe una mayor relación con la creatividad, planeación, defectos y la productividad.

4.3.2. Desventajas

- Puede ser difícil de adoptar por la gran cantidad de detalles que hay que tomar en cuenta.
- La cantidad de reportes y formularios que hay que manejar es exageradamente grande.

- No es un método para realizar diseños completos, sino un método para mejorar el trabajo individual.
- PSP asume que ya existe una definición de requerimientos.
- La tendencia es ahorrar tiempo lo cual puede traducirse a la eliminación de pasos.
- El no considerar las revisiones dentro del proceso puede resultar más costoso.

CONCLUSIONES

1. PSP es un proceso de desarrollo de *software* que persiguen aumentar la productividad de los desarrolladores, empleando cualquier plataforma de análisis y desarrollo de sistemas, exigiendo que el desempeño personal del desarrollador sea cada vez de mayor calidad.

2. La clave para aprender la metodología del proceso personal de *software* (PSP) está en leer y analizar los datos del proyecto que se desarrollará y lo que en realidad esta información dicen acerca del desempeño personal de cada desarrollador.

3. PSP busca las actividades ejercidas por cada persona, con autonomía, responsabilidad, transparencia de acciones y orientadas para el crecimiento personal, a través del conocimiento y de las nuevas habilidades. El conocimiento humano, da resultados globales, es el mayor factor competitivo. La empresa que tiene el conocimiento domina el mercado.

4. En la evaluación se refleja que las empresas guatemaltecas no brindan suficiente capacitación a los desarrolladores, impidiendo que estos no se puedan desarrollar de buena manera. En las empresas guatemaltecas no existe una buena capacitación en el área de la administración del *software*, esto se debería de hacer desde cuando se esta aprendiendo a programar para que se acostumbre al los mejores métodos de administración de los proyectos en esta caso PSP. En la mayoría de empresas no existen un buen control de calidad del producto lo que ocasiona que muchas veces el cliente quede inconforme con el producto recibido.

RECOMENDACIONES

- Fomentar la importancia que se debe de tener sobre la calidad de *software* que se desarrollan, y aplicarlos a través de una metodología de evaluación del *software*.

- Hay que planificar lo que se desea hacer para llevar un mejor control, además, hay que comparar los resultados con lo planeado ya que de esta forma evitaremos los errores cometidos.

- Para la precisión que debe de existir cada vez que se realicen cálculos en cualquier etapa de PSP, es muy importante que la información que se toma en cuenta sea lo más fiel posible, en cuanto a lo que pasa en realidad para que el margen de error sea mínimo y los resultados sean precisos.

- Como proyecto futuro, faltaría ahondar en el contenido de todo el proceso de *software* en equipo (TSP) y CMM (capacidad de los procesos de desarrollo y su madurez), que forman parte del desarrollo de *software* para mejorar la calidad y productividad.

BIBLIOGRAFÍA

- *Software Engineering Institute*. "Publications". http://www.sei.cmu.edu/publications/publications.html www.sei.cmu.edu/publications/documents/97.reports/97tr001/97tr001abstract.html 10-02-04

- Ian Sommerville, "Ingeniería de *Software***",** 5ta. Edición, 1995. 250 Pág.

- Watts S. Humphrey **Introducción del proceso de *software* personal** México, 1997, 120 Pág.

- Villamil Gustavo, Guía de la Computación "ISO 9000 Aplicada al *Software*" http://www.cp.com.uy/42/iso42.htm, 20-04-04

- Proceso de desarrollo de *software* http://cannes.rhon.itam/Alfredo/Espaniol/Publicaciones/MINT, 20-04-04

- Proceso de *software* personal http://serdis.dis.ulpgc.es/~jsanchez/teaching/MDS 15-03-04

- Proceso de *software* personal http://www.sei.cmu.edu/publications/publications.html 10-03-04

- Why Don't They Practice What We Preach? http://www.sei.cmu.edu/publications/articles/practice-preach/practice-preach.html http://www.sei.cmu.edu/publications/articles/practice-preach/table.html 10-03-04

- Boehm, B. ***Software* Engineering Economics***.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

- Watts S. Humphrey **"Managing the *Software* Process".** Addison-Wesley, 1989.

- Watts S. Humphrey **A Discipline for *Software* Engineering.** Addison-Wesley, 1995.