

CAPITULO 2. PROCESO PERSONAL DE SOFTWARE.

Después de la segunda guerra mundial, la estrategia de calidad en la mayoría de las organizaciones industriales se basaba casi por completo en las pruebas. Las empresas establecieron departamentos especiales de la calidad para encontrar y arreglar problemas después de la producción de los productos. No fué sino hasta los años 70 y los años 80 que W. Edwards Deming y J.M. Juran convencieron a la industria estadounidense que se centrara en mejorar la forma en la que la gente hacía sus trabajos y desarrollaban sus procesos. [DEMING; 82], [JURAN 88]

En los siguientes años, este enfoque a los procesos de trabajo, ha sido responsable de las mejoras importantes en la calidad de automóviles, de la electrónica, o de casi cualquier otra clase de producto. La estrategia tradicional que había de "prueba-y-arregla" ahora es reconocida como costosa, que desperdicia tiempo y que además es ineficaz para el trabajo de la ingeniería y de la fabricación.

Aunque la mayoría de las organizaciones industriales ahora han adoptado principios modernos de calidad, la comunidad del software ha continuado confiando en la prueba como el método principal de la administración de la calidad. Para el software, la primera medida principal en la dirección iniciada por Deming y Juran fué tomada por Michael Fagan cuando en 1976 él introdujo las inspecciones del software [FAGAN; 86]

Usando inspecciones, las organizaciones han mejorado substancialmente la calidad del software. Otra medida significativa en la mejora de calidad del software fué tomada con la introducción del modelo de capacidad de madurez (CMM) en 1987.

El enfoque principal de CMM estaba en el sistema que administraba la ayuda que se le proporcionaba a los ingenieros de desarrollo. CMM ha tenido un efecto positivo en el funcionamiento de las organizaciones del software [HERBSLEB; 97]

Otra medida significativa en la mejora de calidad del software fué tomada con la esencia del proceso personal del software (PSP) ya que PSP amplía el proceso de mejora a la gente que realiza el trabajo de desarrollo de software.

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. PSP se diseñó para ayudar a profesionales del software para que utilicen constantemente prácticas sanas de ingeniería de software.

Asímismo les enseña a cómo planear y darle un seguimiento a su trabajo, a utilizar un proceso bien definido y medido, a establecer metas medibles, y finalmente a la utilización del rastreo constante para alcanzar dichas metas. PSP les demuestra a los ingenieros a cómo manejar la calidad desde el principio del trabajo, a cómo analizar los resultados de cada trabajo, y a cómo utilizar los resultados para mejorar el proceso del proyecto siguiente. [SEI; 2000]

2.1 ¿CÓMO FUÉ DESARROLLADO PSP?

Después de que Watts S. Humphrey condujera el desarrollo inicial de CMM para software, se decidió a aplicar los principios de CMM a los programas pequeños.

Después de eso mucha gente preguntaba cómo aplicar CMM a las organizaciones pequeñas o al trabajo de los equipos pequeños de software.

Mientras que los principios de CMM se aplicaron a tales grupos, cada vez se volvía mas necesaria la asesoría para saber que hacer. Fué entonces cuando Humphrey decidió personalmente utilizar los principios de CMM para desarrollar programas modulares para ver si dicho enfoque podría funcionar para convencer a los ingenieros de software a que adoptaran tales prácticas.

Fué entonces en el desarrollo de estos programas modulares, cuando Humphrey utilizó personalmente todas las prácticas de CMM para que él subiera poco a poco hasta llegar al nivel 5. Poco después él comenzó a trabajar en el proyecto tiempo completo en abril de 1989, el Instituto de la Ingeniería de Software (SEI) hizo a Humphrey un colaborador del SEI, permitiéndole trabajar tiempo completo en la investigación detallada de PSP.

Durante los siguientes tres años, él desarrolló un total de 62 programas y definió cerca de 15 versiones de PSP. Utilizó los siguientes lenguajes de programación: PASCAL y C++ , para desarrollar cerca de 25.000 líneas de código que ayudarían a darle la forma final a PSP. [SEI; 2002]

De esta experiencia, él concluyó que los principios de la administración de procesos que desarrolló Deming y de Juran eran totalmente aplicables tanto al trabajo de los ingenieros de software de manera individual como a ingenieros enfocados al trabajo en equipo, el resultado? Proceso en equipo de software (TSP)

Humphrey después escribió un libro que les proporcionó a varios asociados el material necesario para enseñar cursos de PSP. En septiembre de 1993, Howie Dow enseñó el primer curso de PSP a cuatro estudiantes graduados en la Universidad de Massachusetts.

Humphrey también enseñó el curso de PSP durante el semestre del invierno de 1993-1994 en la universidad de Carnegie Mellon, al igual que Nazim Madhavji en la Universidad McGill y Soheil Khajanoori lo enseñó en la Universidad Aeronáutica de Embry. De acuerdo con las experiencias y los datos que proporcionaron estos cursos, Humphrey realizó la revisión del libro de PSP y publicó la versión final a finales de 1994. [HUMPHREY; 95]

Casi al mismo tiempo, Jim Over y Neil Reizer del SEI y Robert Powels de la compañía de Servicios Informativos Avanzados (AIS por sus siglas en inglés) desarrollaron el primer curso para entrenar a los instructores a enseñar el curso de PSP en la industria. Humphrey junto con el SEI han continuado trabajando en el desarrollo de PSP y asimismo han aplicado los mismos principios al Proceso en Equipo de Software o TSP.

2.2 PRINCIPIOS DE PSP

El diseño de PSP se basa en los siguientes principios de planeación y de calidad [HUMPHREY; 95]

- Cada ingeniero es esencialmente diferente; para ser más precisos, los ingenieros deben planear su trabajo y basar sus planes en sus propios datos personales.
- Para mejorar constantemente su funcionamiento, los ingenieros deben utilizar personalmente procesos bien definidos y medidos.
- Para desarrollar productos de calidad, los ingenieros deben sentirse personalmente comprometidos con la calidad de sus productos.

- Cuesta menos encontrar y arreglar errores en la etapa inicial del proyecto que encontrarlos en las etapas subsecuentes.
- Es más eficiente prevenir defectos que encontrarlos y arreglarlos.
- La manera correcta de hacer las cosas es siempre la manera más rápida y más barata de hacer un trabajo.

Para hacer un trabajo de ingeniería de software de la manera correcta, los ingenieros deben planear de la mejor manera su trabajo antes de comenzar y deben utilizar un proceso bien definido para realizar de la mejor manera la planeación del trabajo.

Para que los desarrolladores lleguen a entender su funcionamiento de manera personal, deben medir el tiempo que pasan en cada proceso, los defectos que inyectan y remueven de cada proyecto y finalmente medir los diferentes tamaños de los productos que llegan a producir.

Para producir constantemente productos de calidad, los ingenieros deben planear, medir y rastrear constantemente la calidad del producto y deben centrarse en la calidad desde el principio de un trabajo.

Finalmente, deben analizar los resultados de cada trabajo y utilizar estos resultados para mejorar sus procesos personales.

2.3 NIVELES DE PSP

PSP tiene un marco de proceso de evolución similar al que tiene CMM. PSP trata parcialmente 12 de las 18 KPA's definidas en el CMM.

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

En CMM un nivel de madurez sólo se alcanza si se logran cumplir todas las KPA's que exige cada nivel. Sin embargo PSP sólo cubre de manera parcial estas KPA's debido a que es un complemento de CMM y no depende uno del otro en ningún sentido por lo que es considerado como material de apoyo.

Como se ha visto anteriormente el Instituto de la Ingeniería del Software (SEI) ha desarrollado el proceso personal del software para definir y reparar la holgura que existe entre el modelo de la madurez de la capacidad y el individuo. Por lo tanto es ideal utilizarlo junto con CMM pero no es obligatorio ya que es un proceso y no un modelo como lo es CMM.

Para desarrollar software de alta calidad, cada componente individual también debe de contar con la más alta calidad posible. La estrategia total de PSP es cerciorarse de que todos los componentes individuales se desarrollen con la más alta calidad. PSP logra esto proporcionando un marco de proceso personal ya definido que el programador puede utilizar. Este marco es:

- Desarrollar un plan para cada proyecto y/o componente.
- Registrar su tiempo de desarrollo.
- Registrar sus defectos
- Conservar sus datos en informes del proyecto
- Utilizar sus datos para planear los proyectos y/o los componentes futuros.

- Analizar sus datos para desarrollar sus procesos con mas calidad para mejorar su funcionamiento.

El proceso personal de software fue diseñado para ayudar y guiar a los ingenieros de software a realizar bien su trabajo y haciendo esto pueden llegar a cubrir las KPA's requeridas. PSP también muestra como aplicar métodos avanzados de ingeniería a sus proyectos y/o deberes diarios. Asimismo provee métodos de estimación y de planeación muy bien detallados que son necesarios para dar un seguimiento a su trabajo.

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.

La figura 2 muestra un diagrama que contiene todos los niveles de PSP. Asimismo se muestra que cada nivel cuenta con sus propios requerimientos o KPA's pertenecientes únicamente a PSP pero que podrían compartir intereses con las KPA's de CMM.

Es importante para las personas o empresas que quieran implementar PSP saber que deben de cumplir con todas las KPA's para que avancen de la mejor manera posible al siguiente nivel.

Cabe recalcar que se puede "personalizar" el proceso agregando o removiendo tareas conforme a las exigencias de cada persona o empresa. Esto quiere decir que por lo mismo de que PSP es un proceso y no un modelo, se puede amoldar a las necesidades del programador.

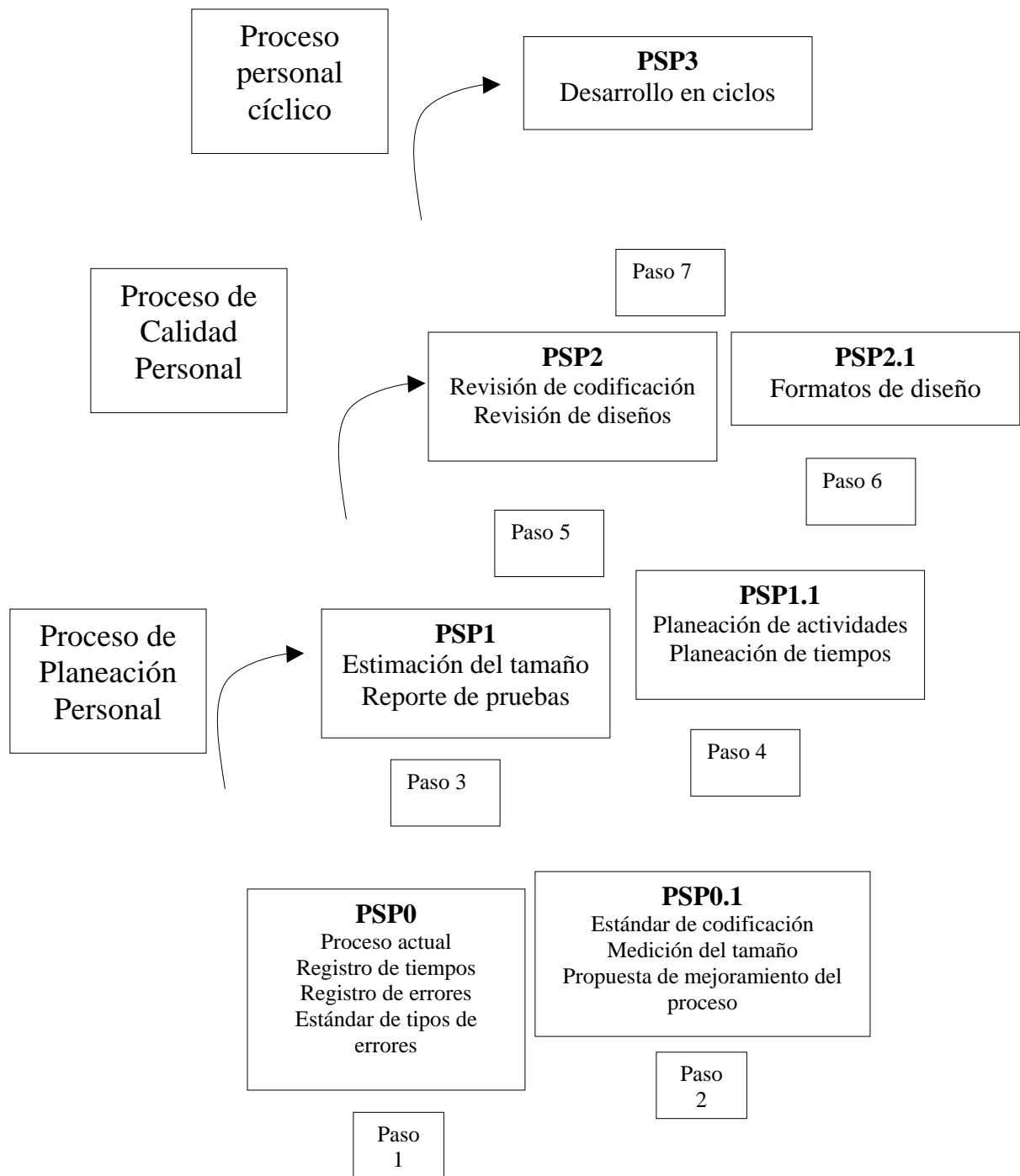


Figura. 2.1 "Evolución del proceso personal de software". [HUMPHREY; 97]

Ahora bien que si lo que se quiere hacer es que el proceso de PSP sea usado para cumplir con el modelo de capacidad de madurez, entonces se deben de tomar en cuenta los puntos que exige CMM de PSP.

En cada nivel de CMM existen KPA's que pueden ser cumplidas siguiendo el proceso personal de software, de hecho esa sería la mejor manera de cumplir con las exigencias de CMM.

No por nada fué que Watts S. Humphrey quiso crear PSP y otros procesos, sino para que fueran el complemento ideal y el "atajo" que se podría tomar para cumplir cuanto antes con los niveles de CMM que se desean alcanzar.

Existen resultados que se muestran en capítulos posteriores que demuestran la eficiencia y una reducción de tiempo considerable cuando se usan estos procesos diseñados para alcanzar los niveles de CMM más rápidamente que si se utilizaran otros "camino".

Hasta ahora se ha visto que PSP tiene sus niveles internos y que también cuenta con niveles que se deben cumplir junto a su modelo por el cual fué creado, CMM. El programador y el proyecto determinan si se debe utilizar PSP en su totalidad o sólo parcialmente.

También se ha visto la gran dinamicidad que puede llegar a tener PSP con respecto a cualquier proyecto que esté en desarrollo. Desgraciadamente éste atributo no es bien aprovechado o en dado caso no se hace de manera correcta la decisión sobre que puntos tomar de PSP y cuales no, si conviene hacer uso de todos los puntos que sugiere el proceso o sólo algunos.

La figura 2.2 presenta los elementos que tienen en común PSP con CMM. De esta manera puede analizarse cuales se podrían aplicar a cada proyecto.

Elementos del PSP en el CMM

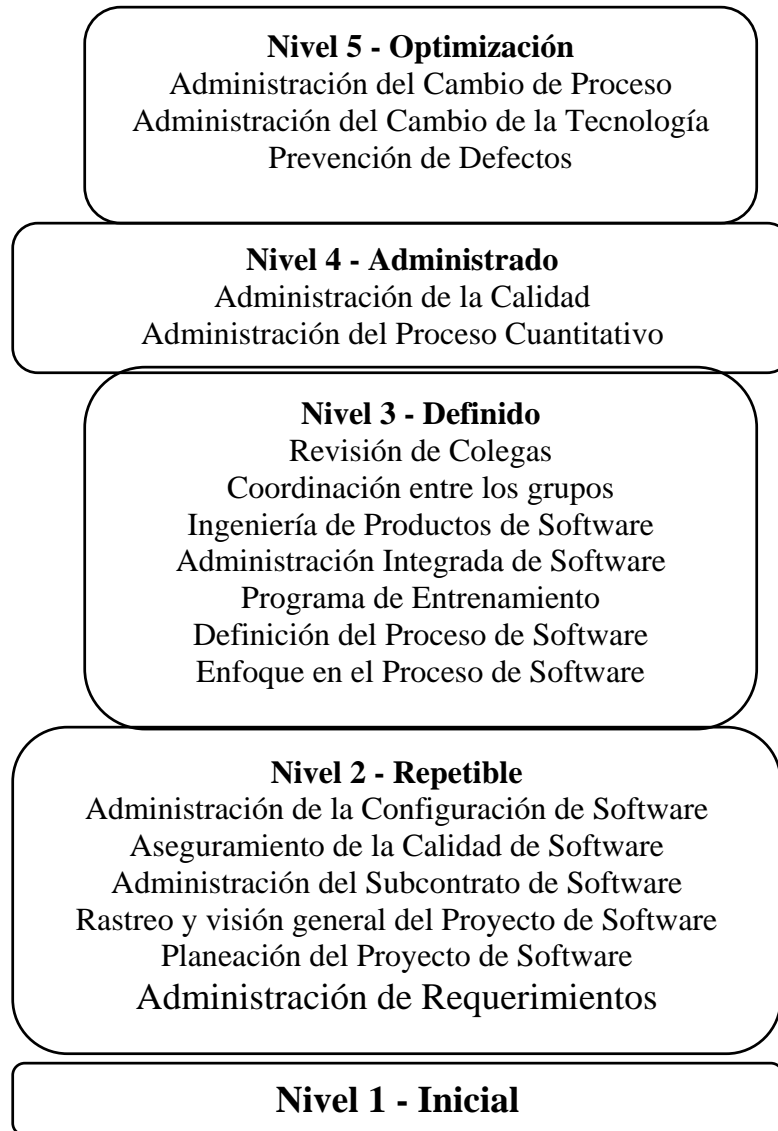


Figura 2.2. "Elementos de PSP en CMM" [CEDILLO; 2000]

Esta información es de gran utilidad porque así el programador tiene una idea clara de los puntos que debe cubrir respecto a sus necesidades.

Esto quiere decir si sus necesidades se encuentran dentro del rango de lo que exige CMM o si se aplican por separado.

En la Figura 2.3 se muestra la proveniencia de PSP y el "corazón" de todos los procesos existentes alrededor de CMM, algunos ya existentes y otros que aún no están completos. Seguramente vendrán procesos más completos en un futuro y que se acomodarán a las exigencias de la tecnología.

Probablemente la mayoría de estos procesos se conviertan en un estándar para las empresas que desarrollan software. Es conveniente saber de donde surgen estas nuevas tecnologías y hacia donde se encaminan. Por lo pronto el primer paso se ha dado, aunque PSP no sea un estándar aún, pero si se ha convertido en un proceso muy útil para desarrollar software de buena calidad.

Ahora que PSP está completo, seguramente los ingenieros de software se están dedicando a mejorar continuamente los procesos futuros sin dejar de lado todos los registros que PSP ha dejado su corta existencia. Por lo tanto PSP se convertirá en el predecesor del futuro cuando se analicen a fondo todo el historial que ha dejado a su paso.

Hasta ahora todo apunta a que éstos procesos se continuarán utilizando por lo que esto no es cosa de auge, seguramente estos procesos se convertirán en algo obligatorio para los programadores y las empresas que se dedican a desarrollar software. Esperemos que así sea y siga existiendo software útil y de buena calidad.

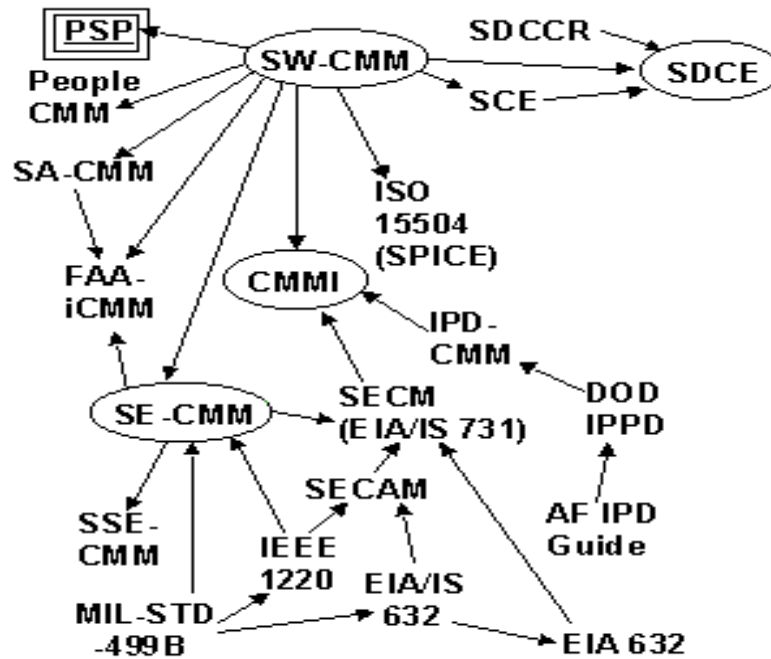


Figura 2.3. "CMM y sus procesos" [GARCÍA; 2001]

Se provee esta información para que haya conocimiento de cuál es el "corazón" del organismo que completa a CMM y de donde surge cada proceso existente y futuro. Ahora sólo resta esperar al futuro que seguramente será promisorio para la industria del software y sus múltiples usuarios.