CAPITULO 1. INTRODUCCION

"La automatización de malos procesos sólo agrava más la ineficiencia" [HAMMER; 90].

La tecnología en la actualidad avanza a pasos cada vez más grandes y difíciles de rastrear. Tanto así que en algunas ocasiones llega a superar el control que el ser humano debe ejercer sobre ella. Esto no representa una excepción para los programas que se vuelven cada vez más complejos en todos los aspectos del ciclo de vida del software.

Este ciclo de vida es algo que tienen en común cada software que se desarrolla y es algo por lo que inevitablemente deben de pasar. El software desarrollado o que se desarrollará tiene normalmente el siguiente ciclo de vida:

1) Análisis de requerimientos.
2) Diseño.
3) Programación.
4) Pruebas.
5) Implantación.
6) Operación.

En algunas metodologías y procedimientos este ciclo de vida llega a sufrir algunos cambios, sin embargo en esencia ha sido, sigue y seguirá siendo lo mismo. Los programadores deben de tener en cuenta que este ciclo de vida requiere de un control absoluto en cada paso que se dé. El software desarrollado de esta manera tiene una gran calidad y un ciclo de vida más largo. Sin embargo la cultura de desarrollo de software en México y en muchos países no están conscientes de este aspecto.

Este hecho levanta una expectación mayúscula debido a que poco a poco se degrada la calidad del software desarrollado.

Las estimaciones varían conforme se pierde el control y el registro de los programas anteriormente desarrollados y esto impide que exista un seguimiento lógico a cada tipo específico de problema. [PAULK; 95]

En la actualidad muchos programadores tienen la idea errónea de que llevar un registro representa una pérdida de tiempo muy grande y que por lo tanto es innecesario realizar esta tarea. Ha llegado a observarse que cuando se desarrollan programas pequeños y que no se lleva un registro, se resuelven de manera más rápida que si se llevara éste. Sin embargo, puede que exista una mayor rapidez de desarrollo pero la calidad resulta ser menor así como la eficiencia y eficacia del programa desarrollado.

Por lo tanto no cubre con los máximos requerimientos del ciclo de vida. También es cierto que el próximo programa que se desarrolle no contará con una estimación de todos los parámetros que se involucran en la primera etapa de desarrollo. Esto representa que a la larga es mejor desarrollar programas con una metodología o un modelo bien definido.

Actualmente existen diversas metodologías de ingeniería de software que guían al programador a dar un mejor seguimiento a sus programas, por mencionar algunos ejemplos se encuentran: COCOMO, COCOMO II, UML, Ley de Parkinson y muchas otras metodologías que actualmente existen y que algunas son nuevas y se adaptan a la tecnología actual.

Por si no se tiene conocimiento del significado de lo que es la ingeniería de software podemos citar lo siguiente:

> "*Es la disciplina cuyo fin es la producción de software libre de fallas, entregado a tiempo, dentro del presupuesto y que satisfaga las necesidades del cliente.*" [SCHACH; 98]

También existen certificados de calidad que evalúan el software desarrollado y las metodologías que los programadores siguen. Sin embargo estos certificados muchas veces no profundizan lo suficiente en cada uno de los niveles de desarrollo y no ponen especial atención a la célula básica por la cual el software no podría existir y nos referimos al *programador.*

Ahora, gracias a la especial preocupación de un hombre que puso todo su esfuerzo en crear y desarrollar un modelo que atendiera todas las etapas y todos los elementos que intervienen en el desarrollo de tecnología. No me equivoco al utilizar esta palabra ya que en la actualidad toda la tecnología existente tiene un software infundido en su "*esencia*".

El nombre de este hombre es: Watts S. Humphrey y su modelo recibe el nombre de "Capability Maturity Model (CMM®)" o Modelo de Capacidad de Madurez por su traducción al español y CMM por sus siglas en inglés. Haremos referencia a este modelo por sus siglas CMM. El "*nacimiento*" de CMM se da en el año de 1986 cuando el Software Engineering Institute (SEI) junto con MITRE Corporation, buscaron mejorar el proceso de software y desarrollaron un marco de trabajo que llamaron proceso de madurez.

Este proceso está basado en el concepto de la administración de la calidad total Total Quality Management (TQM) por sus siglas en inglés.

Éste concepto cuenta con cinco etapas evolutivas que se enfocan en la implementación de prácticas de calidad. CMM es, en pocas palabras, una aplicación de TQM para software.

En el año de 1989 Humphrey que era en ese entonces director del SEI, publica en su libro un marco de trabajo definido por cinco niveles de madurez.[GARCIA; 2001]

El modelo de capacidad de madurez (CMM), provee a las organizaciones de software una guía de cómo aumentar el control de sus procesos de desarrollo y mantenimiento de software. Fué diseñado para guiar a las organizaciones de software en la selección de estrategias para el mejoramiento de procesos mediante la determinación de la madurez de los procesos actuales e identificando los elementos más críticos de la calidad de software y del mejoramiento de procesos.

La arquitectura del modelo está compuesta por niveles que describen la madurez de la organización y por áreas claves de procesos KPAs (*Key Process Areas*). Los diferentes niveles permiten medir la madurez del proceso y evaluar el potencial de él, como también ayudan a una organización a priorizar sus esfuerzos de mejoramiento.

En la Figura 1.1 se da un marco detallado de lo que contiene el modelo de capacidad de madurez en cada uno de sus niveles.

El PSP (*Personal Software Process*) es una técnica probada para mejorar el funcionamiento y la productividad individuales de los ingenieros.[HUMPHREY; 95].

Cabe destacar que PSP puede ser un buen, si no es que el mejor, complemento para cumplir con los requerimientos que nos exige CMM en cada uno de sus niveles. Esto serviría para certificarse de una forma mucho más rápida y que además, se acopla a la medida de cada empresa que quiera alcanzar dicha certificación.

PSP también surge de la necesidad que tienen los individuos programadores de automatizar sus procesos.

NIVEL		ENFOQUE	AREAS CLAVE
5	Optimizado	-Mejora contínua de los procesos.	-Prevención de defectos. -Innovaciones tecnológicas. -Control de cambios al proceso.
4	Administrado	-Calidad del producto y del proceso.	-Mediciones y análisis del proceso. -Administración de la calidad.
3	Definido	-Proceso de ingeniería.	-Enfoque al proceso de la organización. -Definición del proceso de la organización. -Revisiones. -Programa de entrenamiento. -Coordinación entre grupos. -Ingeniería de productos de software. -Administración integrada del software.
2	Repetible	-Administración de proyectos.	-Planificación y control del proyecto. -Administración de la calidad. -Administración de la configuración. -Administración de los requerimientos.
1	Inicial	-Inicio del proceso.	

Figura 1.1 Niveles de capacidad de madurez. [LEWIS; 2002]

Por lo tanto es deseable, mas no obligatorio, hacer uso de PSP y demás procesos creados en torno al modelo CMM, para cumplir con todos los requerimientos de dicho modelo.

Ahora bien, PSP también puede ser utilizado de manera independiente. Esto es, para ayudar al programador a llevar registros de sus proyectos cualesquiera que estos sean. Asímismo PSP se puede ajustar a la medida de otras metodologías ya que al fin y al cabo el ciclo de vida del software es el mismo a todos niveles.

Ya que se ha visto que PSP puede trabajar conjuntamente a CMM al igual que de manera independiente, sólo falta mencionar si puede trabajar de forma combinada.

La respuesta a tal cuestionamiento es sí. PSP puede funcionar de esta manera, al fin y al cabo PSP es sólo una herramienta que cualquier programador puede utilizar para llevar a cabo sus planes. El uso de esta herramienta depende del ámbito en el que se quiera utilizar y del propósito que le dé el programador.

Estos propósitos son 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 más adelante.

1.1 DEFINICION DEL PROBLEMA

Para realizar mejoras en un proceso, es necesario realizar un análisis objetivo en cuanto a las medidas que se tomarán y en base a esto, hacer los cambios pertinentes que pueden resultar difíciles. En algunos casos la persona está acostumbrada a ciertos hábitos y no alcanza a ver nada defectuoso en ellos.

Es por esto que después de dos décadas de promesas incumplidas sobre aumentos de la productividad y de la calidad de aplicar nuevas metodologías y tecnologías del software, la industria y las organizaciones del gobierno están viendo que su problema fundamental es la inhabilidad de manejar el proceso del software.

El problema que se genera a partir de este rezago tecnológico puede tener grandes consecuencias en todos los ámbitos.

Limitándonos solamente al desarrollo de software nos damos cuenta que la cultura de programación es más avanzada en México que en otros países mas subdesarrollados. No cabe duda que tenemos las herramientas necesarias para ponernos a la vanguardia en lo que a procesos de software se refiere.

La cultura del mexicano se limita en hacer los programas fáciles y rápidos, no importando si se dejan registros o mas aún, si se buscan éstos para posteriores comparaciones con lo que se desarrolla. Es cierto que estos procesos pueden resultar largos y tediosos y al principio retrasar de manera considerable cualquier proyecto.

Pero también es cierto que la eficiencia de todos estos procesos está comprobada y que el retraso se debe más que nada a la inexperiencia del programador para seguirlos.

Por lo tanto la problemática se centra en que no existe hasta ahora un organismo *universal* que enseñe a esta cultura a disciplinarse en cuanto a desarrollo de software se refiere.

1.2 OBJETIVOS GENERALES

Dentro de las metas a cumplir en esta investigación está el de comprometer al programador con la mejora continua de su desempeño y a la calidad del software que produce, así como su tiempo. Esta mejora se buscará por medio de un tutorial que muestre paso por paso a los programadores a realizar dicha mejora de forma gradual. Dicho tutorial se encontrará en internet almacenado en los servidores de la Universidad de las Américas - Puebla.

Es importante realizar una investigación detallada de todo el proceso que conlleva la implementación del PSP y de cada uno de sus niveles.

En realidad la implementación no es mas que desarrollar una disciplina en los programadores para que hagan uso del proceso personal de software. Por lo tanto es una enseñanza que se implementa en cada uno de los que desarrollan software.

Asimismo, cumpliendo con estos objetivos es cuando poco a poco los programadores se van a interesar en este proceso y lo lleguen a formar como parte importante de la cultura diaria que practiquen.

1.3 OBJETIVOS ESPECIFICOS

Debido a que es necesaria una investigación profunda del tema, fué necesario consultar de forma exhaustiva el proceso de implementación completo del PSP.

Debido a su extensión se cubrieron los mayores puntos posibles que se involucran en PSP, los detalles de estos puntos se dan en los capítulos posteriores.

Los puntos que se cubrieron para alcanzar estos objetivos son

1. Recabar toda la información que esté relacionada con PSP.
2. Buscar empresas mexicanas que lleven a cabo dicha tarea, la tarea de infundir PSP en sus trabajadores y/o que hagan uso de este proceso tan útil llamado PSP.
3. Una vez recabada toda la información necesaria, ésta se recopilará para sacarle el mejor provecho, ordenarla y analizarla.
4. Finalmente plasmar este proceso lo más que se pueda en un tutorial que se encontrará en internet.

Toda esta información se procesó de tal forma que ésta se adaptó a las características de las empresas mexicanas y a sus trabajadores.

El primer paso se complementó con la consulta de libros que tratan del tema y se resumió la información más importante y relevante.

A continuación se buscó por medio de entrevistas, visitas o vía e-mail; la información que ayudó a realizar una adaptación inicial del tutorial para las empresas mexicanas, con esto se cubre el segundo punto.

Por último se hizo la construcción del tutorial, de forma que sea accesible y lo más específico y simple posible; esto para que el tiempo de implementación se acorte y facilite a los trabajadores su tarea. Uno de los mayores retos que el ingeniero de software tiene que afrontar, se encuentra en el diseño y la programación de aplicaciones de software. Otro reto consiste en el manejo de un método personal para el proceso de ingeniería de software y así trabajar más eficientemente y productivamente.

La necesidad de impartir una disciplina en el proceso de software está basada en que generalmente los ingenieros o estudiantes no entienden la importancia de la administración del tiempo, calendarización y administración de la calidad. Es debido a esto que se consideró como punto importante que exista una guía de fácil acceso para todos aquellos programadores que quieran innovar sus procesos y hasta automatizarlos.

Este objetivo descubre solamente la punta del "*iceberg*" ya que este tema está en constante desarrollo y amerita que se infunda como una cultura.

Como proyecto futuro faltaría ahondar en el contenido de todo el proceso personal de software, tarea que se busca dejar a la posteridad para complementar este inicio y ¿Porqué no? hasta este proceso podría ser una materia de enseñanza para todos aquellos que se interesen en el desarrollo de proyectos.

1.4 ALCANCES Y LIMITACIONES

Dentro de las limitantes con las que me encontré fueron las entrevistas o visitas a las empresas que tienen el modelo implementado, ya que si no se llevaban a cabo de una forma impersonal, pudo ser que la información sufrió de alteraciones y por lo tanto no se pudo alcanzar el objetivo deseado, cosa que no ocurrió.

También una limitante fué que muy pocas empresas a nivel nacional cuentan con la implementación de CMM y menos aún utilizan el proceso personal de software.
Por lo tanto esto impidió que haya un alcance máximo de todas las etapas del PSP.

Gracias a la internet el acceso a la información es mucha y nos pudo ayudar a realizar un enfoque comparativo de las implementaciones a nivel nacional con las de nivel internacional.

1.5 CONTENIDO DEL DOCUMENTO

A manera de resumen, se va a describir brevemente el contenido de cada capítulo. El capítulo 2 explica la historia de PSP desde su nacimiento hasta la actualidad.

Asimismo resalta cada nivel de PSP y su relación con CMM. El capítulo 3 marca un análisis completo de los niveles de PSP 0 y PSP 0.1, se juntan estos niveles en este capítulo debido a la similitud que existe entre ellos. El análisis de estos niveles incluyen los requisitos y los formatos que se manejan en estos procesos para finalizar con las conclusiones tanto personales como de personas experimentadas para que el lector forme su propia opinión.

El capítulo 4 se enfoca en el nivel PSP 1 que marca el principio de una nueva fase de desarrollo y de requerimientos para quien siga el proceso completo, en este caso se refiere al conteo de un programa y a la administración de la calidad que el programador debe de saber manejar en este nivel.

Debido a que PSP es muy extenso, el capítulo 5 presenta los comentarios generales de los niveles de PSP faltantes (1.1-2-2.1-3) y un marco general de los formatos que manejan estos niveles. Finalmente, el capítulo 6 describe el tutorial que completa esta investigación y que es una herramienta más de la ingeniería del software. Este tutorial solamente se aplica completamente a los primeros 3 niveles de PSP y muestra un panorama general del proceso completo.

El documento finaliza con las conclusiones personales y con los resultados que se obtuvieron a partir de un proyecto para que el lector comprenda la importancia de seguir este proceso.